UNIVERSITY OF THE PHILIPPINES MANILA

COLLEGE OF ARTS AND SCIENCES

DEPARTMENT OF PHYSICAL SCIENCES AND MATHEMATICS

CORDIE: A Web-based Collaborative Real-time Diagram Editor

A Special Problem in Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Computer Science

Submitted by:

Gem M. Villarico

March 2013

University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

CORDIE: A Web-based Collaborative Real-time Diagram Editor

A Special Problem in Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Science in Computer Science

Submitted by:

Gem M. Villarico

2007-10919

March 2013

Table of Contents

ACCEPTANCE SHEET

The Special Problem entitled "CORDIE: A Web-based Collaborative Real-time Diagram Editor" prepared and submitted by Gem M. Villarico in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____
Aldrich Colin K. Co, M.S. (candidate)
Adviser

EXAMINERS:

		Approved	**Disapproved**
1.	Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2.	Avegail D. Carpio, M.S.	_____	_____
3.	Richard Bryann L. Chua, M.S.	_____	_____
4.	Geoffrey A. Solano, M.S.	_____	_____
5.	Vincent Peter C. Magboo, M.D., M.S.	_____	_____
6.	Ma. Sheila A. Magboo, M.S.	_____	_____
7.	Bernie B. Terrado, M.S. (candidate)	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____
Avegail D. Carpio, M.S.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

_____
Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

_____
Reynaldo H. Imperial, Ph.D.
Dean
College of Arts and Sciences

ABSTRACT

Distributed software engineering has become the trend in accomplishing software projects. One of the difficulties with this is that it requires team members to participate in the software design, which is an important stage in a software development process, even if they are geographically separated. Instead of the traditional way of gathering around and taking turns sketching on a whiteboard, software engineers adopt various tools to participate in the software design. However, many of the software modeling tools today are insufficient to support the fluidity of interaction among collaborators in the design process.

In order to address the problem CORDIE was developed. CORDIE enables its users to create UML diagrams through the web browser. It also allows simultaneous editing of a single diagram by multiple users. It was concluded that CORDIE can act as a tool in collaborative software design. This also means that users of CORDIE can participate in the software modeling process even if they are in different locations.

KEYWORDS: distributed software engineering, software design, software modeling tool, software diagram, diagramming tool, collaboration, real-time collaborative, web browser

I. Introduction

A. Background of the Study

The early design phase is an important stage of the software development process. This is where software designers collaboratively draw visual models or software designs and in the process come up with an agreed plan. When designers have sound visual models and a clear understanding of the intended solution, it won't be hard to achieve high-quality software.

Presently, software designs are illustrated through UML diagrams. The Unified Modeling Language (UML) has become the standard visual modeling language for software systems [1, 2, 3]. In fact, mainstream software design tools and UML editors refer to the same set of tools [4]. The importance of designs in the development of software can't be underestimated. Evidently, software models provide an easier way to communicate information. Developers would be able to immediately understand a software project by just looking at the visual model. At the same time, it will be equally useful for users or customers who won't be able to understand lines of code. Communication through diagrams moreover lessens the time spent on doing investigative work, meetings and presentations. Therefore, the cost of a software project is reduced dramatically. Another benefit gained from making use of software diagrams is the reduction of time and quality risks. Time and quality risks appear because a task might seem easy on the surface but is actually more complex than initially suspected. The complexity of the problem is likely to be understood better by the engineers through diagrams [1].

The creation of visual models in the design stage of the software development lifecycle involves the participation of a number of designers. According to [5] the design activity includes three important aspects: cooperation among the designers, action which is the actual contact of the designers with the tools, and the usage or meaning of the actions done. Whitehead suggests that design is also an

argumentative process. Software engineers will have different views regarding the structure of the system and competition is unsurprising since not all views will be agreed to by everyone [4]. One of the goals of collaboration is to find out and correct errors [4]. It is critical to discover errors as early as the design phase since the cost of correcting them become more and more expensive with each phase wherein they are not detected and resolved [6].

Collaborative modeling sessions often happen in meeting rooms, where software designers gather around a whiteboard, communicate their ideas among the group and take turns at sketching on the whiteboard. Formal UML syntax isn't always strictly followed. And as the discussion progresses, the software designs naturally evolve [5].

B. Statement of the Problem

Nowadays, distributed software engineering is becoming an increasingly popular way of accomplishing software projects. With this method of software development, members of the software engineering team are operating on different locations around the globe and just rely on computer-based collaboration support systems to coordinate their work [7].

Since collaborative software design is still an important activity even in distributed software development, the geographic distribution of software designers becomes a difficulty. Conducting face-to-face meetings is not really a good idea since it would require the designers to travel which is time-consuming and expensive [8]. For that reason, a tool is needed to support distributed modeling.

However, most development tools today are designed only for single users. There must be a way to allow for and maintain the fluidity of interaction among collaborators since distributed modeling generally involves "intense coordination and communication within the development team" [2].

In addition, the following are some of the important requirements that a software design tool must satisfy according to [9]:

- It should support the remote creation, modification and viewing of the diagrams.
- It must be able to handle concurrent editing of documents and provide concurrency control.

Of course, there are a few already existing visual model creation tools with synchronous editing capabilities (i.e. changes made by users are shown to others in real-time) such as Rosetta [9], SLIM [2], and Pounamu [11]. However they are all implementing a locking pattern as the concurrency-control mechanism. Locking can greatly affect the usability of a groupware. Waiting for locks could be frustrating and it makes the interaction seem less natural.

Another problem with some of the existing diagram tools is that they are somehow restricted by their design or configuration. For instance, Microsoft Visio 2010 is a diagramming program that enables users to create dynamic data-driven visuals and to share them in real-time [12] but it is only available for Windows and not on other operating systems [13]. However, according to [2], "ideally, collaboration partners should be able to participate in the modeling process regardless of the hardware and software equipment they use."

With the trend towards distributed software engineering seems to continue and the fact that broader participation in the design process is being predicted in the future [4], a tool that can fulfill the collaborative software design requirements of physically separated teams has become truly necessary.

C. Objectives of the Study

The objective of the special problem is to develop a web-based UML diagram creation software that allows the users to collaborate with one another in real-time. Essentially the system can be divided into the following functionality areas:

1. Log in Management

   Users can log in by entering their registered username and password. If they are new to the system, they first have to create an account by signing up. Only after successfully logging in can a user make use of the system. Afterwards, he or she can choose to log out.

2. Account Management

   A user can view and update his or her profile and other account information such as name, password, display picture, etc.

3. Diagram Management

   It is in this area where a user can maintain the diagrams that he/she created or is a collaborator on. It can be further divided into the following functions:

   a. View List of Diagrams

      The system shows the list of diagrams which includes both those that are created by the user and those that linked him as a collaborator.

   b. Create, Open and Delete Diagrams

      In the list of the diagrams, the user can straightforwardly add a new diagram, delete selected diagrams, and open a specific diagram.

4. Diagram Editing

The system's main purpose is to facilitate the editing of a diagram by several users which happens in real-time. In order to accomplish this, it must allow the users to do the following:

a. View the Changes as They Happen in Real-time

Each collaborator's modifications to the shared diagram are reflected to everyone's copy in real-time.

b. Use Drawing Tools

A toolbox is available to aid the user in editing a diagram. This toolbox contains the basic diagram editing tools (e.g. pencil, select and rectangle tools) and the different tools for adding UML notational elements (e.g. class, package, and various relationship arrows). In addition, the user can edit the properties (such as position, color, size and font) of a selected object through the properties box.

c. Drag and Resize Objects with the Mouse

This adds some degree of freedom and the user can explore different possibilities first before selecting the object's final position or size.

d. Get Notified of Recent Changes

In the same page where the user edits the diagram, an area is reserved to place textual description of recent changes. This part is also updated in real-time.

e. Download Diagram as Image File

The system allows the user to download a certain diagram's image file.

5. Diagram Storage

Every time the diagram is modified, it is automatically saved as a file in its XML format. While editing a particular diagram, a user can choose to save it as another diagram with a different title.

6. Collaborator Management

   After opening a diagram, the user will see which collaborators are also currently editing it and who are allowed to edit the diagram. If he/she is the creator, he/she can add more collaborators by entering their usernames and/or remove selected collaborators.

D. Significance of the Study

The application will be particularly useful in supporting distributed software modeling. It provides the necessary tools to add different UML notational elements in creating visual models. Designers will be able to simultaneously edit a single diagram even if they are globally distributed and every user's changes are shown in real-time. One significant aspect of the application is that no locking is involved. This implies a more natural group interaction for the users. Moreover, it has the advantage of being web-based. There is no need to install anything to be able to start using it. Thus, it is not restricted by the operating system of the users. It is readily available, provided that they have a web browser that supports the application and an internet connection.

The application can also assist other collaborative drawing activities other than software design. For instance, it can help in collaborative construction of mind maps which sometimes happen during group brainstorming sessions. It can also act as a communication tool since there are cases that some information can be better expressed as sketches rather than in words.

E. Scope and Limitations

1. The real-time collaborative feature is only available in the page where users edit a diagram. Real-time updates may not be available in other parts of the system.

2. The shapes and symbols that can be included in the diagrams are limited by the available drawing tools in the application.
3. The application cannot generate source code from a constructed software model and vice-versa.
4. The tool does not check whether software models generated are syntactically correct or not.
5. The application can only be used to create 2D diagrams.
6. The application cannot import diagrams generated from other applications.
7. The diagram is of fixed size.
8. Collaborators who are off-line while the others are editing the shared diagram are not notified in any way (such as e-mail notifications) about the changes that the other users made.
9. Users will not be able to retrieve previous versions of a diagram.
10. A user cannot view a diagram except if he/she created the diagram or if he/she is a collaborator on that diagram.
11. Users who wish to become a collaborator of a diagram will have to message its creator outside of the system.
12. The application does not provide an instant messaging component which can significantly improve the collaborative work of users.
13. Just like other groupware, the system might give users the wrong idea that their collaborators are present during a group session when in actuality they are not.

F. Assumptions

1. The creator of a diagram already knows the usernames of those that he/she wants to add as collaborators.
2. The application is opened in a browser that supports HTML5 and JavaScript.
3. Users may also be communicating in other ways such as audio and video communications, and instant messaging.

II. Review of Related Literature

The many advancement in collaborative tool applications have significantly improved the way that people work together. Collaborative tools are usually used for communication, sharing of documents and/or providing awareness of other users.

According to Whitehead, in order to support the collaboration or coordination of project work, people have adopted various communication and collaboration technologies such as telephone, teleconferences, email, voice mail, the Web, instant messaging, and videoconferences. Assessing the impact of the introduction of these and other collaboration technologies in a project is difficult and at most times subjective. Knowing the pros and cons of specific collaboration tools is important in determining whether it really is beneficial or not [4].

One particular collaborative tool is GROVE (Group Outline Viewing Editor). GROVE is intended to be used by a group of users simultaneously editing a textual outline. An outline can be viewed through a window that is replicated over a set of machines. A user can modify the outline by performing one or more standard operations such as insert, delete, cut and paste [40].

Another is the Jupiter system-- a multi-user virtual environment meant to facilitate collaboration among distributed group members. It supports sharing of documents, tools and even live audio and video communication. Users who are familiar with programming can also create their own sharing tools using the Jupiter windowing toolkit [38].

The idSpace platform supports the real-time collaborative sketching of group members in a creativity session. The real-time synchronization of the sketching editors was managed through Comet

and reverse AJAX techniques. The idSpace platform encourages collaborative sketching since creativity techniques are better explored when users are able to mash up their work with others [42].

Since software design is inherently a collaborative activity, several studies have aimed to improve collaborative modeling in software projects. For instance, SUMLOW and Knight are diagramming applications for the software design process in face-to-face settings.

SUMLOW is a Unified Modeling Language (UML) diagramming tool that uses an e-whiteboard. The tool allows designers to sketch UML visual modeling language constructs, diagram components and even hand-written text. The sketches can be made formal into computer-recognized and drawn diagrams, and even exported to a third party CASE tool. One advantage of using a whiteboard captured by SUMLOW is its capability of allowing multiple designers to gather around, discuss changing designs and take turns at sketching and annotating designs on the whiteboard [3].

Knight is another Computer Aided Software Engineering (CASE) tool that can aid the collaborative work of software engineers in object-oriented modeling. The Knight tool relies on an electronic whiteboard to get its input. Although many users can use the tool using multiple pens, it can only take an input from one pen at a time. Moreover, Knight supports drawing of both formal UML diagrams and informal sketches. However, the problem is that the user must switch between “Freehand mode” and “UML mode” so that the tool can correctly interpret their sketches as either formal or informal. This switching of modes is obviously annoying for the users and makes the interaction with the tool less natural. The advantage of using Knight is that the interaction style is similar to that of traditional whiteboard while also allowing users to easily modify their diagrams [5].

In distributed software modeling, collaborative tools play an important role in coordinating the work of a number of users who are at separate locations. Although there are many existing diagramming applications, only few have support for real-time collaboration.

An example of a diagramming tool with real-time collaborative feature is SLIM. SLIM is a browser-based tool that can assist in distributed software modeling processes. The application incorporated a combination of SVG and VML for displaying the graphical elements on web pages. SLIM has implemented a Comet architecture for the exchange of messages between clients. In addition, a locking pattern was chosen as the concurrency control mechanism. The development of such tool has shown that it is possible for users to actually engage in collaborative work using only modern web browsers [2].

A Pounamu plug-in collaborative editing component is presented in [11]. The plug-in collaborative editing component can handle synchronous diagram editing of multiple users and even capture and replay of diagram edits. The synchronous and asynchronous collaborative editing features of the plug-in component have been used on ER diagramming tool and UML tool. Therefore, users in a distributed setting can simultaneously edit a visual model together. For synchronous editing, the plug-in component implements locking to control concurrent user edits [11].

Rosetta is a light-weight tool for object-oriented design. Rosetta enables users to view and edit design documents anywhere over the Internet since diagram documents are written in HTML and are stored on a server. UML diagrams are edited using the Rosetta editor which is implemented as a Java applet. In addition, the code implementation can be checked against the design documents using the checker tool. Group work on diagrams is supported but only one person at a time can modify a

particular diagram. And those who wish to view updated diagrams will have to reload their HTML pages in their web browsers [9].

Calico is a sketch-based tool for software design. Just like most drawing tools, Calico allows its users to draw, erase, choose pen color, undo and redo. Calico, however, has additional functionalities which make sketching experience more fluid for the software designers. For instance, the tool organizes all canvases into a grid layout so that user can easily arrange the canvases or access a specific canvas. Moreover, it has support for remote collaboration; users can work together on a single canvas or they can work on different canvases and merge them into a single canvas [10].

Of course a popular example is Microsoft Visio 2010 which is a drawing software package that one can use for drawing and graphing charts and designs including UML diagrams. However it is only available for Windows and not on other operating systems [13]. Microsoft Visio 2010 is a diagramming program that has tools that enable users to create dynamic data-driven visuals and to share on the Web in real-time [12].

III. Conceptual / Theoretical Framework

A. Collaborative Software Design

The creation of models early on in a software development project is one of the collaborative activities that software developers engage in to form an understanding of the problem.

Traditional / Face-to-face

The traditional method of collaborative software design happens in meeting rooms where designers gather around a whiteboard onto which diagrams are drawn. Whiteboards are used because of their simplicity and flexibility. Though the Unified Modeling Language (UML) is the standard graphical notation for software models, working with a formal notation is often a problem to some especially to the inexperienced developers. As a result, informal notations are commonly used and frequently mixed with formal drawings. It is also not unusual that software designers leave their diagrams incomplete in order to save time, to save space, to improve understandability, or just to show lack of knowledge in a certain area. When it comes to making changes in the diagrams on the whiteboard, designers are forced to erase previous sketches and redraw them. Therefore, editing diagrams is time-consuming and laborious. Cooperation among participants during design is observable. The discussion takes place while they take turns sketching on the whiteboard. Usually only one person is drawing at a time, although sometimes interruptions from the other designers can take place. Lastly, visual models drawn on the whiteboard are temporary. The designers redraw the important ones on a paper or they use a digital camera to capture the diagrams [5].

Distributed

Software engineering collaboration revolves around the creation of the various formal and semi-formal artifacts such as the requirements specifications, architecture diagrams, UML diagrams, source

code and bug reports. In order to support collaboration, software engineers have adopted a number of technologies that allow the development of each of these artifacts. These include collaborative diagram creation tools for authoring software designs [4].

Distributed software design relies on design tools since it's the reasonable way to support communication within the team. Moreover the electronic format of designs makes them easier to be shared among several participants. Distributed modeling requires that design tool employed by the team can be used to create, edit and view the design documents. Another consideration is that the design tool should also allow several users to concurrently modify designs. Ideally, final designs in a software project are consistent with the implementation of the system. The accuracy of information contained in visual model documents is within the responsibilities of the software engineers. It is also up to them to check whether the implementation is in agreement with the designs [9].

B. Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a family of graphical notations, backed by a meta-model (a defined diagram, e.g. class diagram), that can be employed to describe and design software systems. The UML 2 has 13 official types of diagrams: activity, class, communication, component, composite structure, deployment, interaction overview, object, package, sequence, state machine, timing and use case. The UML standard indicates which elements are usually drawn on certain diagram types, although one often can legally use elements from one diagram type on another diagram type [14].

1. Class Diagram

A class diagram describes the types of objects in the system and their relationship with one another. Class diagram also provide information on the properties and operations of each class and the constraints that apply to the way objects are connected [14].

Class

A class is shown as a rectangle divided into three compartments: the name of the class centered in boldface, its attributes, and its operations. The UML also allows the suppression of the second and third compartments to create more readable diagrams [15].

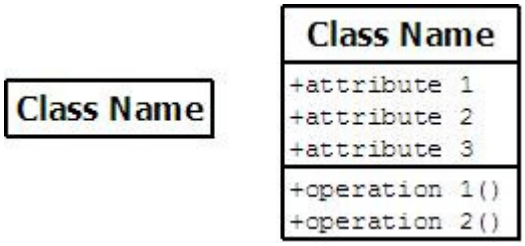


Figure 1. Notations for Class

Association

An association is represented by a solid line that connects two classes. Association can either be unidirectional or bidirectional. In a unidirectional association, the line is directed from the source class to the target class. The name of the property goes at the target end of the association, together with its multiplicity.

Figure 2. Unidirectional Association

Another kind of association is bidirectional. To imply bidirectional association, a double-headed arrow is used instead of a single-headed arrow.

Figure 3. Bidirectional Association

An alternative to labeling an association by a property is labeling it using a verb phrase. This is acceptable and an arrowhead can be added to show direction to avoid ambiguity.

Figure 4. Arrowhead

Generalization

The UML also specifies another type of relationship called generalization. Generalization is used to model inheritance and is represented by an arrow with triangular hollow arrowhead. The source of the arrow is the subclass and the target of the arrow is the superclass.

Figure 5. Generalization

Notes

Notes can also be indicated in the diagrams. A note can stand on its own or it can also be linked to its corresponding element with a dashed line.

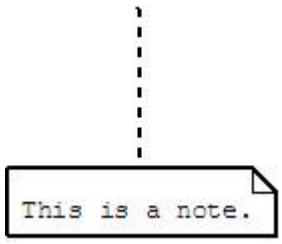


Figure 6. Note

Dependency

A dependency exists between two elements if changes to one element may cause changes to the other. The UML allows one to depict dependencies between elements. The one that may cause changes to the other is called the supplier while the one that depends on it is called the client. Dependency is represented by a dashed arrow with its head pointing towards the supplier and the tail to the client.

Figure 7. Dependency

Constraint Rules

Indicating constraints is another usual task in drawing a class diagram. A way of doing this is specifying multiplicity on either end of a connecting line.

Keywords

Keywords are part of UML because it is used to mark existing UML constructs to indicate that a modeling construct that is different from usual. An example is the interface, a special kind of class. Keywords are usually written as text between guillemets (« and ») or curly brackets ({ and }).

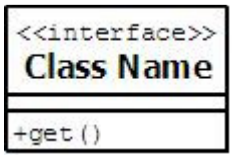


Figure 8. Class with Keyword Interface

Static Operations and Attributes

An attribute or operation that is static is underlined on a class diagram.

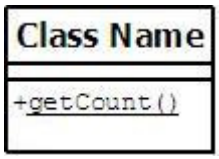


Figure 9. Class with Static Operation

Aggregation

Aggregation is defined as part-of relationship. The UML denotes aggregation by an arrow whose one end is a hollow diamond.

Figure 10. Aggregation

Composition

Composition implies a whole/part relationship [15]. It is often confused with aggregation. But the general rule on composition is "no sharing", that is, an instance of a class can only have one owner even though the class may be a component of many other classes.

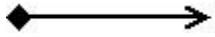


Figure 11. Composition

Abstract Class

An abstract class refers to a class that cannot be instantiated directly. Instead an instance of a subclass is the one instantiated. An abstract class also has one or more abstract operations, those operations that are merely declarations and have no implementation. To indicate that a class or operation is abstract, its name is italicized. One can also instead use the label: {abstract}.

Figure 12. Abstract Class

Realization

The UML expresses the relationship between a class and an interface through what is known as realization. This kind of relationship is modeled by a dashed arrow with a hollow arrowhead pointing from the implementing class to the interface.

Figure 13. Realization

Lollipop Notation

Another way to notate interfaces is by the use of ball icons, also known as lollipops.

Figure 14. Lollipop Notation for Interface

Socket Notation

The socket notation also represents dependency, particularly on an interface expressed in lollipop notation.

Figure 15. Socket Notation

Qualified Association

A qualified association is the UML representation of a programming concept variously known as associative arrays, maps, hashes and dictionaries. A qualifier is used to represent the association between a certain class and another class. Qualified association is shown in a diagram as in figure 16.

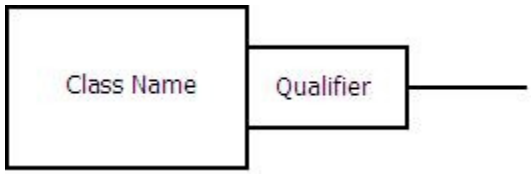


Figure 16. Qualified Association

Classification

Classification is used to show that an object is an instance of a certain type. To show classification, dependency symbol is used with the «instantiate» keyword.

Single and Multiple Classification

In single classification, an object belongs to only one type. While in multiple classification, an object may be described by several types that are not necessarily connected by inheritance. In using multiple classification, to make clear which combinations are legal, each generalization relationship is placed into a generalization set. The generalization arrowhead is then labeled by a discriminator. To illustrate a generalization set, either using several arrows with the same text label or generalizations rolled up into a single arrow is acceptable [14].

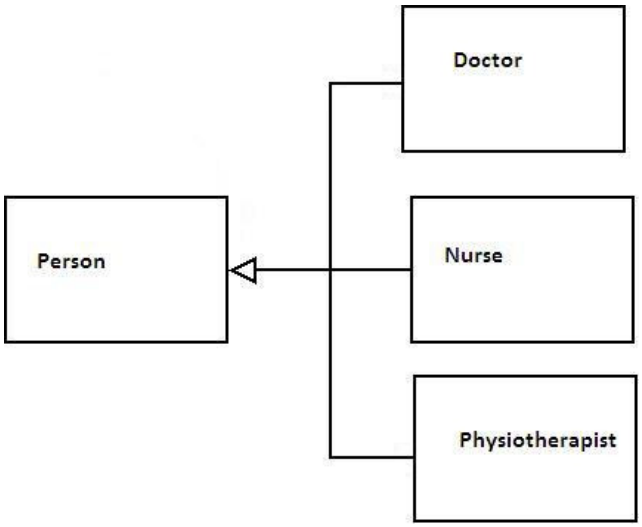


Figure 17. An Example of Multiple Classification

Association Class

In modeling an association, it is sometimes important to include another class because it includes valuable information about the relationship. For this case an association class is attached to the primary association. An association class is drawn like a normal class. The only difference is that the association line between primary classes intersects a dotted line connected to the association class [29].

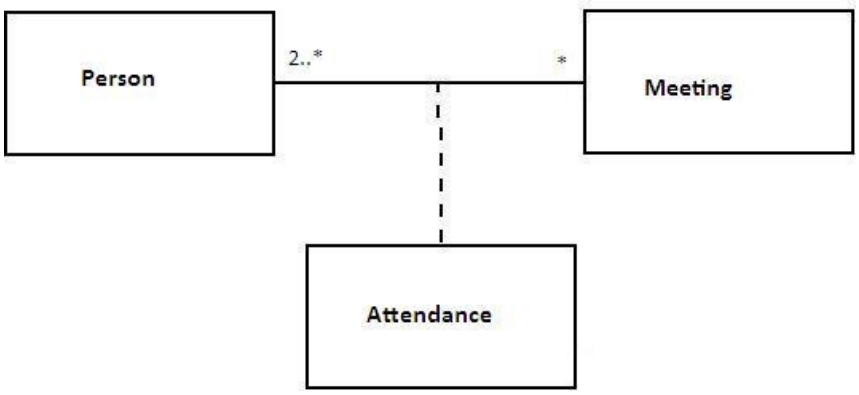


Figure 18. An Example of Use of Association Class

Template Class

Several languages such as C++ have the notion of parameterized class or template. A template class can be declared in the UML by using a diagram as shown in figure 19. The T in the diagram is a placeholder for the type parameter.

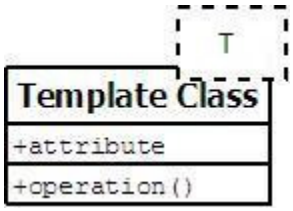


Figure 19. Template Class

Active Class

An active class is a class that has instances that execute and control their own respective thread of control. The notation for active classes in UML 1 is like a normal class with thick border but it in UML 2 it was changed to a class that has extra vertical lines on the side [14].

Figure 20. Active Class

2. Object Diagrams

An object diagram, or instance diagram, provides a snapshot of the objects in a system at a certain point in time. They are useful for showing examples of objects connected together [14]. An object is represented as a rectangle containing the name of the object and its class underlined and separated by a colon. Object attributes and its value can optionally be included in a separate compartment. Objects that are related are linked by lines that are also used in class diagrams [16].

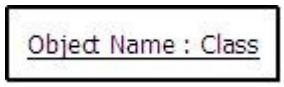


Figure 21. Object

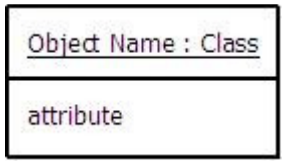


Figure 22. Object with Attribute

3. Sequence Diagrams

The sequence diagram is the most common form of an interaction diagram, a diagram that describes how groups of objects collaborate in a certain behavior [14]. The sequence diagram in addition emphasizes when messages are sent between objects over time [15].

Frame Element

The frame element shows the graphical boundary of a diagram. The notation is as shown in figure 23. The diagram's label is indicated in the top left corner while the actual UML diagram is placed inside the body of the enclosing rectangle [18].

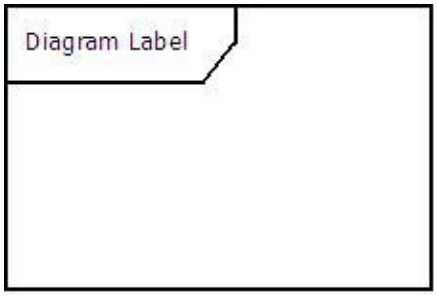


Figure 23. Frame

Class Roles

Class roles explain the way objects behave in context. The object symbol without the object attributes is used to illustrate class roles [17].

Activation

Activation bar or box indicates when a participant is active in an interaction [14].

Figure 24. Activation Bar

Messages

Messages are drawn as arrows to symbolize communication between objects. Different types of arrows are used for different types of messages [17].

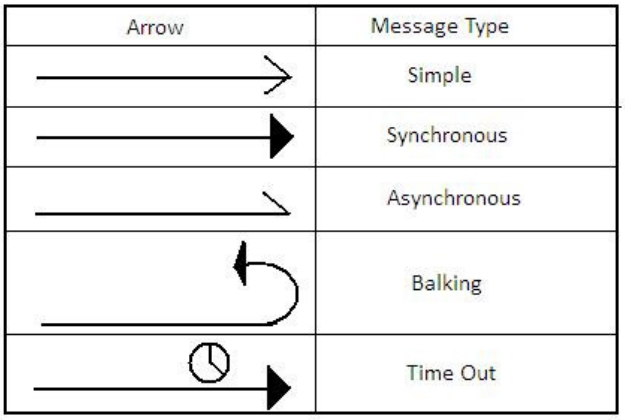


Figure 25. Message Types

Lifelines

A lifeline indicates the object's presence over time and is represented by a dashed line that runs vertically down the page [17].

Figure 26. Lifeline

Deletion

Deletion of a participant is indicated by big X. A message arrow going into the X means that a participant is deleting another, while an X at the end of a lifeline means that a participant is deleting itself [14].

Figure 27. Deletion

4. Package Diagram

Package diagrams are useful in a way that it provides a picture of the dependencies between major elements of a system. Plotting diagrams of packages and dependencies helps one to monitor the dependencies of an application [14].

Package

A package groups elements together into a higher-level unit. It is often used to group classes, though it can also be used to group other constructs in the UML. Packages are shown with a tabbed folder. Either just showing the package name or showing the contents is acceptable [14].

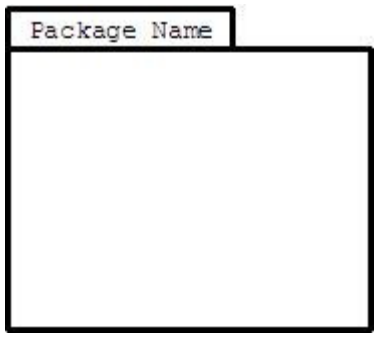


Figure 28. Package

Dependency and Implementing Packages

Since a package diagram shows the packages and their dependencies, the flow of the dependencies has to be clear. Also, a realization relationship often exists among packages where one package defines an interface that can be implemented by other packages. The same dependency and realization arrows are used [14].

5. Component Diagram

Component diagrams show how a system is breakdown into components and their interrelationships through interfaces [14].

Component

Components represent parts that can be purchased and upgraded independently. The difference between a component and a regular class is a little controversial though the difference between their symbols is easily distinguishable. The notation for a component is as shown in figure 29.

Components are connected through provided and required interfaces, usually using the ball-and-socket notation [14].

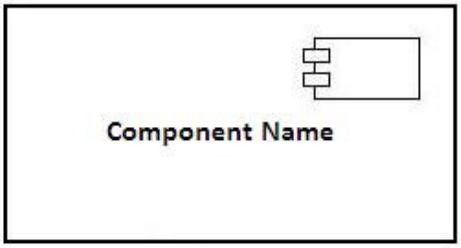


Figure 29. Component

6. Deployment Diagram

A deployment diagram gives picture of the hardware of the system, the software that is run, and the middleware that connects the distinct machines to one another [19].

Nodes

Nodes are one of main items on a deployment diagram. A node is defined as something that can host some kind of software. It can either be a device which is simply a hardware connected to the system, or an execution environment-- a software that can host or contains other software. A node is represented by a three-dimensional box.

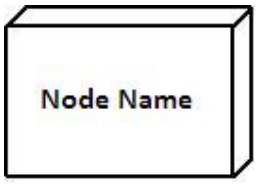


Figure 30. Node

Artifacts

A node can also contain other nodes or software artifacts. Artifacts are usually files that represent software. They may take the form of executables, data files or HTML documents. Artifacts can be shown as class boxes with the document icon or by listing the name within a node.

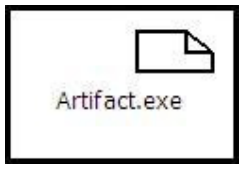


Figure 31. Artifact

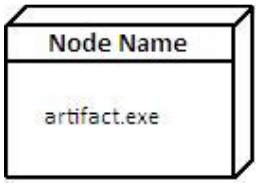


Figure 32. A Node with an Artifact

Communication Paths

Nodes are connected by simple lines called communication paths. These are symbols for how nodes communicate. One can label these paths with some information about the communication protocols that were employed [14].

Figure 33. Communication Path

7. Use Case Diagram

The functionality of a system can be captured using a use case diagram that depicts the system or subsystem of a particular application. A use case diagram shows the actors, the use cases and their relationships with each other.

System Boundary Box

The system boundary box is the rectangle around the uses cases. It specifies the scope of the system.

Use Case

Use cases, shown as horizontal ellipses, are just the functionalities of a system.

Figure 34. Use Case

Actors

An actor can be a person, organization, or an external system that participates in the system. Actors are drawn as stick figures as shown in figure 35.

Figure 35. Actor

Relationships

Relationships among use cases and actors can be association or generalization. Association lines are simple lines with an optional open-headed arrowhead on one end and generalization lines are close-headed arrows with the arrow pointing towards the more general modeling element [20].

8. State Machine Diagram

A common technique in describing the behavior of a system is through state machine diagrams [14]. State machine diagrams, formerly called state chart diagrams, are usually developed to describe how instances of complex classes work [21]. State diagrams are usually drawn not for every class in a system, but rather just for the ones that exhibit interesting behavior where constructing a state diagram will be worthwhile [14].

Initial Pseudostate

An object starts in an initial pseudostate shown as a closed circle. It is not a state but it has an arrow that points to the initial state [14].

Figure 36. Initial Pseudostate

State

States are represented by rounded rectangles as shown in figure 37. In addition, internal activities of a state, or those events that require no transition, can be specified by putting them inside the state box [14].

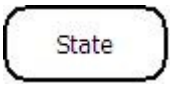


Figure 37. State

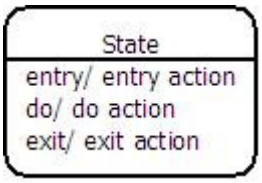


Figure 38. A State with Internal Activities

Transition

Transition, or the movement from one state to another, can be specified by an arrow. Each transition has a label that indicates the event that triggers it and/or the action that results from it [14].

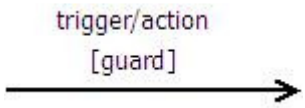


Figure 39. Transition

Final State

When an object reaches a final state, it means that it is completed. The symbol used to denote a final state is a bordered circle [21].

Figure 40. Final State

Superstate

States that have common transitions and internal activities can be clustered together and made substates of a superstate as in figure 41 [14].

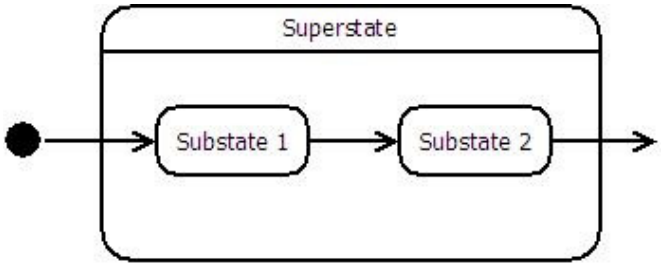


Figure 41. An Example of a Superstate

Shallow History Pseudostate

A shallow history pseudostate "represents the most recent active substate of its containing state (but not the substates of that substate)" and is drawn as a small circle with an "H" [22].

Figure 42. Shallow History Pseudostate

Deep History Pseudostate

A deep history pseudostate "represents the most recent active configuration of the composite state that directly contains this pseudostate" and is drawn as a small circle with "H*" [22].

Figure 43. Deep History Pseudostate

9. Activity Diagram

An activity diagram describes the procedural logic, business process and work flow of a system. What makes an activity diagram particularly useful is its support for parallel behavior [14].

Initial Node

An activity diagram starts with an initial node represented as a filled in circle similar to the initial pseudostate in a state machine diagram.

Figure 44. Initial Node

Flow/Edge

The arrows on the diagram are called flows or edges. Text on a flow is called a condition which must evaluate to true in order to navigate the node or action.

Figure 45. Flow

Fork

A fork marks the beginning of parallel activity. This is signified by a black bar with one flow going into it and several flows leaving it.

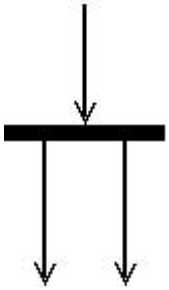


Figure 46. Fork

Join

When a black bar has several incoming flows and one outgoing it is called a join. This marks the end of parallel processing [23].

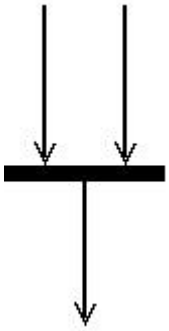


Figure 47. Join

Action

The rounded rectangles on an activity diagram are called actions. An action would execute once all flows coming in and going out triggered [14].

Figure 48. Action

Decision

Decision is represented by a diamond with one incoming flow and one or more guarded outbound flows. When a decision is reached, only one of the outbound flows can be taken. Guards are the Boolean conditions contained in square brackets [14].

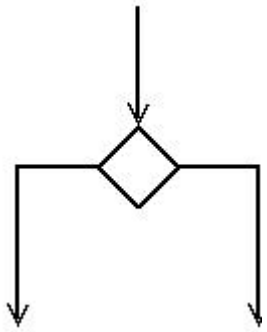


Figure 49. Decision

Merge

Merge is represented by a diamond with several input flows and one output flow. It means that the conditional behavior that was started by a decision has already ended [14].

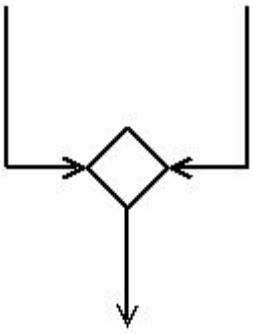


Figure 50. Merge

Final Node

The final node marks the end of an activity diagram. The symbol for this is a filled circle with a border similar to the final state in the state machine diagram [23].

Figure 51. Final Node

Subactivity

One can decompose an action into subactivities. The symbol for subactivity is the same as that for action but it is identified by a rake symbol inside it [14].

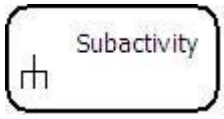


Figure 52. Subactivity

Partitions

Sometimes an activity diagram is divided into partitions to show who does what. This can be done by using swim lanes.

Signal

Actions can react to signals which are events from outside processes.

Time Signal

One kind of signal is the time signal that occurs due to the passage of time. The notation is as shown in figure 53.

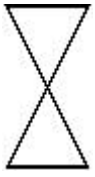


Figure 53. Time Signal

Accept Signal

A signal can be accepted by using the symbol in figure 54.

Figure 54. Accept Signal

Send Signal

Signals can be sent as well using the symbol shown in figure 55 [14].

Figure 55. Send Signal

Connector

Connectors can be used to avoid having to draw very lengthy lines. Connectors are used in pairs: one with the input flow and the other one with the output flow and both have the same label.

Figure 56. A Connector with Label "A"

Pin

Actions can have parameters and though parameters need not be shown they can still be indicated using pins. The notation for a pin is a small square on a side of an action.

Figure 57. Pin

Transformation

Transformation is used to match one parameter to another. The expression for transformation is as shown in figure 58.

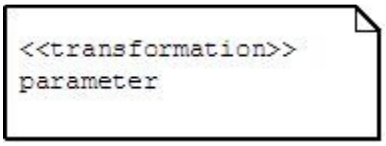


Figure 58. Transformation

Expansion Region

An expansion region is marks the area in an activity diagram where "actions occur once for each item in a collection [14]."

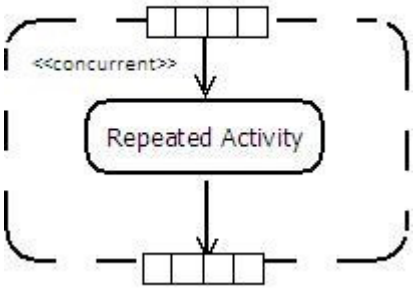


Figure 59. An Example of Use of Expansion Region

Flow Final

When a flow final is reached, it means that a particular flow is ended without terminating the entire activity [14].

Figure 60. Flow Final

10. Communication Diagram

Communication diagrams, formerly known as collaboration diagrams, highlight the data links between the participants involved in an interaction [14]. Communication diagrams use the same notations in other UML diagrams for its elements including classes, objects and associations and this demonstrates the consistency of the UML [24].

11. Composite Structure Diagram

Composite structures are a new addition to the UML. It provides a way to view the internal structure of a class [14].

Part

"A part represents a set of instances that are owned by a containing classifier instance [25]."

Figure 61. Part

Connector

A connector is a simple line that represents the communication link between its source and target.

Figure 62. Connector

Delegating Connector

A delegating connector, shown as in figure 63, is drawn to indicate parts that either need or implement an interface. A part is the target of the delegating connector when it implements an interface, while it is the source when it needs an interface [14].

Figure 63. Delegating Connector

Port

"Ports allow you to group the required and provided interfaces into logical interactions that a component has with the outside world [14]."

Figure 64. Port

12. Interaction Overview Diagram

Interaction overview diagrams show the general idea of the flow of control [26]. Most of the notation used for activity diagrams is also the notation for interaction overview diagrams. One difference, though, is that interaction overview diagrams introduce two new elements: interaction occurrences and interaction elements [27].

Interaction Occurrences

An interaction occurrence indicates an activity or operation to invoke [25]. It is shown as a frame identified with a "ref" in the top-left corner [27].

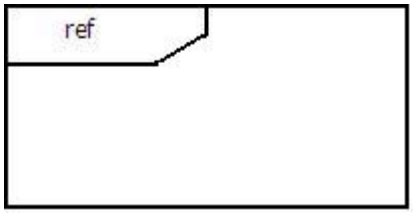


Figure 65. Interaction Occurrence

Interaction Elements

Interaction elements, like interaction occurrences, display a representation of interaction diagrams within a rectangular frame. However, its difference from interaction occurrences is that they show the contents of the references diagram inline [27].

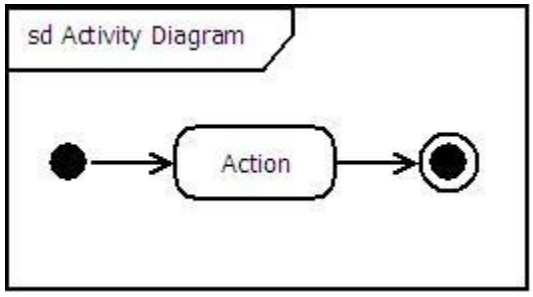


Figure 66. An Example of Use of Interaction Element

13. Timing Diagram

Timing diagrams focus on showing the timing constraints between state changes on different objects. There are a number of ways timing constraints can be shown. State changes can be indicated either by moving from one horizontal line to another or showing them with a cross to another state [14].

C. Basic Diagram Components

The simplest geometric components of pictures are points and straight line segments. Additionally, circles, other conic sections, quadratic surfaces, polygons, and character strings can also be used to construct pictures. Usually, these basic geometric structures are used to describe a scene. Each of them is specified with input coordinate data and other information about the way that object is to be displayed [28].

Line Segments

A line segment is defined to be "a straight line which links two points with known coordinates without extending beyond them" [29].

The basic attributes of a straight line segment are the type, the width, and the color. A line segment's type may be solid, dashed or dotted. Its width is a usually a positive number that indicates the relative width of the line to be displayed. A line's color can also be set when the system provides color options. Sometimes lines can also be displayed with pen or brush selections. Options in this category include shape, size and pattern [28].

Rectangle

A rectangle is a quadrilateral whose interior angles are all 90 degrees. Its location on a coordinate plane is determined by the coordinates of the coordinates of the four vertices. From these coordinates, properties such as width and height of the rectangle can be found [30].

Ellipse

An ellipse is defined by two points, each called a focus. The sum of the distances from the two points to every point on the ellipse is constant. The longest and shortest diameters of an ellipse are called major axis and minor axis, respectively. When the major and minor axis are of the same length, the result is a circle [31].

Circle

A circle is a closed loop in which all points are equidistant from the center. Some important properties of a circle are its center and radius which is the distance from the center to any point on the circle [32].

Polygon

A polygon is defined as "a number of coplanar line segments, each connected end to end to form a closed shape." Polygons that have certain number of sides are usually given names. For example a polygon with 3 sides is called a triangle and that with 5 sides is called pentagon [33].

Text

The appearance of displayed characters depends on several attributes that include font, size, color and alignment. There is choice of font, which is a set of characters with a particular design style such as Courier, Times Roman, Helvetica, etc. Also, the text size can be adjusted by scaling the height and width. The alignment attribute specifies how text is to be positioned with respect to the start coordinates [28].

D. Interactive Picture-Construction Techniques

To aid the interactive construction of pictures, there are a number of techniques incorporated into graphics packages. The following are some of these techniques:

Basic Positioning Methods

To specify a location to display an object, the coordinate values supplied by the input are used. One can interactively select the coordinate positions with a pointing device, usually by positioning the screen cursor. For example, one can select two screen positions in displaying a straight line segment.

Rubber-Band Methods

Objects such as straight lines, rectangles and circles can be constructed and positioned using rubber-band methods. For lines, one first selects a screen position for one endpoint of the line. Then as the cursor is being moved, the line is displayed from the start position to the current position of the cursor. The other line endpoint is set only when a second screen position is finalized.

Dragging

Moving objects into position by dragging them with the screen cursor is a technique often used in interactive construction of pictures. One first selects an object, and then moves the cursor in the direction he wants it to move, and the selected object follows the path. Dragging is useful in certain applications because one might want to explore different possibilities first before selecting a final location in positioning an object [28].

E. Browser-native Technologies for Image Manipulation

There are primarily three technologies for image manipulation that adapt to the browser functionality. These are VML, SVG and Canvas.

VML

The Vector Markup Language (VML) was originally aimed to extend the HTML. Instead, it was made as a separate entity so that it can be easily integrated into webpages. The syntax of VML is almost similar to that of XML and HTML. VML also has the advantage of supporting styling with CSS. It can also be rotated, flipped or grouped. One drawback of VML is that it offers few image manipulation options. Also, it remains a Microsoft proprietary standard since the W3C never made it a web standard [34].

SVG

Scalable Vector Graphics (SVG) is another graphics language that is based on XML. Many concepts are incorporated into SVG: predefined objects, use of XML, and manipulating and grouping of objects.

In using SVG, an SVG root DOM element has to be defined first. This is to where all SVG elements are attached. SVG is also a W3C web standard and is supported by mainstream browsers. SVG is stored as a plain text that can be edited with any text editor. Moreover, it has the advantage of being vector in nature and so it can be scaled at any level. There are also plenty of graphics software packages that can aid in altering SVG images. Other features of SVG are its support for animated and interactive graphics. A subset of SVG called SVG Tiny that can be supported by mobile phones was also developed [34].

HTML5 Canvas

The Canvas is another technology that has support for certain graphics requirements. Basically, one defines a <canvas> element and then he can use JavaScript for drawing. Also, it is easily learned by most developers since it has no notion of XML unlike SVG [34]. One weakness of Canvas is the fact that objects drawn on a canvas are not part of the page's DOM or any namespace. On the other hand, the reasons why it performs well are because it does not have to store objects for each primitive it generates, and it is easy to implement the Canvas API based on many popular 2D drawing APIs that are found in other programming languages [35].

According to [34], of the three, the Canvas is the most powerful technology when it comes to image manipulation capabilities. It was also expected that Canvas will be used extensively in many future web applications. Furthermore, it was demonstrated that Canvas, SVG and VML are powerful enough to support image manipulation techniques [34].

F. Real Time Collaborative Applications

Real-time distributed groupware allows multiple people to work together at the same time, even when they are in different physical locations. These systems usually support the team's ability to manipulate documents via a shared work space. Each participant's site is kept synchronized with the others' by interchanging appropriate control messages. A system supporting real time collaboration should be able to handle human factors of how people collaborate as well as the expected technical considerations [36].

G. Concurrency Control

Concurrency control is the managing of parallel processes that may interfere with each other. Collaborative systems may suffer concurrency control problems which may result to conflicts. These conflicts may cause the collaborators to become confused because of inconsistencies. The document may even get corrupted if events are handled out of order. Concurrency control problems arise when the software, data and interface are distributed over many computers. Time delays are also a factor that can cause conflicting actions [36].

No Computer-Mediated Concurrency Control

Collaborative systems without concurrency control may be acceptable on some situations. This is allowed if inconsistencies don't matter or if users are able to mediate and repair the conflicts with their own actions. This also assumes that people naturally follow social protocols for interactions, such as turn-taking in conversations. Moreover, people usually would not perform actions that would interfere with others when they see that they are working on something. However, without concurrency control, problems may be encountered such as accidental interference if one person does not notice other person's actions [36].

Locking

The locking scheme guarantees that people that share a workspace access objects one at a time. This can be as strict as having a single lock on an entire document wherein only one can edit the document while the others just have to watch. The obvious drawback is that the collaboration is not natural and waiting for locks is not desirable. Also, the interface has to provide information that an object or document is waiting for a lock request to be approved. Locking may be considered equivalent to a turn-taking protocol in shared view systems [36].

Differential Synchronization

A strategy for keeping documents synchronized among distributed team members is the Differential Synchronization method developed by Fraser. The complete description of DS is given in [37] but some of the important ideas of the algorithm are: both client and server keeps their own copy of the document and its shadow; the algorithm depends on a diff algorithm, which will generate the edits that will be sent to the other side; and a patch algorithm which will process the edits received from the other side to update the local copy.

The advantages of implementing DS are the following:

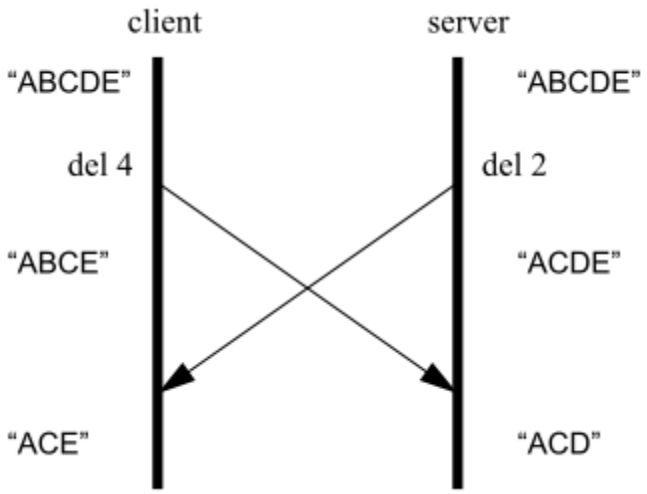
- The algorithm is symmetrical; meaning the code running on both the client and the server are almost the same.
- The application does not need to maintain a history of edits.
- It is asynchronous. Thus, the user input is not blocked while waiting for the server response from the previous edit request.
- It guarantees delivery of edits and is forgiving of unreliable and high-latency networks.
- Convergent. Even if errors are encountered, they do not cause the copies to diverge.
- It can be applied to any kind of document content, with the condition that diff and patch algorithms for those kinds of content exist.
- It is also highly scalable.

Differential Synchronization was implemented in several systems such as the MobWrite. MobWrite is an application that accommodates multiple users who are working on the same text. Tests on MobWrite showed that its technical scalability is impressive [37].

Although Differential Synchronization looks promising and implementing it is practically easy, the difficult part is on finding diff and patch algorithms compatible with the kind of content to be shared by participants. The diff and patch libraries for plain text are readily available. However, for structured data such as XML, not only that diff and patch algorithm libraries are hard to find, they are also much more complicated compared to diff and patch for simple text. Using diff and patch algorithms for plain text on structured content can be done, but there's no guarantee that one will always end up with a valid content.

Operational Transformation

Operational Transformation is a technique used to correct conflicts encountered among groupware systems. An example of a conflict in a collaborative text editing application is borrowed from [38]. As shown in the following figure, the client and server starts with the same text string. On the client side, a delete operation was done on the fourth character while on the server side, an operation was also done to delete the second character. And then both sent their operations and received the one from the other side. On the client side, applying the received operation without alterations will not be a problem. However, the problem occurs on the server, because the client's intention was to delete the character "D" and not the character "E" which will be deleted instead if the received operation is directly applied. To resolve conflicts like this, Operational Transformation works by fixing the operations to be applied by considering previously completed operations so not only that intention is preserved, but the participants end up with the same exact document as well.



Operational Transformation has been embraced by various groupware such as GROVE[40], Jupiter Collaboration System[38] and Google Docs[39].

a. GROVE

The concept of OT was initiated by Ellis and Gibbs in their work on the group editor GROVE. GROVE uses an algorithm called Distributed Operational Transformation Algorithm (dOPT): When an operation is generated at a site, it is right away executed locally and is given a priority. The operation and its priority are transmitted to all the other sites. When an operation is received at site i, it will figure out if it's a "future operation" (the sending site executed operations which have not been executed by site i). If so, the operation is set aside in a queue until it is ready to be executed. If not, the operation is executed immediately. If the operation was found out to be a "past operation" (site i has executed operations which have not yet been executed at the sending site) then the priority is checked and the operation will be transformed as necessarily before it can be executed.

The Transformation Matrix is the key to resolving conflicts in dOPT. The Transformation Matrix basically takes in the two operations involved in the conflict and their priorities and outputs the

transformed operation to be applied. The Transformation Matrix is so called because it assigns a function to every combination of operations to give out the transformed operation. Thus, if there are m kinds of operations defined in the system, the Transformation Matrix is an m x m matrix.

The dOPT algorithm and the transformation matrix are explained in more detail in [40].

b. Jupiter Collaboration System

Jupiter is a multi-user virtual environment meant to facilitate collaboration among distributed group members. As in GROVE, the Jupiter system relies on Operational Transformation technique to fix conflicting messages which were caused by concurrent activities among users.

Users of Jupiter can see their changes displayed immediately on their windows even without the server confirming the changes (i.e., it is optimistic). The system makes use of an algorithm that is based on dOPT. The main difference though, is that dOPT is n-way-- the sites are interconnected to one another, while the Jupiter algorithm is 2-way-- the clients are not directly connected to each other. Rather, the system is centralized so that every client needs to synchronize its copy just with that of the server. Also, if dOPT has Transformation Matrix to fix conflicting operations, the Jupiter algorithm has its xform function that determines the manner that the operations are to be transformed. The full description of Jupiter system is found in [38].

c. Google Docs

Google Docs is a web-based document-editing application from Google. It supports collaborative editing on a single document. It relies on Operational Transformation to resolve conflicts and merge user changes. It also applies its own collaboration protocol to know when to transform operations.

The Google Docs collaboration protocol has many advantages [39]:

- Collaboration is fast and optimistic. Therefore, the editor can apply user changes in the local copy without waiting for the server to acknowledge them.
- Collaboration is accurate.
- It is also efficient.
- Collaboration is scalable, meaning the complexity of processing changes on the server does not increase as more editors are added.
- The workload is divided between all parties involved.

H. Awareness of Other Users

Awareness is also an issue on collaborative systems. Its goal is to provide the user the feeling of presence of others, as if they were together in a face-to-face meeting. Of course, one way to provide awareness is by displaying the names of the participants in a group session. One can improve this by also displaying small pictures of each. When a participant enters or leaves a session, his picture appears or disappears accordingly. Cartoon faces can also be added to express certain moods of the users. Other ways of providing awareness are assigning a unique color to each user and the respective objects that they select, and by indicating the areas of the space which have been modified along with some more information such as the type of change that happened and the identity of who made the change [41].

IV. Design and Implementation

Overview

A Web-based **Co**llaborative **R**eal-time **Di**agram **E**ditor (CORDIE) is presented (Fig. 67). The system operates by the following steps:

1. When a user opens a diagram, it will request for the diagram's current state.
2. The server will then send it so that the user can have its own copy. If it's the first time that a user is requesting for the diagram (i.e. no other user has opened it and is currently editing the diagram) the server will retrieve the diagram from its corresponding file.
3. The user will make modifications on the diagram. The changes done by the user are displayed immediately in his or her editor even if the server has not yet received or processed them.
4. Each modification done is sent as a sequence of one or more operations that are then sent to the server.
5. The server applies the operations sent by every client on the server copy, correcting conflicting operations if needed.
6. The server, in addition, keeps an array of updates to be sent for every connected client. The operations executed by the server are added to the updates array of every client except the one who originally generated the operation.
7. Whenever a client requests for updates, the server will send all of the contents of the updates array for that client.
8. The client will apply every operation it receives to its local copy, also correcting conflicting operations if necessary.

Steps 3 to 8 are repeated indefinitely until all users are done editing.

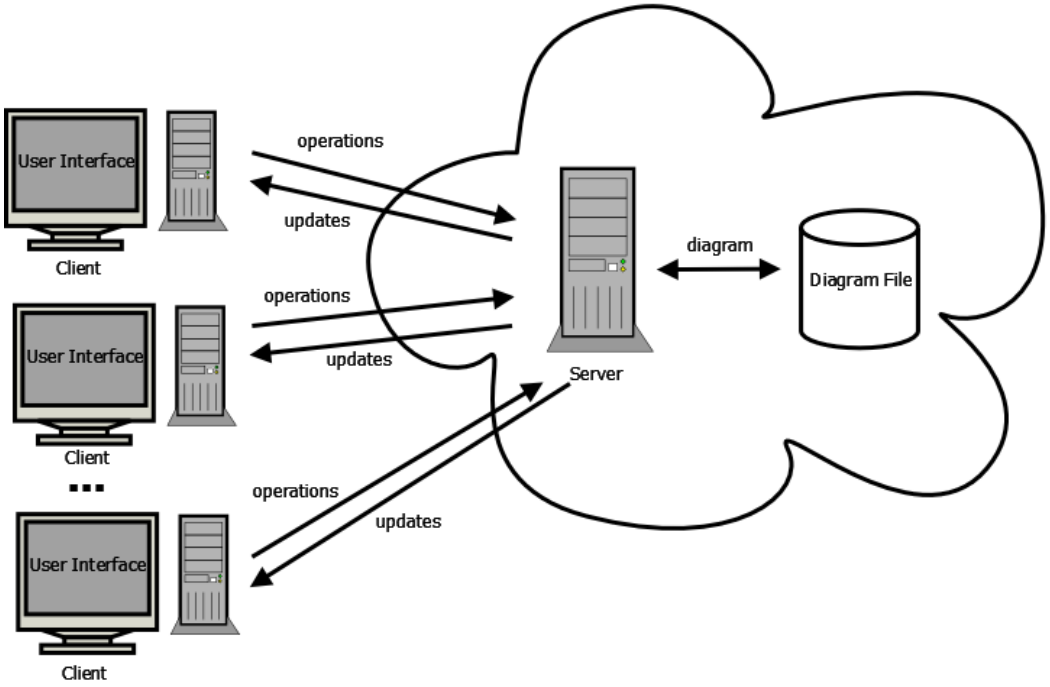


Figure 67. Overview of the System

Context Diagram

The system's context diagram is as displayed in Fig. 68. It shows that the user has to input some user information (username and password), the title of the diagram he/she wants to edit, the modifications and usernames of collaborators he/she wants to add or remove. In return, the system will output the diagram and image file. In addition, diagram files are saved in a repository and a database is kept to save and retrieve user and diagram information.

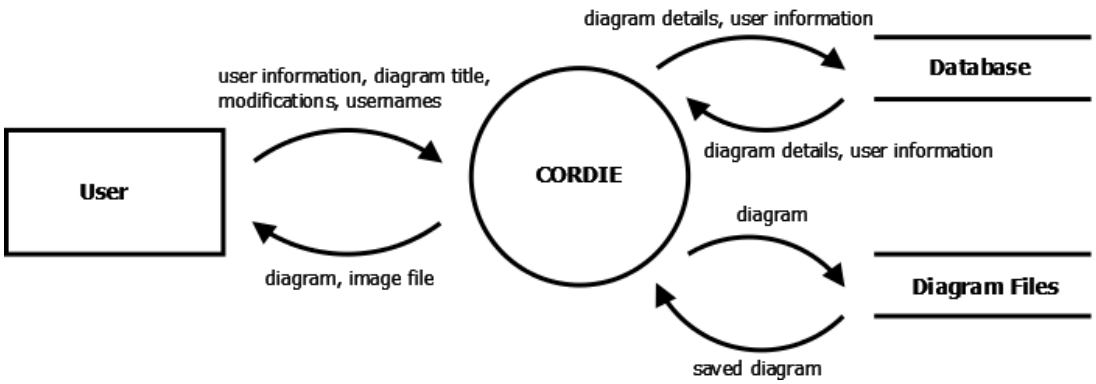


Figure 68. Context Diagram

Entity / Relationship Diagram

The Entity Relationship diagrams for the system are shown in figure 69, figure 70 and figure 71. There are only two main entities: User and Diagram. The user can create zero to many diagrams. On the other hand, a single diagram can be edited by several users.

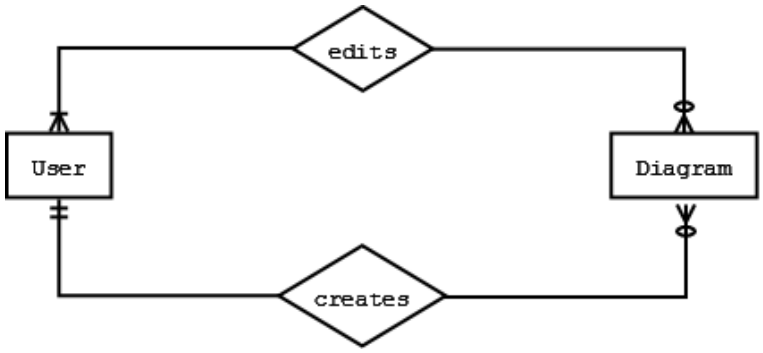


Figure 69. Relationships Between Entities

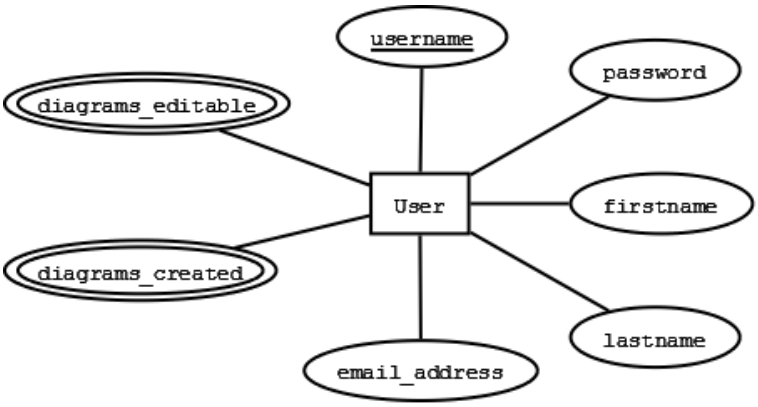


Figure 70. Attributes of Entity User

The entity User has the following attributes: username, password, firstname, lastname, email_address, diagrams_created, and diagrams_editable. The attribute diagrams_created indicates which diagrams were created by the user. The attribute diagrams_editable specifies which diagrams are allowed to be edited by the user. The rest of the attributes are described in the data dictionary.

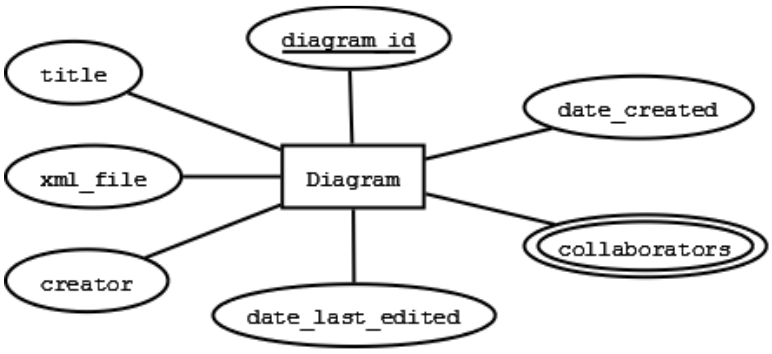


Figure 71. Attributes of Entity Diagram

The entity Diagram has the following attributes: diagram_id, date_created, collaborators, date_last_edited, creator, xml_file and title. Its attribute collaborators indicate the users who are allowed to edit the diagram aside from its creator. Other attributes of the entity Diagram are described in the data dictionary.

Data Dictionary

Table user

Field Name	Data Type	Description
username	String (required)	Unique username Primary key
password	String (required)	User's password
firstname	String	User's first name
lastname	String	User's last name
email_address	String	User's email address

Table collaborators

Field Name	Data Type	Description
username	String (required)	Username of the user allowed to edit the diagram Foreign key from USER Composite Primary Key (1)
diagram_id	Numeric (required)	Diagram ID Foreign key from DIAGRAM Composite Primary Key (2)

Table diagram

Field Name	Data Type	Description
diagram_id	Numeric Auto increment	Unique diagram ID Primary key
xml_file	String (required)	Unique path to diagram's XML file
creator	String (required)	Username of the creator of the diagram
title	String	Title of the diagram
date_last_edited	Date	Date the diagram was last edited
date_created	Date (required)	Date the diagram was created

Diagram Format

A diagram is represented as a tree and is saved in XML format. As shown in Fig. 72, the root of the tree is a `diagram` node which has two children—the `collaborators` node and `diagramobjects` node. The children of the `collaborators` node are `collaborator` nodes, each of which contains the information about a collaborator. On the other hand, the children of the `diagramobjects` node are

the objects that are contained in the diagram. The schema and a sample document are shown in Fig. 73 and Fig. 74, respectively.

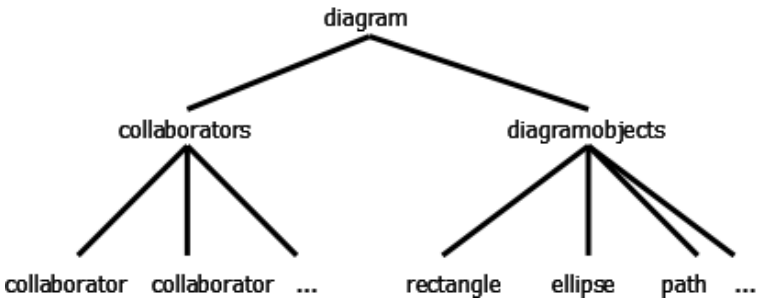


Figure 72. Document Tree

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:complexType name="lineType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="linestyle" type="xs:string" use="required"/>
   <xs:attribute name="arrowstyle1" type="xs:string" use="required"/>
   <xs:attribute name="arrowstyle2" type="xs:string" use="required"/>
   <xs:attribute name="x1" type="xs:integer" use="required"/>
   <xs:attribute name="y1" type="xs:integer" use="required"/>
   <xs:attribute name="x2" type="xs:integer" use="required"/>
   <xs:attribute name="y2" type="xs:integer" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="otherobjectsType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="width" type="xs:integer" use="required"/>
   <xs:attribute name="height" type="xs:integer" use="required"/>
   <xs:attribute name="fillcolor" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="textType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="value" type="xs:string" use="required"/>
   <xs:attribute name="fontstyle" type="xs:string" use="required"/>
   <xs:attribute name="fontweight" type="xs:string" use="required"/>
   <xs:attribute name="fontsize" type="xs:string" use="required"/>
   <xs:attribute name="fontfamily" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="alignment" type="xs:string" use="required"/>
   <xs:attribute name="fillcolor" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="pathType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="points" type="xs:string" use="required"/>
```

```
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="polygonType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="points" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:integer" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="curveType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="linestyle" type="xs:string" use="required"/>
   <xs:attribute name="arrowstyle1" type="xs:string" use="required"/>
   <xs:attribute name="arrowstyle2" type="xs:string" use="required"/>
   <xs:attribute name="x1" type="xs:integer" use="required"/>
   <xs:attribute name="y1" type="xs:integer" use="required"/>
   <xs:attribute name="x2" type="xs:integer" use="required"/>
   <xs:attribute name="y2" type="xs:integer" use="required"/>
   <xs:attribute name="ctrlx" type="xs:integer" use="required"/>
   <xs:attribute name="ctrly" type="xs:integer" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="classType">
   <xs:sequence>
      <xs:element name="classattributes">
         <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="classattribute" type="xs:string"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="operations">
         <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="operation" type="xs:string"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="templates">
         <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="template" type="xs:string"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
```

```
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="width" type="xs:integer" use="required"/>
   <xs:attribute name="height" type="xs:integer" use="required"/>
   <xs:attribute name="fillcolor" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
   <xs:attribute name="abstractclass" type="xs:boolean" use="required"/>
   <xs:attribute name="activeclass" type="xs:boolean" use="required"/>
   <xs:attribute name="interface" type="xs:boolean" use="required"/>
   <xs:attribute name="name" type="xs:string" use="required"/>
   <xs:attribute name="qualifiedassociation" type="xs:boolean"
use="required"/>
   <xs:attribute name="qualifier" type="xs:string" use="required"/>
   <xs:attribute name="templateclass" type="xs:boolean" use="required"/>
   <xs:attribute name="textcolor" type="xs:string" use="required"/>
   <xs:attribute name="fontsize" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="umlobjectType">
   <xs:sequence>
      <xs:element name="umlobjectattributes">
         <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="umlobjectattribute" type="xs:string"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="width" type="xs:integer" use="required"/>
   <xs:attribute name="height" type="xs:integer" use="required"/>
   <xs:attribute name="fillcolor" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
   <xs:attribute name="classname" type="xs:string" use="required"/>
   <xs:attribute name="objectname" type="xs:string" use="required"/>
   <xs:attribute name="textcolor" type="xs:string" use="required"/>
   <xs:attribute name="fontsize" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="labeledobjectsType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="width" type="xs:integer" use="required"/>
```

```
   <xs:attribute name="height" type="xs:integer" use="required"/>
   <xs:attribute name="fillcolor" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
   <xs:attribute name="fontstyle" type="xs:string" use="required"/>
   <xs:attribute name="fontweight" type="xs:string" use="required"/>
   <xs:attribute name="fontsize" type="xs:string" use="required"/>
   <xs:attribute name="fontfamily" type="xs:string" use="required"/>
   <xs:attribute name="textcolor" type="xs:string" use="required"/>
   <xs:attribute name="label" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="classroleType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="width" type="xs:integer" use="required"/>
   <xs:attribute name="height" type="xs:integer" use="required"/>
   <xs:attribute name="fillcolor" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
   <xs:attribute name="classname" type="xs:string" use="required"/>
   <xs:attribute name="objectname" type="xs:string" use="required"/>
   <xs:attribute name="textcolor" type="xs:string" use="required"/>
   <xs:attribute name="fontsize" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="nodeType">
   <xs:sequence>
      <xs:element name="artifacts">
         <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="artifact" type="xs:string"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="width" type="xs:integer" use="required"/>
   <xs:attribute name="height" type="xs:integer" use="required"/>
   <xs:attribute name="fillcolor" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
   <xs:attribute name="nodename" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="stateboxType">
   <xs:sequence>
      <xs:element name="internalactivities">
```

```
         <xs:complexType>
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
               <xs:element name="internalactivity" type="xs:string"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:sequence>
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="width" type="xs:integer" use="required"/>
   <xs:attribute name="height" type="xs:integer" use="required"/>
   <xs:attribute name="fillcolor" type="xs:string" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
   <xs:attribute name="statename" type="xs:string" use="required"/>
   <xs:attribute name="showinternalactivities" type="xs:boolean"
use="required"/>
</xs:complexType>

<xs:complexType name="expansionregionType">
   <xs:attribute name="linewidth" type="xs:integer" use="required"/>
   <xs:attribute name="linecap" type="xs:string" use="required"/>
   <xs:attribute name="linejoin" type="xs:string" use="required"/>
   <xs:attribute name="strokestyle" type="xs:string" use="required"/>
   <xs:attribute name="x" type="xs:integer" use="required"/>
   <xs:attribute name="y" type="xs:integer" use="required"/>
   <xs:attribute name="width" type="xs:integer" use="required"/>
   <xs:attribute name="height" type="xs:integer" use="required"/>
   <xs:attribute name="rotate" type="xs:decimal" use="required"/>
   <xs:attribute name="z-order" type="xs:integer" use="required"/>
   <xs:attribute name="listboxpin1x" type="xs:integer" use="required"/>
   <xs:attribute name="listboxpin1y" type="xs:integer" use="required"/>
   <xs:attribute name="listboxpin2x" type="xs:integer" use="required"/>
   <xs:attribute name="listboxpin2y" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="collaboratorType">
   <xs:attribute name="name" type="xs:string" use="required"/>
</xs:complexType>

<xs:element name="diagram">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="collaborators">
            <xs:complexType>
               <xs:sequence minOccurs="1" maxOccurs="unbounded">
                  <xs:element name="collaborator" type="collaboratorType"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="diagramobjects">
            <xs:complexType>
               <xs:all minOccurs="0" maxOccurs="unbounded">
```

```
                    <xs:element name="line" type="lineType"/>
                    <xs:element name="rectangle" type="otherobjectsType"/>
                    <xs:element name="ellipse" type="otherobjectsType"/>
                    <xs:element name="text" type="textType"/>
                    <xs:element name="path" type="pathType"/>
                    <xs:element name="polygon" type="polygonType"/>
                    <xs:element name="curve" type="curveType"/>
                    <xs:element name="class" type="classType"/>
                    <xs:element name="umlobject" type="umlobjectType"/>
                    <xs:element name="note" type="labeledobjectsType"/>
                    <xs:element name="frame" type="labeledobjectsType"/>
                    <xs:element name="package" type="labeledobjectsType"/>
                    <xs:element name="component" type="labeledobjectsType"/>
                    <xs:element name="artifact" type="labeledobjectsType"/>
                    <xs:element name="usecase" type="labeledobjectsType"/>
                    <xs:element name="actor" type="labeledobjectsType"/>
                    <xs:element name="superstate" type="labeledobjectsType"/>
                    <xs:element name="action" type="labeledobjectsType"/>
                    <xs:element name="subactivity" type="labeledobjectsType"/>
                    <xs:element name="timesignal" type="labeledobjectsType"/>
                    <xs:element name="acceptsignal" type="labeledobjectsType"/>
                    <xs:element name="sendsignal" type="labeledobjectsType"/>
                    <xs:element name="connector" type="labeledobjectsType"/>
                    <xs:element name="transformation"
type="labeledobjectsType"/>
                    <xs:element name="part" type="labeledobjectsType"/>
                    <xs:element name="interactionoccurrence"
type="labeledobjectsType"/>
                    <xs:element name="classrole" type="classroleType"/>
                    <xs:element name="node" type="nodeType"/>
                    <xs:element name="statebox" type="stateboxType"/>
                    <xs:element name="expansionregion"
type="expansionregionType"/>
                    <xs:element name="arrowhead" type="otherobjectsType"/>
                    <xs:element name="activationbar" type="otherobjectsType"/>
                    <xs:element name="lifeline" type="otherobjectsType"/>
                    <xs:element name="deletion" type="otherobjectsType"/>
                    <xs:element name="initialpseudostate"
type="otherobjectsType"/>
                    <xs:element name="finalstate" type="otherobjectsType"/>
                    <xs:element name="shallowhistorypseudostate"
type="otherobjectsType"/>
                    <xs:element name="deephistorypseudostate"
type="otherobjectsType"/>
                    <xs:element name="initialnode" type="otherobjectsType"/>
                    <xs:element name="blackbar" type="otherobjectsType"/>
                    <xs:element name="diamond" type="otherobjectsType"/>
                    <xs:element name="finalnode" type="otherobjectsType"/>
                    <xs:element name="pin" type="otherobjectsType"/>
                    <xs:element name="flowfinal" type="otherobjectsType"/>
                    <xs:element name="port" type="otherobjectsType"/>
                </xs:all>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="title" type="xs:string"/>
```

```
      <xs:attribute name="creator" type="xs:string"/>
   </xs:complexType>
</xs:element>
```

Figure 73. Schema

```
<?xml version="1.0"?>
<diagram id="1234" title="Diagram1" creator="gemvillarico">
   <collaborators>
      <collaborator username="gemvillarico"/>
      <collaborator username="username1234"/>
   </collaborators>
   <diagramobjects>
      <rectangle linewidth="4" linecap="butt" linejoin="bevel"
       strokestyle="brown" x="0" y="0" width="50" height="30"
       fillcolor="rgba(0,205,0,0.7)" rotate="0" z-order="0"/>

      <ellipse linewidth="5" linecap="round" linejoin="round"
       strokestyle="grey" x="100" y="100" width="200" height="100"
       fillcolor="rgba(2,165,165,0.7)" rotate="0" z-order="1"/>

      <text linewidth="2" linecap="round" linejoin="round" strokestyle="aqua"
       value="Lorem ipsum dolor sit amet"  fontstyle="italic"
       fontweight="bold" fontsize="60px" fontfamily="sans-serif" x="28"
       y="332" alignment="left" fillcolor="darkorange" rotate="0"
       z-order="2"/>

      <line linewidth="3" linecap="square" linejoin="miter"
       strokestyle="blue" linestyle="solid" arrowstyle1="hollow triangle"
       arrowstyle2="none" x1="10" y1="10" x2="100" y2="100" rotate="90"
       z-order="3"/>
   </diagramobjects>
</diagram>
```

Figure 74. Sample Document

Objects

The system supports drawing of 45 types of objects: Straight Line, Rectangle, Ellipse, Text, Path, Polygon, Curved Line, Class, UML Object, Note, Frame, Package, Component, Artifact, Use Case, Actor, Superstate, Action, Subactivity, Time Signal, Accept Signal, Send Signal, Transformation, Part, Interaction Occurrence, Class Role, Node, State Box, Expansion Region, Arrowhead, Activation Bar, Lifeline, Deletion, Initial Pseudostate, Final State, Shallow History Pseudostate, Deep History Pseudostate, Initial Node, Black Bar, Diamond, Final Node, Connector, Pin, Flow Final and Port. Each object is described by

attributes that determine how they will appear on the canvas. The attributes of each object are described in the following tables.

Line

Attribute	**Description**
arrowstyle1	The arrow style for one of the ends of the line.
arrowstyle2	The arrow style for the other end of the line.
linecap	Describes the cap style. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the path. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linestyle	The line style can be *solid*, *dashed* or *dotted*.
linewidth	Specifies the thickness of the line.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
x1	The x location for one of the endpoints.
x2	The x location for the other endpoint.
y1	The y location for one of the endpoints.
y2	The y location for the other endpoint.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Rectangle, Ellipse, Arrowhead, Activation Bar, Lifeline, Deletion, Initial Pseudostate, Final State, Shallow History Pseudostate, Deep History Pseudostate, Initial Node, Black Bar, Diamond, Final Node, Pin, Flow Final and Port

Attribute	**Description**
fillcolor	The fill style of the object.
height	The height of the object.
linecap	Describes the cap style of the border. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the border. It can have only one of the three line joins: *miter*, *round* or *bevel*.
line width	Specifies the thickness of the border.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
width	The width of the object.
x	The x-coordinate of the top-left corner of the object.
y	The y-coordinate of the top-left corner of the object.

z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Text

Attribute	**Description**
alignment	The text alignment. Its value can be *start*, *end*, *left*, *right* or *center*.
fillcolor	The fill style of the text.
fontfamily	The font-family of the text.
fontsize	The font size of the text.
fontstyle	The font style of the text. Its value can be *normal* or *italic*.
fontweight	The font weight of the text. It can either be *normal* or *bold*.
linecap	Describes the cap style. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the line. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linewidth	Specifies the thickness of the line.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
value	The value of the text.
x	The x and y attributes specify the position of the text in the canvas.
y	The x and y attributes specify the position of the text in the canvas.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Path and Polygon

Attribute	**Description**
linecap	Describes the cap style. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the path. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linewidth	Specifies the thickness of the path/polygon.
points	The coordinates of the points in the path/polygon.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Curve

Attribute	Description
arrowstyle1	The arrow style for one of the ends of the line.
arrowstyle2	The arrow style for the other end of the line.
ctrlx	The x coordinate of the control point.
ctrly	The y coordinate of the control point.
linecap	Describes the cap style. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the path. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linestyle	The line style can be *solid*, *dashed* or *dotted*.
linewidth	Specifies the thickness of the line.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
x1	The x location for one of the endpoints.
x2	The x location for the other endpoint.
y1	The y location for one of the endpoints.
y2	The y location for the other endpoint.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Class

Attribute	Description
abstractclass	This attribute is set to true if the class is an abstract class.
activeclass	This attribute is set to true if the class is an active class.
attributes	Contains the attributes of the class.
fillcolor	The fill style of the class.
fontsize	The font size of the text inside the class.
height	The height of the class.
interface	Set to true if the class is an interface.
linecap	Describes the cap style of the border. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the border. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linewidth	Specifies the thickness of the border.
name	Specifies the class name.
operations	Contains the operations of the class.
qualifiedassociation	True if the class contains a qualifier.
qualifier	The qualifier of the class, if there is any.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
templateclass	Set to true if it is a template class.

templates	Contains the type parameters of the class if it is a template class.
textcolor	The color of the text inside the class.
width	The width of the class.
x	The x-coordinate of the top-left corner of the class.
y	The y-coordinate of the top-left corner of the class.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

UML Object

Attribute	**Description**
attributes	Contains the attributes of the object.
classname	The class of the object.
fillcolor	The fill style of the object.
fontsize	The font size of the text inside the object.
height	The height of the object.
linecap	Describes the cap style of the border. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the border. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linewidth	Specifies the thickness of the border.
objectname	Contains the name of the object.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
textcolor	Color of the text inside the object.
width	The width of the object.
x	The x-coordinate of the top-left corner of the object.
y	The y-coordinate of the top-left corner of the object.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Note, Frame, Package, Component, Artifact, Use Case, Actor, Superstate, Action, Subactivity, Time Signal, Accept Signal, Send Signal, Connector, Transformation, Part and Interaction Occurrence

Attribute	**Description**
fillcolor	The fill style of the object.
fontfamily	The font-family of the label accompanying the object.
fontsize	The font size of the label accompanying the object.
fontstyle	The font style of the label accompanying the object. Its value can be *normal* or *italic*.
fontweight	The font weight of the label accompanying the object. It can either be *normal* or *bold*.

height	The height of the object.
label	The label of the object.
linecap	Describes the cap style of the border. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the border. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linewidth	Specifies the thickness of the border.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
textcolor	Color of the textual label.
width	The width of the object.
x	The x-coordinate of the top-left corner of the object.
y	The y-coordinate of the top-left corner of the object.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Class Role

Attribute	**Description**
classname	The class to which the object belongs.
fillcolor	The fill style of the class role.
fontsize	The font size of the text inside the object.
height	The height of the class role.
linecap	Describes the cap style of the border. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the border. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linewidth	Specifies the thickness of the border.
objectname	The name of this object.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
textcolor	Color of the text inside the object.
width	The width of the class role.
x	The x-coordinate of the top-left corner of the class role.
y	The y-coordinate of the top-left corner of the class role.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Node

Attribute	**Description**
artifacts	The artifacts contained in the node.
fillcolor	The fill style of the node.

height	The height of the node.
linecap	Describes the cap style of the border. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the border. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linewidth	Specifies the thickness of the border.
nodename	Name of the node.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
width	The width of the node.
x	The x-coordinate of the top-left corner of the node.
y	The y-coordinate of the top-left corner of the node node.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

State Box

Attribute	**Description**
fillcolor	The fill style of the state.
height	The height of the state.
internalactivities	Contains the internal activites of the state.
linecap	Describes the cap style of the border. The cap style can be *butt*, *round* or *square*.
linejoin	The join style of the border. It can have only one of the three line joins: *miter*, *round* or *bevel*.
linewidth	Specifies the thickness of the border.
rotate	The angle (in degrees) at which the object is rotated.
showinternalactivities	Set to true if internal activities are to be shown.
statename	Name of the state.
strokestyle	This can be the color or the pattern of the stroke.
width	The width of the state box.
x	The x-coordinate of the top-left corner of the state box.
y	The y-coordinate of the top-left corner of the state box.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Expansion Region

Attribute	**Description**
height	The height of the expansion region.
linewidth	Specifies the thickness of the border.
linecap	Describes the cap style of the border. The cap style can be *butt*, *round* or *square*.

linejoin	The join style of the border. It can have only one of the three line joins: *miter*, *round* or *bevel*.
listboxpin1x	The x-coordinate of one of the list box pins.
listboxpin1y	The y-coordinate of one of the list box pins.
listboxpin2x	The x-coordinate of the other list box pin.
listboxpin2y	The y-coordinate of the other list box pin.
rotate	The angle (in degrees) at which the object is rotated.
strokestyle	This can be the color or the pattern of the stroke.
width	The width of the expansion region.
x	The x-coordinate of the top-left corner of the expansion region.
y	The y-coordinate of the top-left corner of the expansion region.
z-order	The order in which the object is drawn. Objects with higher z-order are drawn on the top.

Diagram Editor

The diagram editor is responsible for the following:

- showing the diagram to the user through the canvas
- allowing the user to make changes
- capturing each change as a set of operations
- processing the operations
- and sending and receiving operations.

Canvas

Diagram editing has its graphics requirements because generation of shapes and lines is involved. Moreover, this is done by the users through a browser. Therefore, one of the challenges is to find a way that satisfies the graphics requirements, is supported by most browsers and is easy to implement and control. This is solved by incorporating HTML5 Canvas API and JavaScript in the user interface.

HTML5 Canvas API

The canvas is an element that can be added into a web page. It allocates a rectangular area of a specified size on the page. Once a canvas element is added to the page, one can use JavaScript to manipulate it in any way he or she wants. Graphics, lines and text can be drawn on it. It also provides for advanced animations. The HTML5 Canvas API supports the same two-dimensional drawing operations that are employed by most modern frameworks and operating systems. By now, HTML5 Canvas is supported in most browsers such as Chrome (version 1.0 and greater), Firefox (version 1.5 and greater), Opera (version 9.0 and greater) and Safari (version 1.3 and greater) [35].

Diagram Operations

Every change that can be done using the editor such as dragging and resizing an object can be represented as a sequence of one or more operations. These operations are the following:

a. Insert Object

This operation adds a child on the `diagramobjects` node. It takes the object and the position on which it is to be placed as parameters. That is, if the specified position is 0, the object will become the 0^{th} child of `diagramobjects`. Afterwards, the position of the other children nodes will be adjusted accordingly.

b. Delete Object

This operation takes the position in the child nodes of the `diagramobjects` node of the object to be deleted as its only parameter. Like the insert operation, the position of the other object nodes will be adjusted after applying a delete operation.

c. Edit Object Attribute

This operation sets a value to an attribute of an object. The parameters are the position of the object on the `diagramobjects` children, the attribute to be set and the value it will take.

Merging Changes

Conflicts

To merge the changes of all the collaborators, every operation generated from one site will have to be sent to the server and broadcasted to all other sites. However, simply sending and receiving operations won't be enough because concurrent modifications to the diagram will give rise to conflicts. A conflict can be better understood with the following example.

As shown in Fig. 75, the client and server started with the same diagram. Suppose that the client does a "delete object at position 1" operation. At the same time, on the server side, the diagram was changed because of an "insert shape D at position 0" operation. When the client and the server both sent their operation and received that from the other side, they will not end up with the same diagram.

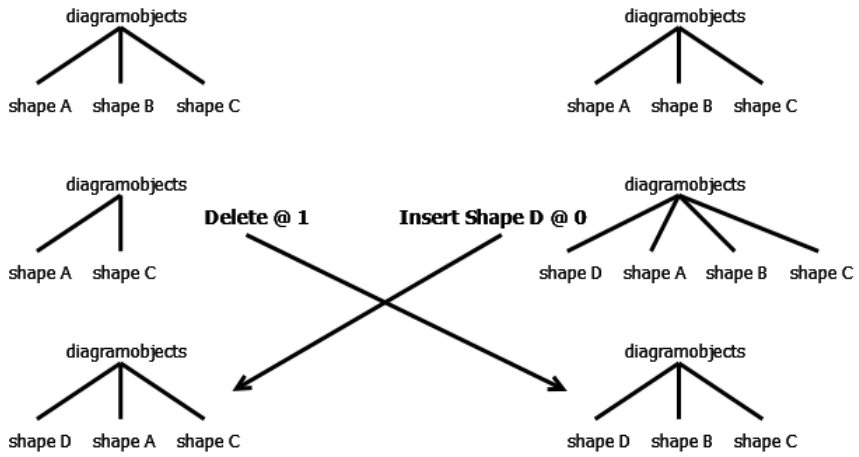


Figure 75. Example of a Conflict

Operational Transformation

The solution adopted by the system is through the use of Operational Transformation. Operational transformation fixes up the conflicting operations by considering previously executed operations so that they make sense relative to the local version of the document. So after applying operational transformation, the solution to the previous example will look like the one shown in Fig. 76. When the server receives the operation from the client, it will transform the operation to “delete object at position 2”. While on the client side, even after performing operational transformation on the one that it received, the operation will still be “insert shape D at position 0”.

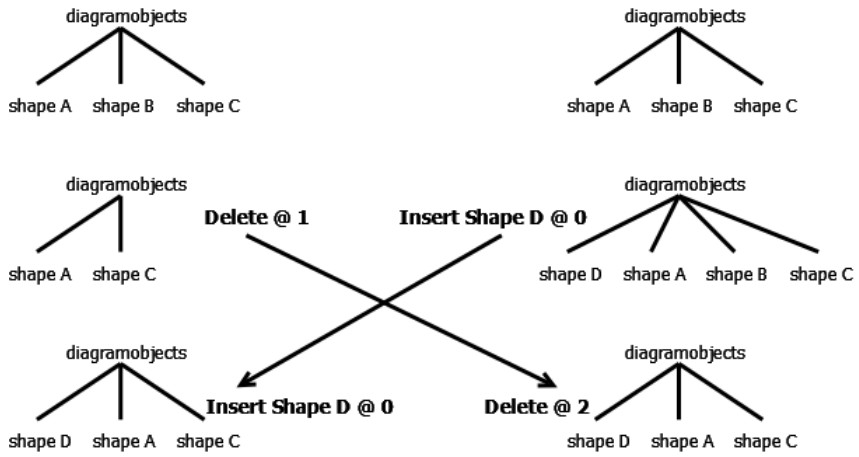


Figure 76. Resolving Conflicts with Operational Transformation

Algorithm

Several algorithms based on Operation Transformation exist. For the CORDIE system, the Jupiter algorithm was chosen. The algorithm plays an essential role in the system because it determines when the operations are to be transformed and which operations are to be transformed against each other. One of the reasons why the Jupiter algorithm was chosen is because it is designed for systems with centralized architecture.

```
int myMsgs = 0;  /* number of messages generated */
int otherMsgs = 0;  /* number of messages received */
queue outgoing = {};

Generate(op) {
      apply op locally;
      send(op, myMsgs, otherMsgs);
      add (op, myMsgs) to outgoing;
      myMsgs = myMsgs + 1;
}

Receive(msg) {
      /* Discard acknowledged messages. */
      for m in (outgoing) {
            if (m.myMsgs < msg.otherMsgs)
            remove m from outgoing
```

```
        }

        /* ASSERT msg.myMsgs == otherMsgs. */
        for i in [1..length(outgoing)] {
              /* Transform message and the ones in the queue. */
              {msg, outgoing[i]} = transform(msg, outgoing[i]);
        }
        apply msg.op locally;
        otherMsgs = otherMsgs + 1;
}
```

Figure 77. Jupiter Algorithm

The algorithm, presented in Fig. 77, is the same for both the client and the server. Essentially, each side will keep track of the following information:

- the number of messages it has sent (myMsgs in the algorithm)
- the number messages received from the other side (otherMsgs)
- and the messages waiting to be acknowledged by the other side (outgoing queue)

When an operation is generated on the client, it is applied instantly on the local copy. It is then sent together with the information about the number of messages the site has generated (myMsgs) and the number of messages it has received (otherMsgs). Afterward, the operation and myMsgs is appended onto the outgoing queue.

On the other hand, when a message is received, the algorithm will remove messages from the outgoing queue that has been acknowledged. Next, the received operation will be transformed against every operation left in the outgoing queue. In the process, the received operation and all of those in the queue may all be altered. Finally, the transformed operation is executed.

Transformation Functions

For the transformation functions, each one was defined for every combination of operations. A transformation function takes up two operations as arguments and returns the transformed version of both operations. In symbols,

Transform(a, b) = {a', b'}

where a and b are the original operations, and a' and b' are the transformed versions. It must have the property that if the client applied a followed by b', and the server applied b followed by a', both the client and server must have the same document state [38].

The transformation functions are the same for the client and the server, except the transformation function for two insert operations and the transformation function for two edit operations. The functions are shown in Fig. 78.

```
Transform(insert[A, p], insert[B, q]) {
      if(p < q)
            return {insert[A, p], insert[B, q+1]};
      else if(p > q)
            return {insert[A, p+1], insert[B, q]};
      else
            return {insert[A, p+1], insert[B, q]};
}

//for server only
Transform(insert[A, p], insert[B, q]) {
      if(p < q)
            return {insert[A, p], insert[B, q+1]};
      else if(p > q)
            return {insert[A, p+1], insert[B, q]};
      else
            return {insert[A, p], insert[B, q]};
}

Transform(insert[A, p], delete[q]) {
      if(p > q)
```

```
            return {insert[A, p-1], delete[q]};
      else
            return {insert[A, p], delete[q+1]};
}

Transform(insert[A, p], edit[q, attr, value]) {
      if(p > q)
            return {insert[A, p], edit[q, attr, value]};
      else
            return {insert[A, p], edit[q+1, attr, value]};
}

Transform(delete[p], insert[A, q]) {
      if(p < q)
            return {delete[p], insert[A, q-1]};
      else
            return {delete[p+1], insert[A, q]};
}

Transform(delete[p], delete[q]) {
      if(p < q)
            return {delete[p], delete[q-1]};
      else if(p > q)
            return {delete[p-1], delete[q]};
      else
            return {'no-op', 'no-op'};
}

Transform(delete[p], edit[q, attr, value]) {
      if(p < q)
            return {delete[p], edit[q-1, attr, value]};
      else if(p > q)
            return {delete[p], edit[q, attr, value]};
      else
            return {delete[p], 'no-op'};
}

Transform(edit[p, attr, value], insert[q]) {
      if(p < q)
            return {edit[p, attr, value], insert[q]};
      else
            return {edit[p+1, attr, value], insert[q]};
}

Transform(edit[p, attr, value], delete[q]) {
      if(p < q)
            return {edit[p, attr, value], delete[q]};
      else if(p > q)
            return {edit[p-1, attr, value], delete[q]};
      else
            return {'no-op', delete[q]};
}

//for client only
Transform(edit[p, attr1, value1], edit[q, attr2, value2]) {
      if(p == q && attr1 == attr2)
            return {'no-op', edit[q, attr2, value2]};
```

```
      else
            return {edit[p, attr1, value1], edit[q, attr2, value2]};
}

//for server only
Transform(edit[p, attr1, value1], edit[q, attr2, value2]) {
      if(p == q && attr1 == attr2)
            return {edit[q, attr2, value2], 'no-op'};
      else
            return {edit[p, attr1, value1], edit[q, attr2, value2]};
}
```

Figure 78. Transformation Functions

Technical Architecture

The system will have a client-server type of architecture. The requirements for the client and the server are as follows:

Client

Hardware:

Internet Connection

Software:

Web browser supporting HTML5 and JavaScript

Server

Software:

Apache Tomcat 6.0.24

MySQL DBMS version 5.0 or higher

V. Results

Figure 1 shows the home page for CORDIE. The home page introduces the CORDIE system to the user to give them an idea about what it is and its features.

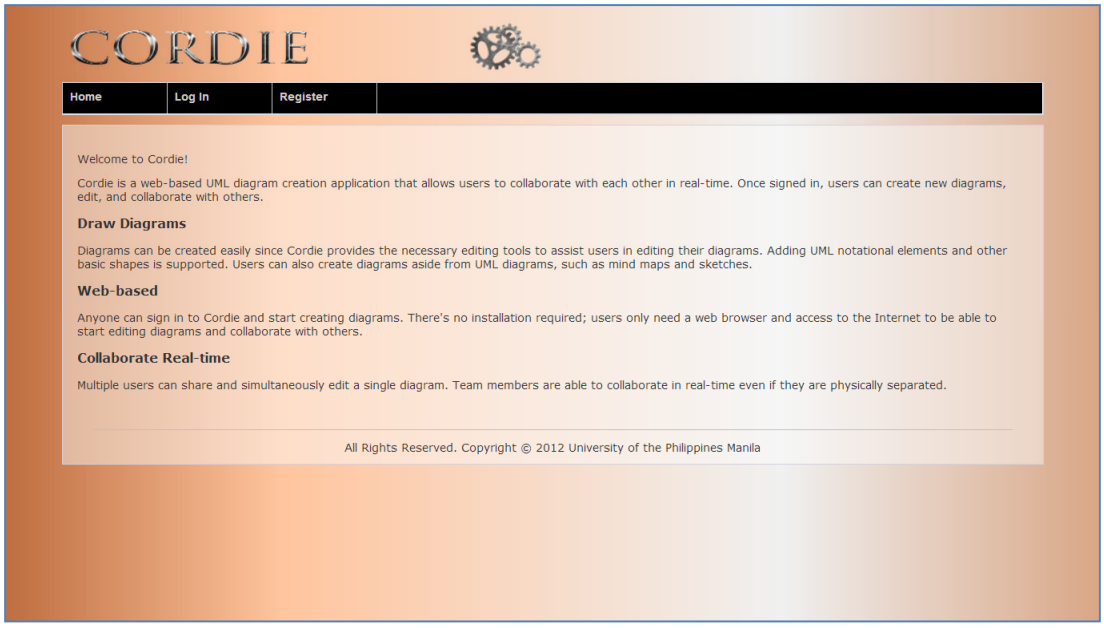


Figure 79. Home Page

Only registered members can make use of the system. Registration can be done through the sign up page (figure 2). The user must fill in at least the required fields (marked by asterisks) and click submit to register. Once the user is registered, he/she can now log in through the log in page (figure 3). The user has to input his/her username and password to be able to access the other parts of the system.

Figure 80. Registration Page

Figure 81. Log In Page

Once the user is signed in, he/she can now view his/her profile in the profile page as shown in figure 4. The user can also edit his/her profile information (figure 5) and password (figure 6).

Figure 82. Profile Page

Figure 83. Edit Profile Page

Figure 84. Edit Password Page

Figure 7 shows the page when the user logs out.

Figure 85. Log Out Page

Users can view the list of diagrams he created and/or is a collaborator on through the diagrams page as shown in figure 8. Here, the user can also perform other functions such as add new, delete, leave, open and copy diagrams.

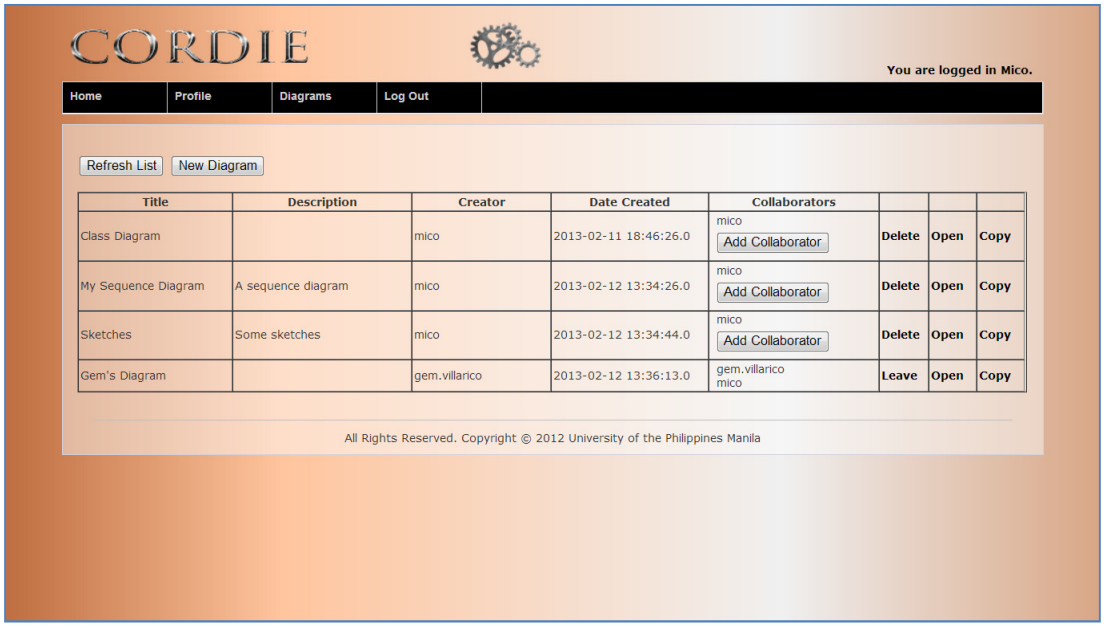


Figure 86. Diagrams Page

Figure 9 shows a user creating a new diagram. After clicking the "New Diagram" button, the user is asked to enter the title of the diagram and a description. After he/she clicks "Add," the diagram just created will be added in his/her list of diagrams (figure 10).

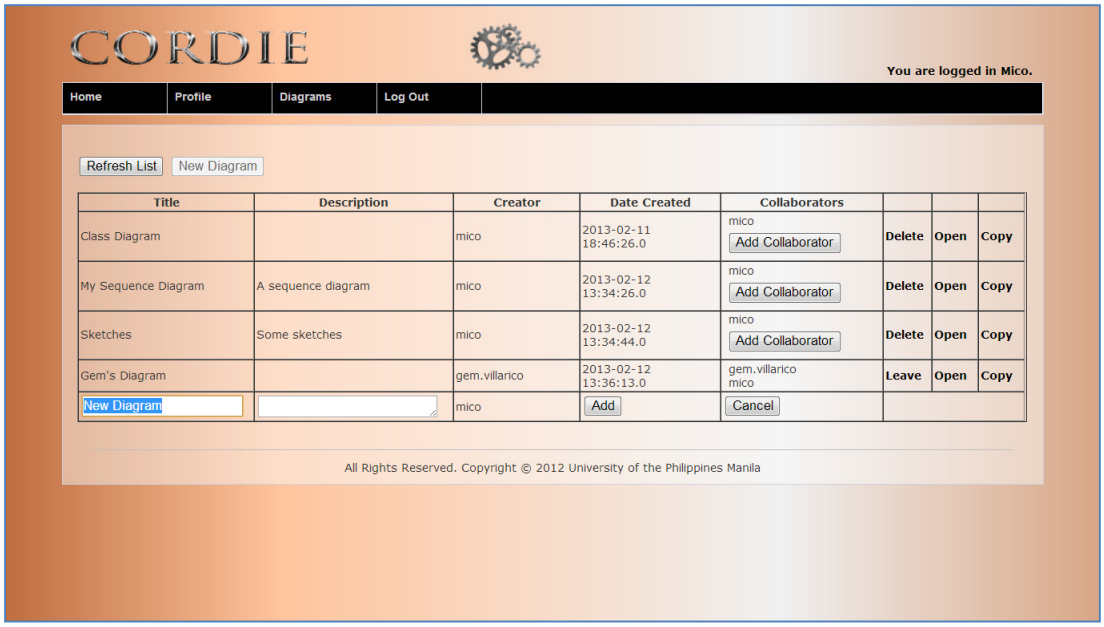


Figure 87. User Creating a New Diagram

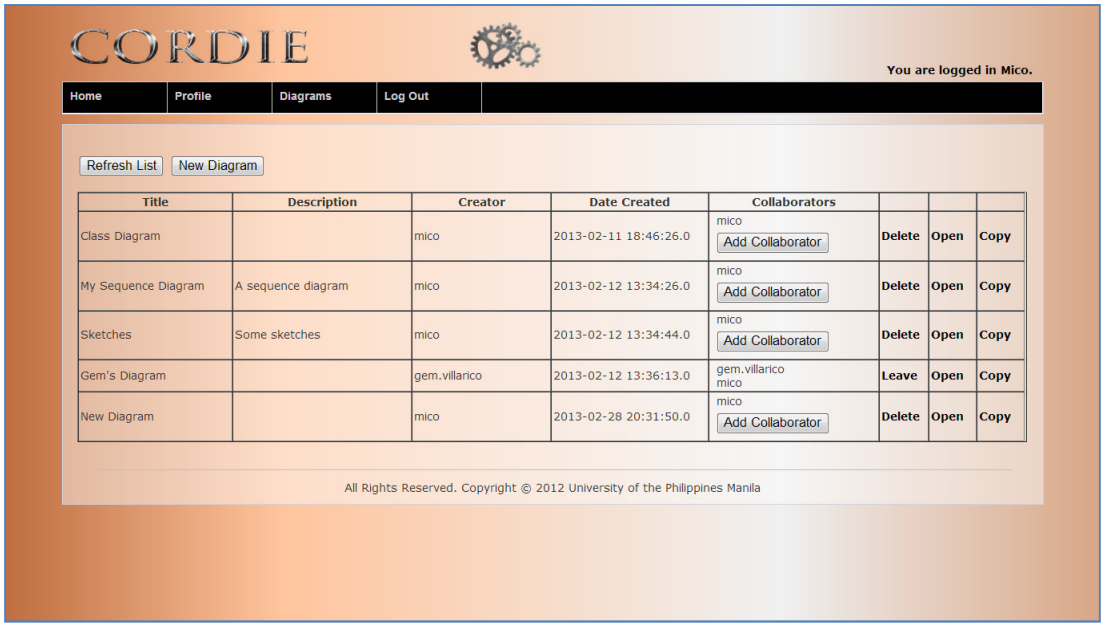


Figure 88. New Diagram Added to User's Diagram List

When the user clicks "Delete," he/she will be asked to confirm the action (figure 11). The deleted diagram will then be removed from the list (figure 12) and can no longer be retrieved.

Figure 89. User Deleting a Diagram

Figure 90. Deleted Diagram Removed from the List

When a user no longer wants to be a collaborator of a diagram of another user, he/she can click "Leave" and confirm the action as demonstrated in figure 13. After confirming the action, he/she will no

longer be a part of the collaborators of that diagram and it will be removed from the list of his/her diagrams.

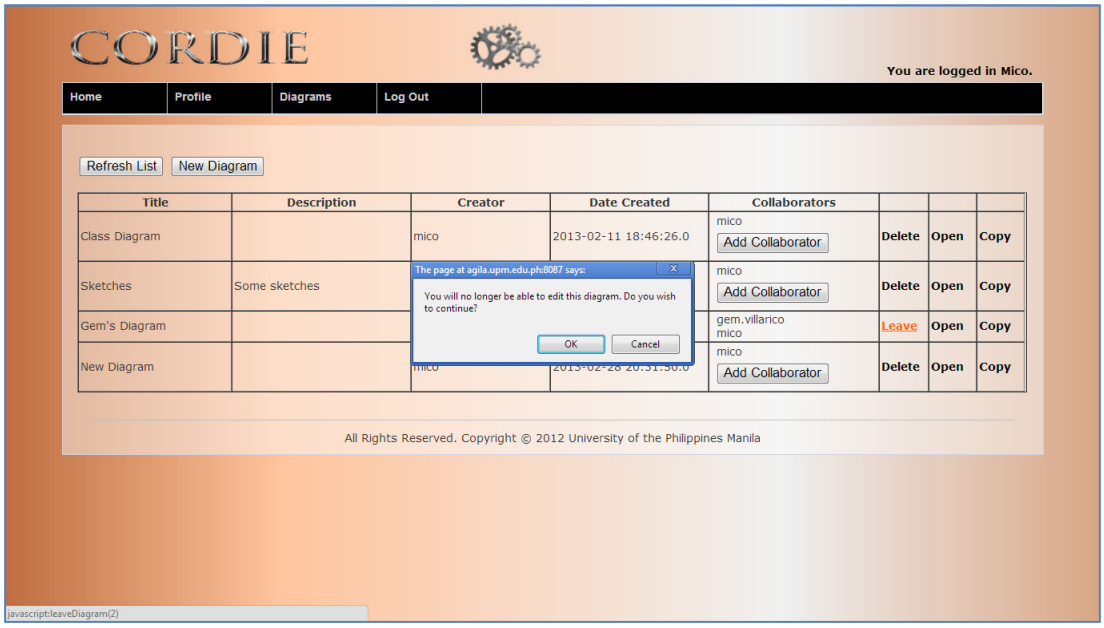


Figure 91. User Leaving as Collaborator of a Diagram

The user can also make a copy of an existing diagram by clicking "Copy". Similar to adding a new diagram, the user should enter the title of the copy and the description, as shown in figure 14. Then, it will be added to his/her list of diagrams (figure 15).

Figure 92. Copying a Diagram

Figure 93. Diagram Copy Added to the List

When a user opens a diagram, he/she will be directed to the editor page as can be seen in figure 16. The editor page provides the necessary tools to edit a diagram. Here, the canvas displays the diagram. A tool box is also available to add shapes, UML notations and other tools for editing.

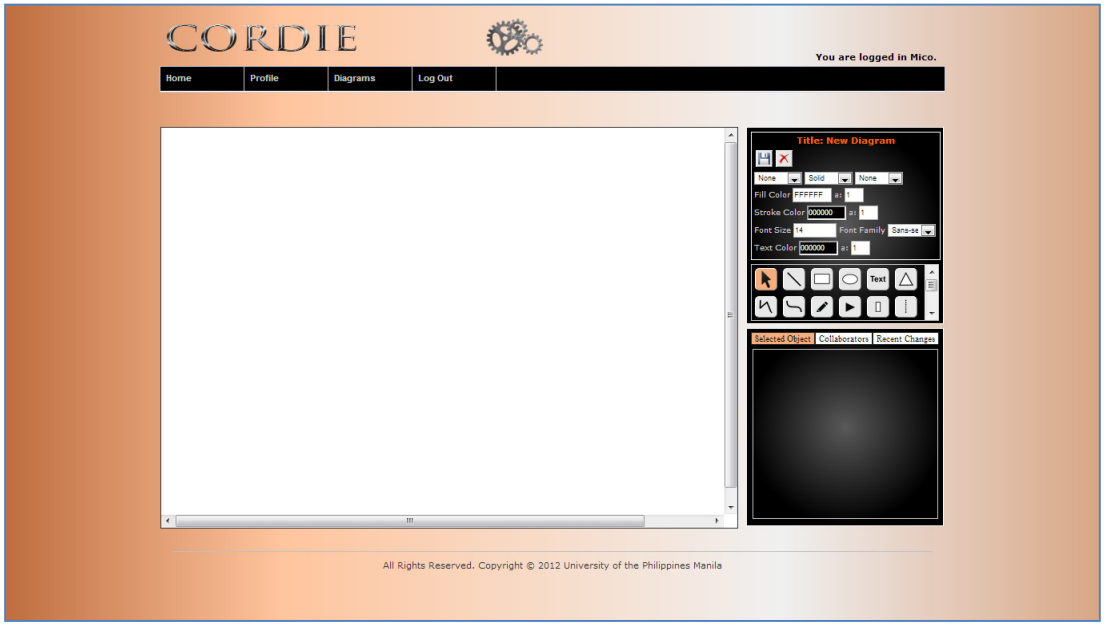


Figure 94. Editor Page

Figures 17 and 18 show an example of using the editor. First, the user clicks the class tool on the tool box (figure 17). Next, the user clicks on the canvas to place the class notation. Figure 18 shows the added class. The object can then be selected, dragged around and/or resized. Figure 19 shows the object being dragged around the canvas.

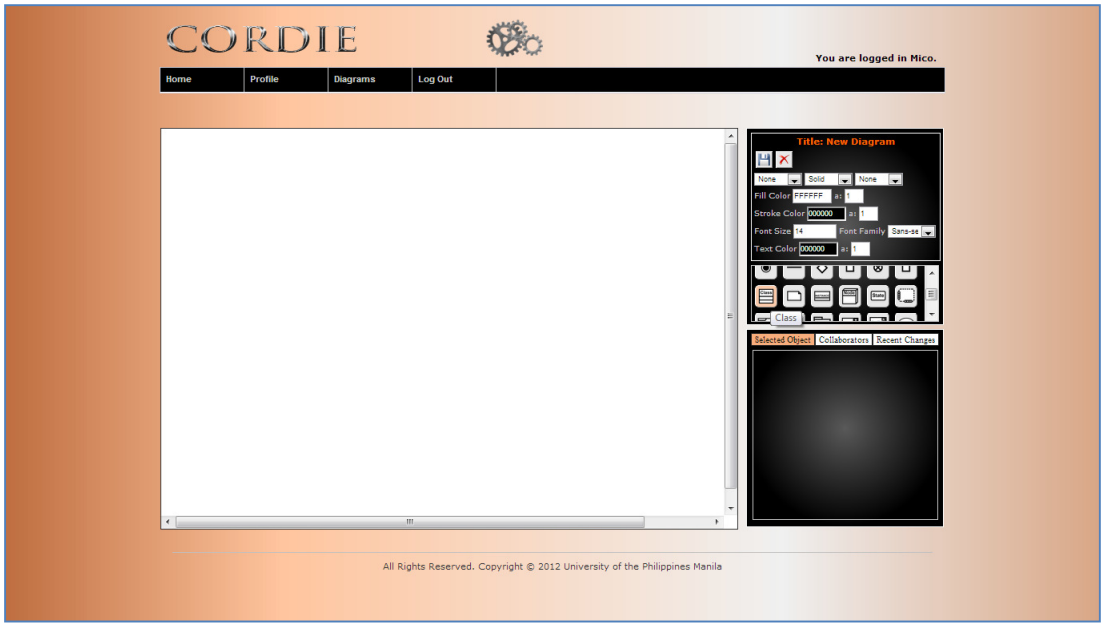


Figure 95. Using the Class Tool of Tool Box

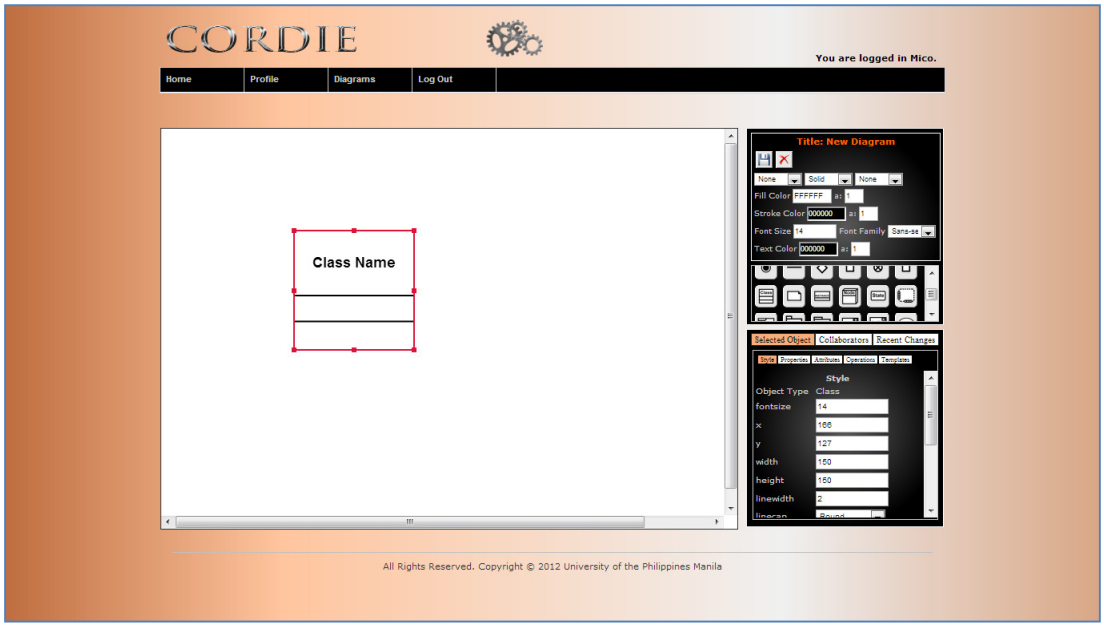


Figure 96. Adding a Class Notation Using the Editor

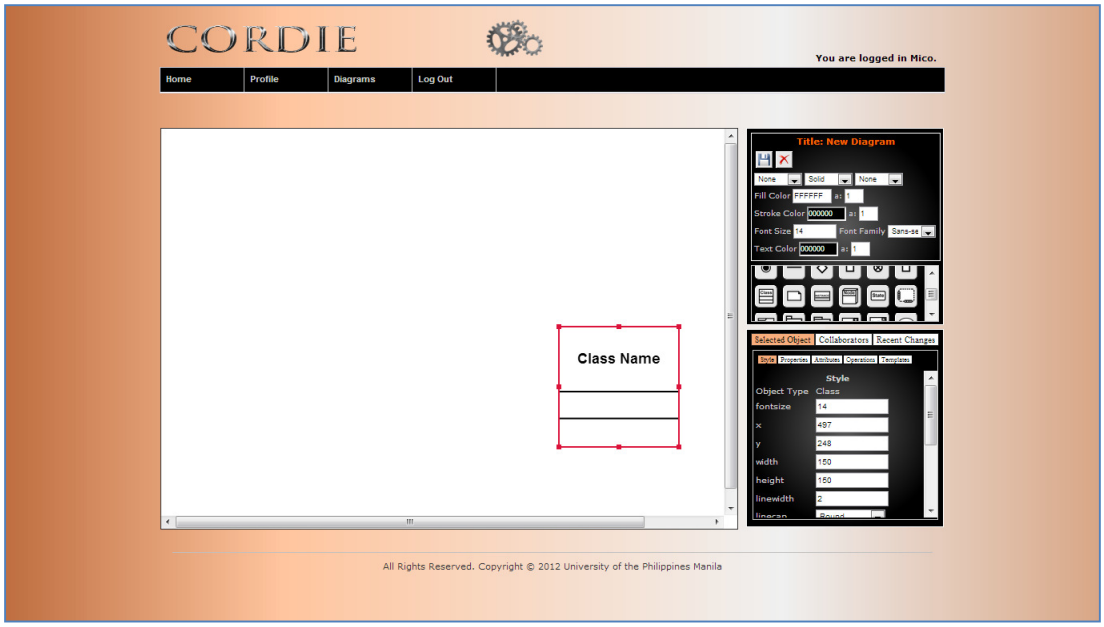


Figure 97. Class Notation Being Dragged Around the Canvas

The user can edit a diagram object through the properties box. The properties box is displayed on the bottom right side of the page. Figure 19 also shows the properties box when a class notation is selected. The user can then edit the style, properties, attributes, operations and templates of the class through the properties box.

The editor page has a collaborators box section where the user can view the collaborators of the diagram (figure 20), and add a collaborator (figure 21). Only the creator can add and remove collaborators. The creator can add a new collaborator by clicking "Add Collaborator" button and entering the username of the user he/she wants to add (figure 21). The added user will then be displayed on the list of collaborators as shown in figure 22 and can then edit the diagram. The creator can also remove a collaborator by clicking "Remove" (figure 23). That user will then be removed from the list of collaborators and will no longer be able to edit the diagram.

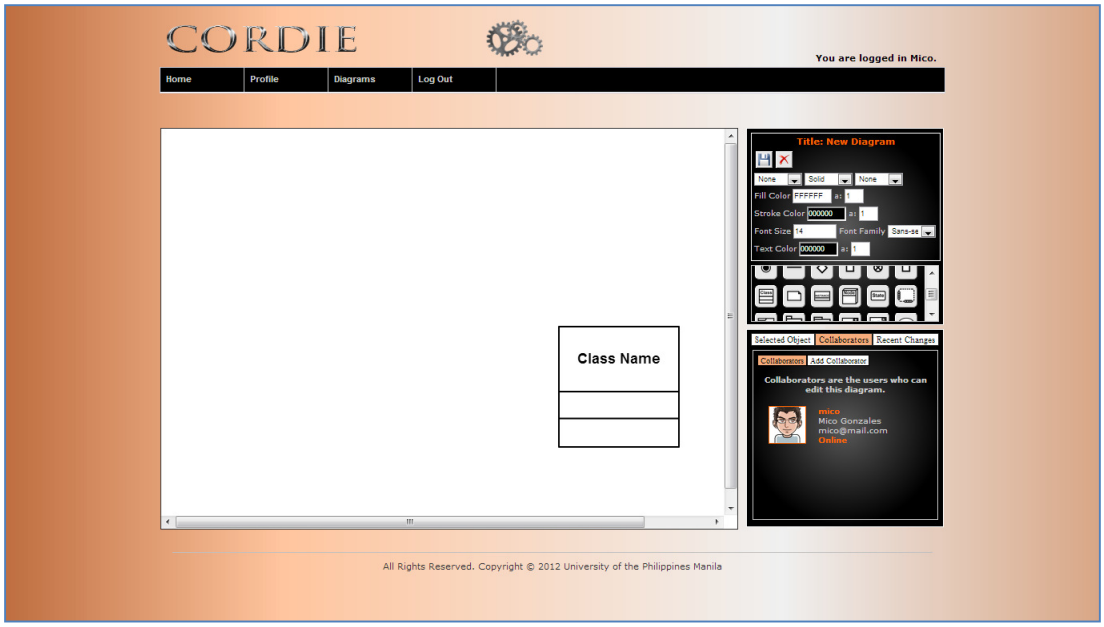


Figure 98. Collaborators of the Diagram

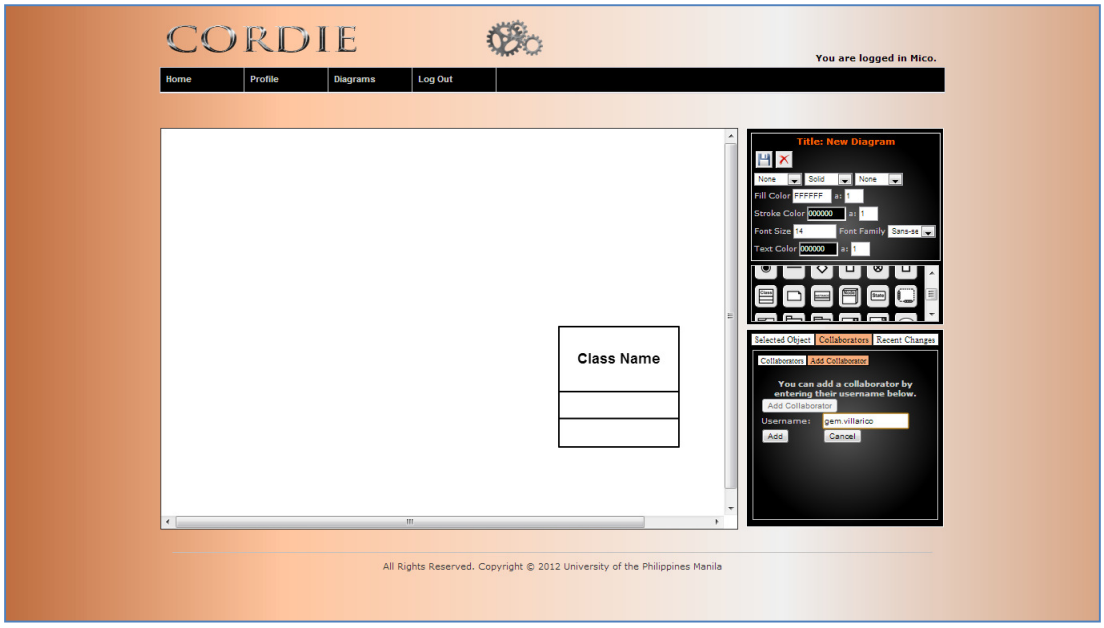


Figure 21. Adding a Collaborator

Figure 22. New Collaborator Added to List of Collaborators

Figure 23. Removing a Collaborator

The user can view the recent changes on the diagram by clicking "Recent Changes" as shown in figure 24. The recent modifications done on the diagram will be displayed to the user.

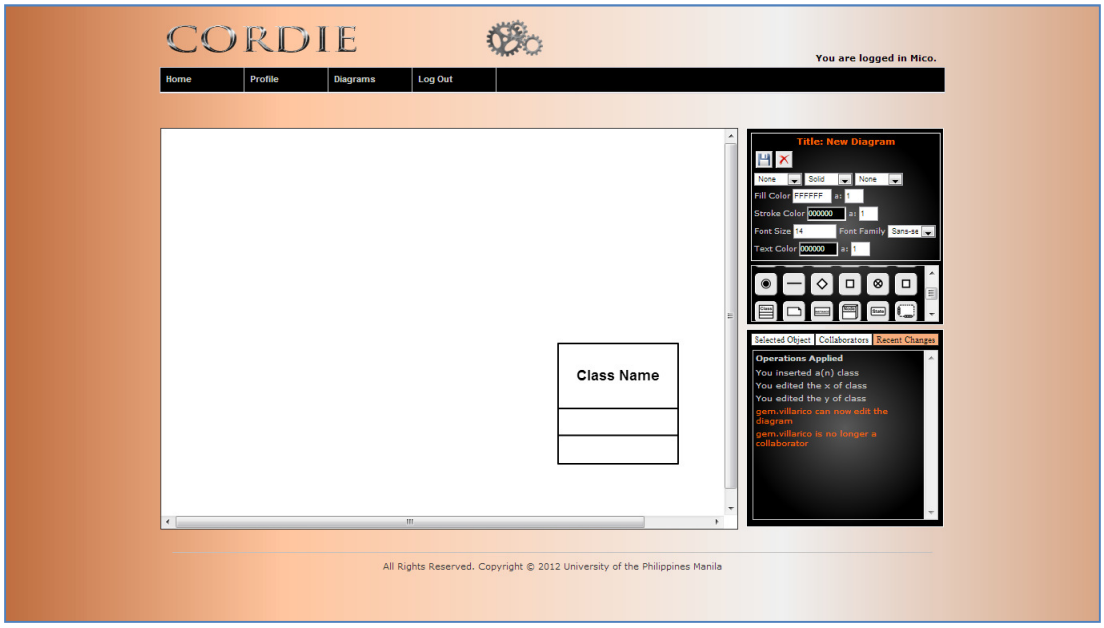


Figure 24. Viewing Recent Changes

The user can save the diagram as an image by clicking the save icon on the top right corner of the page. The diagram will then be downloaded as an image (figure 25).

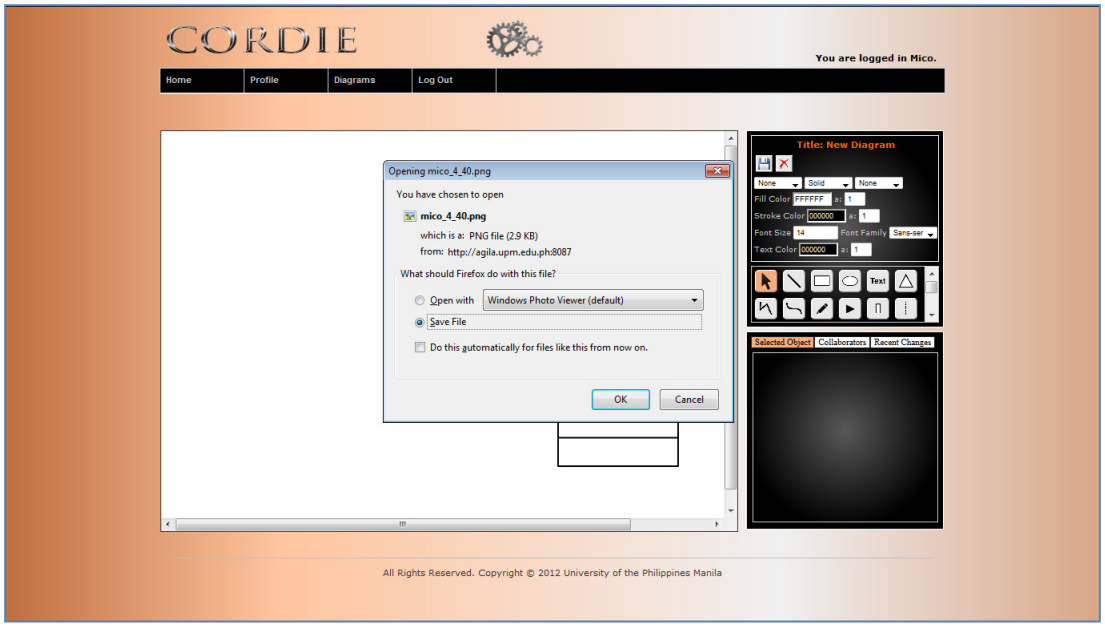


Figure 25. Diagram Being Saved as an Image

Figures 26 to 28 show an example when multiple users are editing the diagram. In figure 26, user A and user B start with the same diagram. Next, user A adds a line to the diagram. Figure 27 shows the different views of users A and B after user A added a line. Figure 28 shows the views of user A and user B after user B's diagram is updated.

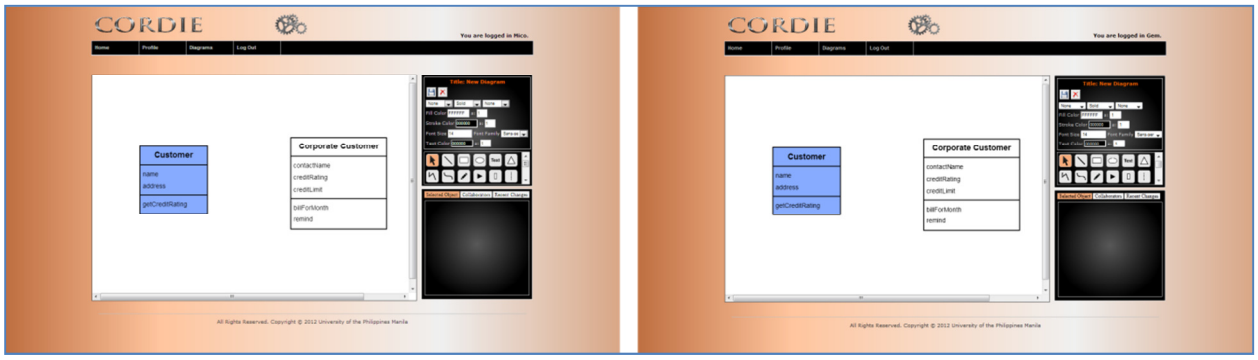


Figure 26. User A and User B Have the Same View of the Diagram at the Start

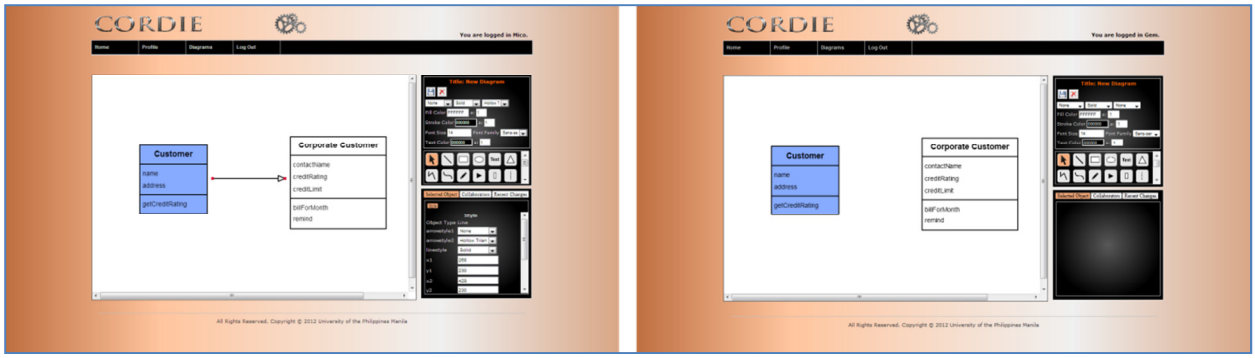


Figure 27. Views of Both Users after User A Added a Line

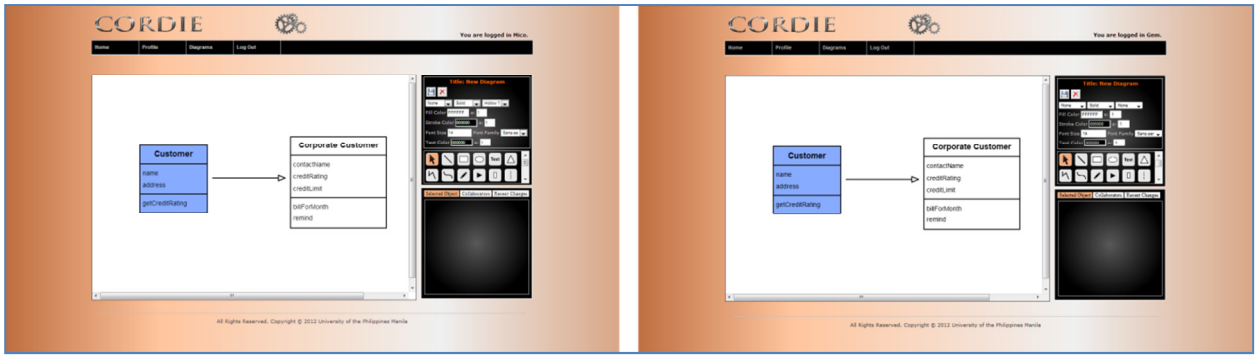


Figure 28. Views of Both User after User B's View is Updated

Figures 29 to 31 show another example of an editing session where users simultaneously edit the same diagram. In figure 29, users A and B start with the same diagram. In figure 30, it can be seen that user A deletes a note notation and edits the class notation, while user B adds a line to the diagram and also edits the class notation. Finally, figure 31 shows the views of user A and user B after their modifications are merged and their canvases are updated.

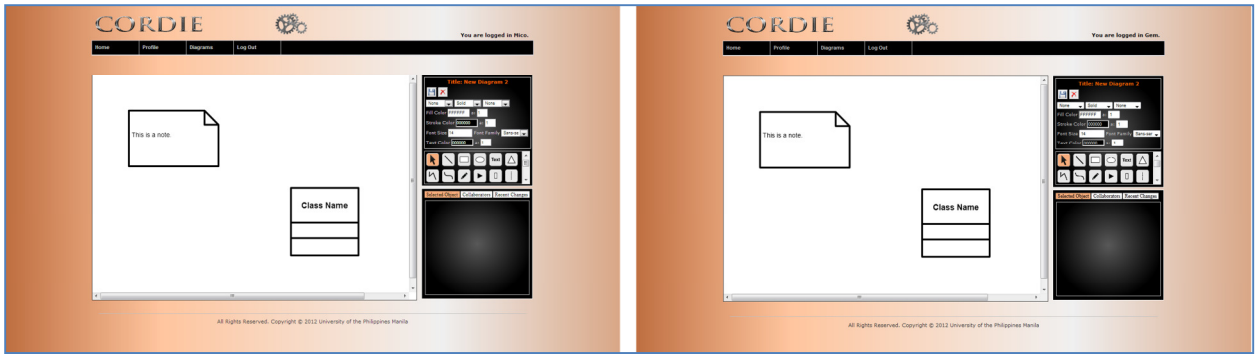


Figure 29. Views of Both Users at the Start

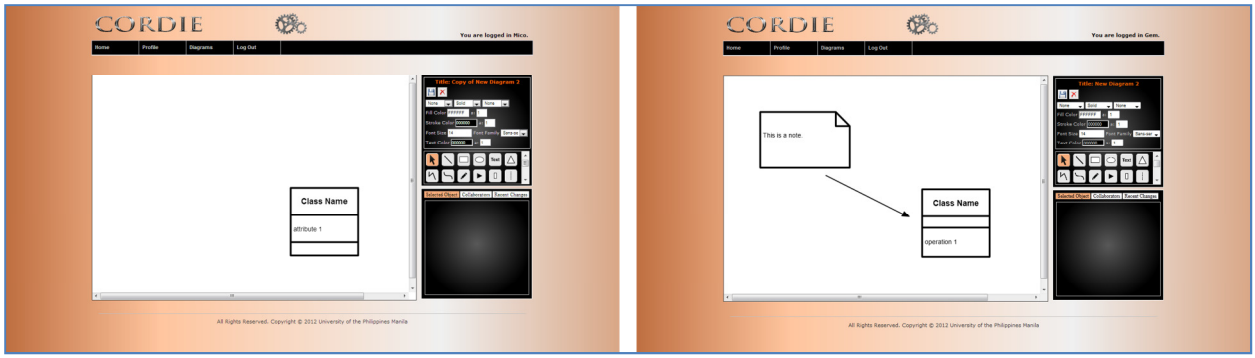


Figure 30. Views of User A and User B after Both Users Made Separate Edits

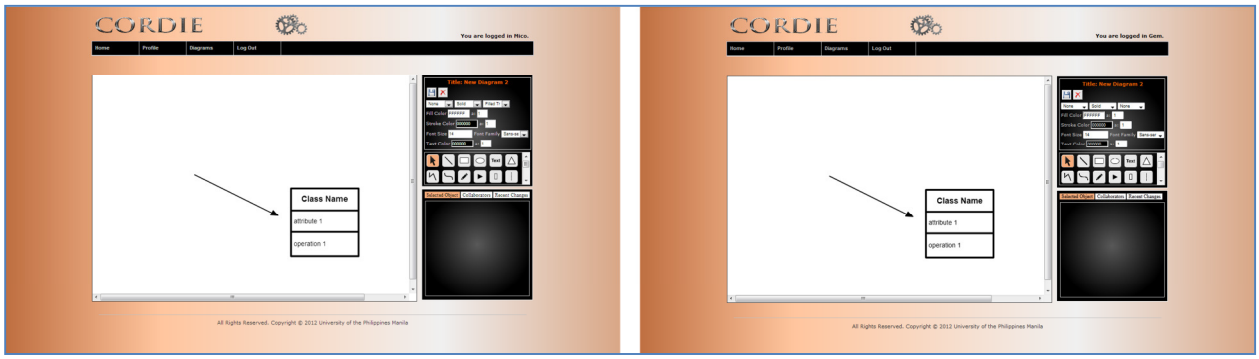


Figure 31. Views of User A and User B after Merging Their Edits

VI. Discussion

Distributed software engineering is becoming the trend in completing software projects. One of the difficulties with this method of software development is that it requires team members, who are most probably in different locations, to participate in the software design process. CORDIE was developed so that users can get involved in the software design even if they are geographically distributed.

Once logged in, a user can start creating new diagrams, delete unneeded diagrams, and open diagrams for editing. CORDIE provides the necessary tools to modify a diagram. The users can add UML notations and basic shapes in designing their diagrams. Its web-based nature makes it easily accessible using only a web browser provided that the user has Internet connection. The CORDIE application also has other features such as saving a diagram as an image and making a copy of a diagram.

Unlike some existing UML editors which supports only single users, CORDIE allows multiple users to edit a single diagram. The creator of a diagram can add and/or remove collaborators through the editor page. If a user is a collaborator of a diagram, he/she can make modifications to the diagram. The editors are also updated in real-time; therefore, users always see the updated version of the diagram. Also, CORDIE can handle concurrent editing. This means that even though collaborators simultaneously edit a diagram, their modifications are merged and they will always end up with exactly the same view of the diagram.

Some collaborative tools implements locking as the concurrency control mechanism. However, locking can greatly affect the fluidity of the collaboration and this is why CORDIE employs Operational Transformation instead. Therefore, users can edit the diagram without waiting for other users to finish

editing. Finally, since CORDIE is web-based, it makes for a great alternative to existing tools which require users to install applications.

VII. Conclusion

The CORDIE application can be used to create UML diagrams. In addition, it allows real-time collaboration of multiple users. It has enough features that qualify it as a tool that supports distributed software engineering, particularly in the collaborative software design process. Users of CORDIE can participate in the software modeling process even if they are in different locations.

Moreover, the design of CORDIE supports the following ideas:

- It is possible to accomplish software modeling process in distributed manner using collaborative tools.
- Documents can be edited using only web browsers.
- Web-based applications can be developed to support real-time collaborative work.

VIII. Recommendation

The system can be further improved by adding support for other types of diagrams. The set of tools can be expanded so that notations for other kinds of diagrams can be drawn using the application. It is also suggested that an instant messaging component is integrated into the application as it can improve the way that users collaborate.

IX. Bibliography

[1] Garrett, C. (2003). Software modeling introduction: What do you need from a modeling tool?

[2] Thum, C., Schwind, M., & Schader, M. (2009). SLIM- a lightweight environment for synchronous collaborative modeling. *MODELS '09 Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*, 137-151.

[3] Chen, Q., Grudy, J., & Hosking, J. (2003). An E-whiteboard application to support early design-stage sketching of UML diagrams. *Proceedings of the 2003 IEEE Conference on Human-Centric Computing*, 219-226.

[4] Whitehead, J. (2007). Collaboration in software engineering: A roadmap. *FOSE '07 2007 Future of Software Engineering*, 214-225.

[5] Damm, C., Hansen, K., & Thomsen, M. (2000). Tool support for cooperative object-oriented design: Gesture based modeling on an electronic whiteboard. *Proceedings of the ACM Conference on Human Factors in Computing Systems, 2(1)*, 518-525.

[6] Cook, C. (2007). *Towards computer-supported collaborative software engineering*. Ph. D. Thesis. University of Canterbury.

[7] Wong, F., Fernandez, G., & McGovern, J. (2007). CONFER: Towards groupware for building consensus in collaborative software engineering. *Proceedings of the eight Australasian conference on User interface, 64*.

[8] Andres, H. (2002). A comparison of face-to-face and virtual software development teams. *Team Performance Management: An International Journal, 8(1/2)*, 39-48.

[9] Graham, T., Stewart, H., Kopaee, A., Ryman, A., & Rasouli, R. (1999). A world-wide-web architecture for collaborative software design. *Proceedings of the Conference on Software Technology and Engineering Practice*, 22-32.

[10] Mangano, N., Dempsey, M., Lopez, N., & Van der Hoek, A. (2011). A demonstration of a distributed software design sketching tool. *ICSE '11 Proceedings of the 33rd International Conference on Software Engineering*, 1028-1030.

[11] Mehra, A., Grundy, J., & Hosking, J. (2004). Supporting collaborative software design with a plug-in, web services-based architecture. *Workshop on Directions in Software Engineering*, 13-20.

[12] *Microsoft Visio 2010: A diagram is worth a thousand words*. (n.d.). Retrieved April 24, 2011, from http://www.microsoft.com/presspass/features/2009/dec09/12-17visio2010.mspx

[13] *Microsoft Visio 2010*. (n.d.). Retrieved April 24, 2011, from http://www.osalt.com/visio

[14] Fowler, M. (2003). *UML distilled: A brief guide to the standard object modeling language (3rd Ed.)*. Addison Wesley.

[15] Deitel, H., & Deitel, P. (2005). *Java how to program (6th Ed.)*. Prentice Hall.

[16] *What is a UML object diagram*. (n.d.). Retrieved December 1, 2010, from http://www.smartdraw.com/resources/tutorials/uml-component-diagram/#/resources/tutorials/UML-Object-Diagrams

[17] *What is a UML sequence diagram*. (n.d.). Retrieved December 1, 2010, from http://www.smartdraw.com/resources/tutorials/uml-sequence-diagram/

[18] *UML basics: The sequence diagram*. (n.d.). Retrieved December 1, 2010, from http://www.ibm.com/developerworks/rational/library/3101.html

[19] *UML 2 deployment diagrams*. (n.d.). Retrieved December 1, 2010, from http://www.agilemodeling.com/artifacts/deploymentDiagram.htm

[20] *UML use case diagrams*. (n.d.). Retrieved December 1, 2010, from http://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

[21] *UML 2 state machine diagrams*. (n.d.). Retrieved December 1, 2010, from http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

[22] *State machine diagrams*. (n.d.). Retrieved December 1, 2010, from http://www.uml-diagrams.org/state-machine-diagrams.html

[23] *UML 2 activity diagrams*. (n.d.). Retrieved December 1, 2010, from http://www.agilemodeling.com/artifacts/activityDiagram.htm

[24] *UML 2 communication diagrams*. (n.d.). Retrieved December 1, 2010, from http://www.agilemodeling.com/artifacts/communicationDiagram.htm

[25] *Composite structure diagram*. (n.d.). Retrieved December 1, 2010, from http://www.visual-paradigm.com/VPGallery/diagrams/CompositeStructureDiagram.html

[26] *UML 2 interaction overview diagrams*. (n.d.). Retrieved December 1, 2010, from http://www.agilemodeling.com/artifacts/interactionOverviewDiagram.htm

[27] *UML 2 interaction overview diagram*. (n.d.). Retrieved December 1, 2010, from http://www.sparxsystems.com/resources/uml2_tutorial/uml2_interactionoverviewdiagram.html

[28] Hearn, D., & Baker, M. (1996). *Computer graphics C version (2nd Ed.)*. Prentice Hall.

[29] Page, J. (2009). *Line segment definition (coordinate geometry)*. Retrieved April 25, 2011, from http://www.mathopenref.com/coordsegment.html

[30] Page, J. (2009). *Rectangle and its properties (coordinate geometry)*. Retrieved April 25, 2011, from http://www.mathopenref.com/coordrectangle.html

[31] Page, J. (2009). *Ellipse*. Retrieved April 25, 2011, from http://www.mathopenref.com/ellipse.html

[32] Page, J. (2009). *Circle*. Retrieved April 25, 2011, from http://www.mathopenref.com/circle.html

[33] Page, J. (2009). *Polygon*. Retrieved April 25, 2011, from http://www.mathopenref.com/polygon.html

[34] Steenbergen, T. (2009). *Analysis of using browser-native technology to build rich internet applications for image manipulation*. M.S. Thesis. University of Leiden.

[35] Lubbers, P., Albers, B., & Salim, F. (2010). Using the HTML Canvas API. In *Pro HTML5 programming: Powerful APIs for richer internet application development* (pp. 25-64). Apress.

[36] Greenberg, S., & Marwood, D. (1994). Real time groupware as a distributed system: Concurrency control and its effect on the interface. *Proceedings of the ACM CSCW Conference on Computer Supported Cooperative Work*.

[37] Fraser, N. (2009). Differential synchronization. *DocEng '09*. Retrieved from http://research.google.com/pubs/archive/35605.pdf

[38] Nichols, D., Curtis, P., Dixon, M., & Lamping, J. (1995). High-latency, low-bandwidth windowing in the Jupiter collaboration systems. *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, 111-120.

[39] Day-Richter, J. (2010). *What's different about the new Google Docs: Making collaboration fast*. Retrieved from http://googledocs.blogspot.com/2010/09/whats-different-about-new-google-docs_23.html

[40] Ellis, C.A., & Gibbs, S.J. (1989). Concurrency control in groupware systems. *Proc. 1989 ACM SIGMOD International Conference on the Management of Data*, 399-407.

[41] Antunes, P., & Guimaraes, N. (1994). *Multiuser interface design in CSCW systems*. University of Bologna.

[42] Petersen, D., Dolog, P., Pedersen, E., Pedersen, K., & Lin, Y. (2009). Real-time synchronization for creativity in distributed innovation team. *Proceedings of the Workshop on Methods & Tools for Computer Supported Collaborative Creativity Process: Linking Creativity & Informal Learning*.

X. Appendix

SOURCE CODES:

index.jsp

```
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<%@page session="true"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
  <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
    <title>Cordie</title>
    <link rel="shortcut icon"
type="image/x-icon"
href="images/icon.gif">
    <link rel="stylesheet"
type="text/css" href="css/cordie.css"/>
  </head>
  <body>
   <div id="main-header">
     <img src="images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
     <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
   </div>
   <div id="menu">
     <ul id="menulist">
       <li class="home"><a
href="index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
       <li><a href="login.jsp">Log
In</a></li>
       <li><a
href="signup.jsp">Register</a></li>
       <li id="empty"></li>
<%  } else { %>
       <li><a
href="members/profile.jsp">Profile</a>
</li>
       <li><a
href="members/diagrams.jsp">Diagram
s</a></li>
       <li><a href="logout.jsp">Log
Out</a></li>
       <li id="empty2"></li>
<%  } %>
     </ul>
   </div>
   <div id="main-content">
       <p>Welcome to Cordie!</p>
       <p>Cordie is a web-based UML
diagram creation application that
           allows users to collaborate
with each other in real-time.
           Once signed in, users can
create new diagrams, edit, and
           collaborate with others.</p>
       <h3>Draw Diagrams</h3>
       <p>
           Diagrams can be created easily
since Cordie provides the
           necessary editing tools to assist
users in editing their
           diagrams. Adding UML
notational elements and other basic
shapes
           is supported. Users can also
create diagrams aside from UML
           diagrams, such as mind maps
and sketches.
       </p>
       <h3>Web-based</h3>
       <p>Anyone can sign in to Cordie
and start creating diagrams. There's
           no installation required; users
only need a web browser and
           access to the Internet to be
able to start editing diagrams and
           collaborate with others.
       </p>
       <h3>Collaborate Real-time</h3>
       <p>Multiple users can share and
simultaneously edit a single
           diagram. Team members are
able to collaborate in real-time even
           if they are physically separated.
       </p>
       <br/>
       <br/>

       <div id="main-footer">
         All Rights Reserved. Copyright
© 2012 University of the Philippines
Manila
       </div>
   </div>
  </body>
</html>
```

denyAccess.jsp

```
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
  <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
    <title>Cordie - Error</title>
    <link rel="shortcut icon"
type="image/x-icon"
href="images/icon.gif">
    <link rel="stylesheet"
type="text/css" href="css/cordie.css"/>
  </head>
  <body>
   <div id="main-header">
     <img src="images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
     <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
   </div>
   <div id="menu">
     <ul id="menulist">
       <li class="home"><a
href="index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
       <li><a href="login.jsp">Log
In</a></li>
       <li><a
href="signup.jsp">Register</a></li>
       <li id="empty"></li>
<%  } else { %>
       <li><a
href="members/profile.jsp">Profile</a>
</li>
       <li><a
href="members/diagrams.jsp">Diagram
s</a></li>
       <li><a href="logout.jsp">Log
Out</a></li>
       <li id="empty2"></li>
<%  } %>
     </ul>
   </div>
   <div id="main-content">
     <p>Only members are allowed to
view this page. Please <a
href="login.jsp">log in</a></p>
       <br/>
       <br/>

       <div id="main-footer">
         All Rights Reserved. Copyright
© 2012 University of the Philippines
Manila
       </div>
```

```
    </div>
  </body>
</html>
```

login.jsp

```
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<%@page session="true"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
  <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
    <title>Log In</title>
    <link rel="shortcut icon"
type="image/x-icon"
href="images/icon.gif">
    <link rel="stylesheet"
type="text/css" href="css/cordie.css"/>
  </head>
  <body>
   <div id="main-header">
     <img src="images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
      <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
    </div>
    <div id="menu">
      <ul id="menulist">
        <li class="home"><a
href="index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
        <li><a href="login.jsp">Log
In</a></li>
        <li><a
href="signup.jsp">Register</a></li>
        <li id="empty"></li>
<%  } else { %>
        <li><a
href="members/profile.jsp">Profile</a>
</li>
        <li><a
href="members/diagrams.jsp">Diagram
s</a></li>
        <li><a href="logout.jsp">Log
Out</a></li>
        <li id="empty2"></li>
<%  } %>
      </ul>
    </div>
    <div id="main-content">
<% if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
     <div>
       <form name="loginForm"
action="LogInExec" method="POST">
         <br/><br/><br/>
         <table width="300"
align="center" border="0"
cellpadding="0">
           <thead>
             <tr>
               <th colspan="2">Log In
Form</th>
             </tr>
           </thead>
           <tbody>
             <tr>
               <td
width="112"><b>Username</b></td>
               <td width="188"><input
type="text" name="username" value=""
size="25" /></td>
             </tr>
             <tr>

<td><b>Password</b></td>
               <td><input
type="password" name="password"
value="" size="25" /></td>
             </tr>
             <tr>
               <td></td>
               <td><input
type="submit" value="Log In"
name="Submit" /></td>
             </tr>
             <tr>
               <td> </td>
               <td> </td>
             </tr>
             <tr>
               <td colspan="2"
style="text-align: center"><a
href="signup.jsp">Create an
account</a></td>
             </tr>
           </tbody>
         </table>
       </form>
     </div>
<% } else {

response.sendRedirect("index.jsp");
  } %>
       <br/>
       <br/>

       <div id="main-footer">
         All Rights Reserved. Copyright
© 2012 University of the Philippines
Manila
       </div>
   </div>

  </body>
</html>
```

loginfailed.jsp

```
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<%@page session="true"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
  <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
    <title>Log In Failed</title>
    <link rel="shortcut icon"
type="image/x-icon"
href="images/icon.gif">
    <link rel="stylesheet"
type="text/css" href="css/cordie.css"/>
  </head>
  <body>
   <div id="main-header">
     <img src="images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
      <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
    </div>
    <div id="menu">
      <ul id="menulist">
        <li class="home"><a
href="index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
        <li><a href="login.jsp">Log
In</a></li>
        <li><a
href="signup.jsp">Register</a></li>
        <li id="empty"></li>
<%  } else { %>
        <li><a
href="members/profile.jsp">Profile</a>
</li>
        <li><a
href="members/diagrams.jsp">Diagram
s</a></li>
        <li><a href="logout.jsp">Log
Out</a></li>
        <li id="empty2"></li>
<%  } %>
      </ul>
    </div>
    <div id="main-content">
```

```
<% if(session == null || session.getAttribute("USERNAME") == null) { %>
    <div>
      <form name="loginForm" action="LogInExec" method="POST">
        <br/>
        <div style="text-align:center">Log in failed. Please try again.</div>
        <br/>
        <table width="300" align="center" border="0" cellpadding="0">
          <thead>
            <tr>
              <th colspan="2">Log In Form</th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td width="112"><b>Username</b></td>
              <td width="188"><input type="text" name="username" value="" size="25" /></td>
            </tr>
            <tr>

<td><b>Password</b></td>
              <td><input type="password" name="password" value="" size="25" /></td>
            </tr>
            <tr>
              <td></td>
              <td><input type="submit" value="Log In" name="Submit" /></td>
            </tr>
            <tr>
              <td> </td>
              <td> </td>
            </tr>
            <tr>
              <td colspan="2" style="text-align: center"><a href="signup.jsp">Create an account</a></td>
            </tr>
          </tbody>
        </table>
      </form>
    </div>
<% } else {

response.sendRedirect("./members/home.jsp");
  } %>
      <br/>
      <br/>

      <div id="main-footer">
        All Rights Reserved. Copyright © 2012 University of the Philippines Manila
      </div>
    </div>
  </body>
</html>
```

---

logout.jsp

```
<% session.invalidate();%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Log Out</title>
    <link rel="shortcut icon" type="image/x-icon" href="images/icon.gif">
    <link rel="stylesheet" type="text/css" href="css/cordie.css"/>
  </head>
  <body>
   <div id="main-header">
    <img src="images/cordie banner.png">
   </div>
   <div id="menu">
    <ul id="menulist">
     <li class="home"><a href="index.jsp">Home</a></li>
     <li><a href="login.jsp">Log In</a></li>
     <li><a href="signup.jsp">Register</a></li>
     <li><a href="logout.jsp">Log Out</a></li>
     <li id="empty2"></li>
    </ul>
   </div>
   <div id="main-content">
    <p>You are now logged out. <a href="login.jsp">Log in</a> again</p>
     <br/>
     <br/>

     <div id="main-footer">
       All Rights Reserved. Copyright © 2012 University of the Philippines Manila
     </div>
   </div>
  </body>
</html>
```

---

signup.jsp

```
<%@page import="org.apache.commons.lang3.StringEscapeUtils"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.io.*"%>
<%@page import="java.util.*"%>
<%@page import="javax.servlet.*"%>
<%@page import="javax.servlet.http.*"%>
<%@page import="org.apache.commons.fileupload.*"%>
<%@page import="org.apache.commons.fileupload.disk.*"%>
<%@page import="org.apache.commons.fileupload.servlet.*"%>
<%@page import="java.sql.*"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Sign Up</title>
    <link rel="shortcut icon" type="image/x-icon" href="images/icon.gif">
    <link rel="stylesheet" type="text/css" href="css/cordie.css"/>
  </head>
  <body>
   <div id="main-header">
    <img src="images/cordie banner.png">
<%  if(session != null && session.getAttribute("USERNAME") != null) { %>
    <div id="usernameDisplay">You are logged in <%= session.getAttribute("FIRSTNAME") %>.</div>
<%  } %>
   </div>
   <div id="menu">
    <ul id="menulist">
     <li class="home"><a href="index.jsp">Home</a></li>
<%  if(session == null || session.getAttribute("USERNAME") == null) { %>
     <li><a href="login.jsp">Log In</a></li>
     <li><a href="signup.jsp">Register</a></li>
     <li id="empty"></li>
<%  } else { %>
```

```
        <li><a
href="members/profile.jsp">Profile</a>
</li>
        <li><a
href="members/diagrams.jsp">Diagram
s</a></li>
        <li><a href="logout.jsp">Log
Out</a></li>
        <li id="empty2"></li>
<%  } %>
       </ul>
    </div>
    <div id="main-content">
<%
  HashMap<String, String> errors = new
HashMap<String, String>();;
  boolean valid = true;
  String username = "";
  String password1 = "";
  String password2 = "";
  String firstname = "";
  String lastname = "";
  String email = "";


if(request.getMethod().equals("POST"))
{
     try {

Class.forName("com.mysql.jdbc.Driver")
;
        Connection connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");
        //Statement statement =
connection.createStatement();

        File file ;
        int maxFileSize = 5000 * 1024;
        int maxMemSize = 5000 * 1024;
        String filePath =
"uploaded_images/";
        FileItem uploadedImage = null;
        String extension = "";

        // Verify the content type
        String contentType =
request.getContentType();

if((contentType.indexOf("multipart/for
m-data") >= 0)) {
          DiskFileItemFactory factory =
new DiskFileItemFactory();

factory.setSizeThreshold(maxMemSize);
          factory.setRepository(new
File(""));

          ServletFileUpload upload = new
ServletFileUpload(factory);
          upload.setSizeMax( maxFileSize
);
          try {
            // Parse the request to get
file items.
            List fileItems =
upload.parseRequest(request);
            // Process the uploaded file
items
            Iterator i =
fileItems.iterator();

            while (i.hasNext()) {
              FileItem item = (FileItem)
i.next();

              if(item.isFormField()) {

if(item.getFieldName().equals("usernam
e")) {
                  username =
item.getString();
                } else
if(item.getFieldName().equals("passwor
d1")) {
                  password1 =
item.getString();
                } else
if(item.getFieldName().equals("passwor
d2")) {
                  password2 =
item.getString();
                } else
if(item.getFieldName().equals("firstnam
e")) {
                  firstname =
item.getString();
                } else
if(item.getFieldName().equals("lastnam
e")) {
                  lastname =
item.getString();
                } else
if(item.getFieldName().equals("email"))
{
                  email =
item.getString();
                }
              } else {
                String fileName =
item.getName();

                if(!fileName.equals("")) {

if(fileName.lastIndexOf(".") >= 0) {
                    extension =
fileName.substring(fileName.lastIndexO
f("."));
                  }

                  uploadedImage = item;
                }
              }
            } // end while

            if(username.equals("")) {
              errors.put("username",
"Please enter a username");
              valid = false;
            } else {
              //check if username is
already taken
              String sql = "SELECT
COUNT(*) FROM user WHERE username
= ? ";
              PreparedStatement stmt =
connection.prepareStatement(sql);
              stmt.setString(1,
username);
              ResultSet rs =
stmt.executeQuery();
              //ResultSet rs =
statement.executeQuery("SELECT
COUNT(*) "
              //       + "FROM user
WHERE username = '" + username + "'");
              rs.next();
              if(rs.getInt(1) > 0) {
                errors.put("username",
"The username " + username +
                    " is already taken by
another user.");
                valid = false;
              }
            }
            if(password1.equals("") ) {
              errors.put("password1",
"Please enter a valid password");
              valid = false;
            }
            if(!password1.equals("") &&
(password2.equals("") ||

!password1.equals(password2))) {
              errors.put("password2",
"Please make sure your passwords
match.");
              password2 = "";
              valid = false;
            }
          } catch(Exception ex) {
            System.out.println(ex);
          }
        } else {
        }

        if(valid) {
          String sql = "INSERT INTO user
(username, password, firstname, "
              + "lastname, email,
displaypic) VALUES( ?, md5(?), ?, ?, ?,
?)";
          PreparedStatement stmt =
connection.prepareStatement(sql);
          stmt.setString(1, username);
          stmt.setString(2, password1);
          stmt.setString(3, firstname);
          stmt.setString(4, lastname);
          stmt.setString(5, email);
          stmt.setString(6, username +
extension);
          stmt.executeUpdate();
          /*int i =
statement.executeUpdate("INSERT
INTO user (username, "
```

```
                + "password, firstname,
lastname, email, displaypic)"
                + " VALUES('" + username +
"', md5('" + password1 + "'), '"
                + firstname + "', '" +
lastname + "', '" + email + "', '"
                + username + extension +
"')");*/

          if(uploadedImage != null)
            uploadedImage.write(new
File(filePath + username + extension)) ;
%>
  <table align="center">
    <thead>
      <tr>
        <th colspan="2">User
Registration Successful</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Username: </td>
        <td><%= username %></td>
      </tr>
      <tr>
        <td>Firstname: </td>
        <td><%= firstname %></td>
      </tr>
      <tr>
        <td>Lastname: </td>
        <td><%= lastname %></td>
      </tr>
      <tr>
        <td>Email: </td>
        <td><%= email %></td>
      </tr>
      <tr><td> </td></tr>
      <tr>
        <td colspan="2">You can now
<a href="login.jsp">log in</a>.</td>
      </tr>
    </tbody>
  </table>
      <br/>
      <br/>
      <div id="main-footer">
        All Rights Reserved. Copyright
© 2012 University of the Philippines
Manila
      </div>
   </div>
  </body>
</html>
<%
          return;
       } else { %>
    <div>
      <p>Please fill in the necessary
fields.</p>
    </div>
<%      }
    } catch (ClassNotFoundException e) {
      System.err.println("Driver Error");
    } catch (SQLException e) {
      System.err.println("SQLException:
" + e.getMessage());
    }
  } %>
    <form name="signupForm"
action="signup.jsp" method="POST"
enctype="multipart/form-data">
      <table align="center">
        <thead>
          <tr>
            <th colspan="2">User
Registration</th>
          </tr>
          <tr>
            <th></th>
            <th><font size="1"
color="red"><sup>*</sup> Required
Fields</font></th>
          </tr>
        </thead>
        <tbody>
          <tr>

<td>Username<sup>*</sup></td>
            <td><input type="text"
id="username" name="username"
value="" size="25" maxlength="25"/>
              <br><font
color="red"><%=
(errors.get("username") == null) ? "" :
errors.get("username") %></font>
            </td>
          </tr>
          <tr>

<td>Password<sup>*</sup></td>
            <td><input
type="password" id="password1"
name="password1" value="" size="25"
maxlength="25" />
              <br><font
color="red"><%=
(errors.get("password1") == null) ? "" :
errors.get("password1") %></font>
            </td>
          </tr>
          <tr>
            <td>Retype
Password<sup>*</sup></td>
            <td><input
type="password" id="password2"
name="password2" value="" size="25"
maxlength="25" />
              <br><font
color="red"><%=
(errors.get("password2") == null) ? "" :
errors.get("password2") %></font>
            </td>
          </tr>
          <tr>
            <td>First Name</td>
            <td><input type="text"
id="firstname" name="firstname"
value="" size="25" maxlength="25"
/></td>
          </tr>
          <tr>
            <td>Last Name</td>
            <td><input type="text"
id="lastname" name="lastname"
value="" size="25" maxlength="25"
/></td>
          </tr>
          <tr>
            <td>Email</td>
            <td><input type="text"
id="email" name="email" value=""
size="25" maxlength="50" /></td>
          </tr>
          <tr>
            <td>Display Picture</td>
            <td><input type="file"
name="displaypic" value=""
size="25"/></td>
          </tr>
          <tr>
            <td><input type="submit"
value="Submit" name="submit" /></td>
            <td><input type="reset"
value="Reset" name="reset" /></td>
          </tr>
        </tbody>
      </table>
    </form>
    <% if (!valid) { %>
    <script type="text/javascript">

(document.getElementById("username"
)).value = '<%= username %>';

(document.getElementById("password1
")).value = '<%= password1 %>';

(document.getElementById("password2
")).value = '<%= password2 %>';

(document.getElementById("firstname")
).value = '<%= firstname %>';

(document.getElementById("lastname")
).value = '<%= lastname %>';

(document.getElementById("email")).va
lue = '<%= email %>';
    </script>
    <% } %>
      <br/>
      <br/>
      <div id="main-footer">
        All Rights Reserved. Copyright
© 2012 University of the Philippines
Manila
      </div>
   </div>
  </body>
</html>
```

---

cordie.css

```
body {
  color: #333333;
```

```
   font-family: Verdana,Helvetica,Sans-
Serif;
   font-size: 11px;
   margin: 0 auto;
   padding: 0;
}

body {
   background: repeat scroll 0 0
#BF6F41;
   /* IE10 Consumer Preview */
   background-image: -ms-linear-
gradient(left, #BF6F41 0%, #FFC59E
25%, #F0F0F0 71%, #D9A787 100%);
   /* Mozilla Firefox */
   background-image: -moz-linear-
gradient(left, #BF6F41 0%, #FFC59E
25%, #F0F0F0 71%, #D9A787 100%);
   /* Opera */
   background-image: -o-linear-
gradient(left, #BF6F41 0%, #FFC59E
25%, #F0F0F0 71%, #D9A787 100%);
   /* Webkit (Safari/Chrome 10) */
   background-image: -webkit-
gradient(linear, left top, right top, color-
stop(0, #BF6F41), color-stop(0.25,
#FFC59E), color-stop(0.71, #F0F0F0),
color-stop(1, #D9A787));
   /* Webkit (Chrome 11+) */
   background-image: -webkit-linear-
gradient(left, #BF6F41 0%, #FFC59E
25%, #F0F0F0 71%, #D9A787 100%);
   /* W3C Markup, IE10 Release Preview
*/
   background-image: linear-gradient(to
right, #BF6F41 0%, #FFC59E 25%,
#F0F0F0 71%, #D9A787 100%);
}

#main-header {
   margin: 0 auto;
   padding: 10px 0;
   width: 980px;
   position: relative;
}

#main-content {
   background: none repeat scroll 0 0
#FFFFFF;
   background: none repeat scroll 0 0
rgba(255, 255, 255, 0.45);
   border: 1px solid #D2D2DE;
   margin: 0 auto;
   padding: 15px 15px 0;
   width: 950px;
}

#main-footer {
   background: none repeat scroll 0 0
transparent;
   border-top: 1px solid #C0C0C0;
   padding: 10px 0;
   text-align: center;
   /*width: 950px;
   margin: 0;*/
   width: auto;
   margin: 0 15px;
}

#usernameDisplay {
   bottom: 0;
   color: #000000;
   font-weight: bold;
   padding: 5px 10px 5px 5px;
   position: absolute;
   right: 0;
   width: auto;
}

#menu {
   background: none repeat scroll 0 0
#FFFFFF;
   border: 1px solid #D2D2DE;
   margin: 0 auto 10px;
   width: 980px;
   font-family: arial,helvetica,sans-serif;
   text-align: right;
}

#menu:after {
   clear: both;
   color: transparent;
   content: ".";
   display: block;
   font-size: 1px;
   line-height: 1px;
}

ul#menulist {
   background-color: #000000;
   font-size: 12px;
   list-style: none outside none;
   margin: 0;
   padding: 0;
}

ul#menulist li {
   background-color: #000000;
   border-left: 1px solid #CCCCCC;
   float: left;
   height: 30px;
   margin: 0;
   padding: 0;
   text-align: left;
   width: 104px;
}

ul#menulist li.home {
   border-left: medium none;
}

ul#menulist li#empty{
   width: 665px;
}

ul#menulist li#empty2{
   width: 560px;
}

#menu > ul#menulist li a {
   height: auto;
   width: auto;
}

ul#menulist li a {
   color: #CCCCCC;
   display: block;
   font-size: 11px;
   height: 1%;
   padding: 7px 5px 10px 7px;
   text-decoration: none;
   text-shadow: 0 0 0 #000000;
}

a:link {
   background-color: transparent;
   color: #000000;
   text-decoration: none;
}

a {
   background-color: transparent;
   color: #000000;
   font-weight: bold;
   text-decoration: none;
}

a:visited {
   background-color: transparent;
   color: #000000;
   text-decoration: none;
}

a:active {
   background-color: transparent;
   color: #FF6600;
   text-decoration: underline;
}

a:hover, a span:hover {
   background-color: transparent;
   color: #FF6600;
   text-decoration: underline;
}

#menu > ul#menulist li:hover {
   background-color: #FF6600;
   color: white;
}

#menu > ul#menulist li#empty:hover,
#menu > ul#menulist li#empty2:hover {
   background-color: #000000;
   color: white;
}

#menu > ul#menulist li:hover > a {
   color: white;
}

.displayPicContainer {
   border: 1px solid;
   height: 100px;
   width: 100px;
   margin: auto;
}

.displayPic {
```

```
   background: white;
   height: 100px;
   width: 100px;
}

.addCollaboratorSection {
   display: none;
}

#diagramListTable ul {
   float: left;
   list-style-type: none;
   margin: auto 5px;
   padding: 0;
   width: 70px;
}
```

---

cordie-editor.css

```
#main-content {
   background: none repeat scroll 0 0
transparent;
   width: 980px;
   padding: 0;
   border: none;
   display: none;
}

#content {
   clear: both;
   display: block;
   position: relative;
   height: auto;
   width: auto;
   margin: 0;
}

#canvasArea {
   border: 1px solid;
   display: block;
   position: relative;
   height: 500px;
   margin-right: 10px;
   overflow: auto;
   float: left;
   width: 720px;
}

#header {
   width: 725px;
   height: 20px;
}

#error {
   background: none no-repeat scroll 0 0
rgba(0, 0, 0, 0.6);
   border: 1px solid #D2D2DE;
   color: #CCCCCC;
   padding: 10px;
}

#drawingArea {
   background: none repeat scroll 0 0
white;
   position: absolute;
}

#drawingAreaTemp {
   position: absolute;
}

#rightBoxes {
   /*background: none repeat scroll 0 0
green;
   border: 1px solid;*/
   display: block;
   position: relative;
   margin: 0;
   padding: 0;
   float: left;
}

#collaboratorsAndCurrentUsersTable
a:link {
   color: #FFFFFF;
}

#rightBoxes * {
   font-size: 10px;
}

#rightBox1, #rightBox2 {
   background: repeat scroll 0 0
#000000;
   /*background: -moz-radial-
gradient(45deg, #8F8F8F, #000000)
repeat scroll 0 0 transparent;
   background: -webkit-gradient(radial,
center center, 3, center center, 1000,
from(#8F8F8F), to(#000000));*/
   border: 1px solid #FFFFFF;
   display: block;
   position: relative;
   height: 245px;
   width: 245px;
   margin: 0 0 5px;
   padding: 0;
   color: #CCCCCC;

   /* IE10 Consumer Preview */
background-image: -ms-radial-
gradient(center, circle farthest-side,
#595959 0%, #000000 100%);

/* Mozilla Firefox */
background-image: -moz-radial-
gradient(center, circle farthest-side,
#595959 0%, #000000 100%);

/* Opera */
background-image: -o-radial-
gradient(center, circle farthest-side,
#595959 0%, #000000 100%);

/* Webkit (Safari/Chrome 10) */
background-image: -webkit-
gradient(radial, center center, 0, center
center, 487, color-stop(0, #595959),
color-stop(1, #000000));

/* Webkit (Chrome 11+) */
background-image: -webkit-radial-
gradient(center, circle farthest-side,
#595959 0%, #000000 100%);

/* W3C Markup, IE10 Release Preview
*/
background-image: radial-
gradient(circle farthest-side at center,
#595959 0%, #000000 100%);
}

#footer {
   background: none repeat scroll 0 0
#DDDDDD;
   clear: both;
}

/*********************************
*********************************
************/

#toolSettings {
   border: 1px solid #FFFFFF;
   display: block;
   margin: 5px;
   padding: 0;
   position: relative;
   white-space: nowrap;
   /*height: 126px;
   overflow: hidden;*/
   overflow: visible;
   width: 235px;
   height: 158px;
}

#toolSettings * {
   font-size: 9px;
   /*color: #FFFFFF;
   font-weight: bold;*/
}

#toolSettings tr {
   border: 1px solid;
   /*background: red;*/
}

#toolSettings select {
   width: 60px;
}

.rgbaInput {
   width: 20px;
}

#selectedFillColor,
#selectedStrokeColor,
#selectedTextColor {
   border: 1px solid;
   width: 20px;
   height: 20px;
   background-color:
rgba(255,255,255,1);
}

#selectedFontsize, #selectedFontfamily
{
```

```
    width: 50px;
}

#shapeButtons {
    border: 1px solid;
    display: block;
    height: 70px;
    margin: 5px;
    overflow: auto;
    position: relative;
    /*border-radius: 6px 6px 6px 6px;*/
}

#shapeButtons > ul {
    list-style: none outside none;
    margin: 0;
    padding: 0;
}

#shapeButtons > ul > li {
    display: inline;
    float: left;
    margin: 0.15em;
}

#shapeButtons input {
    background-color: #E8E8E8;
    border-color: #FFFFFF #808080
#808080 #FFFFFF;
    border-radius: 5px 5px 5px 5px;
    border-style: solid;
    border-width: 1px;
    color: #0D2474;
    cursor: pointer;
    display: block;
    float: left;
    height: 22px;
    line-height: 2em;
    margin: 2px;
    padding: 2px;
    text-align: center;
    text-decoration: none;
    width: 22px;
}

#shapeButtons input:hover {
    background-color: #fbcbab;
}

#shapeButtons input.active {
    background-color: #f8ad7c;
}

/*********************************
*********************************
*************/

#boxMenu {
    /*border: 1px solid;
    overflow: auto;*/
    display: block;
    height: 20px;
    margin: 5px 5px 0;
    padding: 0;
    position: relative;
}

#boxMenu > ul {
    margin: 0;
    padding: 0;
    list-style-type: none;
}

#boxMenu > ul > li {
    display: inline;
    float: left;
    position: relative;
}

#boxMenu > ul > li > a {
    background: #FFFFFF center center
repeat-x;
    border: 1px solid #202020;
    color: black;
    display: inline;
    margin: 0;
    padding: 0 4px;
    text-align: center;
    text-decoration: none;
    width: auto;
    font: 11px normal;
}

#boxMenu > ul > li > a:hover {
    background: #fbcbab center center
repeat-x;
}

#boxMenu > ul > li > a.active {
    background-color: #f8ad7c;
}

#boxTab {
    border: 1px solid;
    height: 210px;
    width: 230px;
    margin: auto;
    /*border-radius: 6px 6px 6px 6px;*/
    overflow: auto;
}

/*********************************
*********************************
*************/

#propertiesBoxTabs {
    display: block;
    height: 15px;
    margin: 5px;
    overflow: hidden;
    padding: 0;
    position: relative;
}

#propertiesBoxTabs > ul {
    margin: 0;
    padding: 0;
}

#propertiesBoxTabs > ul > li {
    display: inline;
}

#propertiesBoxTabs > ul > li > a {
    background: none repeat-x scroll left
bottom #FFFFFF;
    border: 1px solid black;
    color: black;
    display: block;
    float: left;
    margin: 0;
    padding: 0 3px;
    text-align: center;
    text-decoration: none;
    width: auto;
    font: 8px normal;
}

#propertiesBoxTabs > ul > li > a:hover {
    background: #fbcbab center center
repeat-x;
}

#propertiesBoxTabs > ul > li > a.active {
    background-color: #f8ad7c;
}

#propertiesTable {
    display: block;
    position: relative;
    height: 183px;
    overflow-y: auto;
    overflow-x: hidden;
    /*border-radius: 6px 6px 6px 6px;
    overflow: auto;*/
}

#propertiesTable table tbody {
    width: 100px;
}

#propertiesTable input,
#propertiesTable select {
    width: 87px;
}

#propertiesTable input.rgbaInput {
    width: 20px;
}

#propertiesTable label {
}

#propertiesTable #attributesTable label
input {
    width: 13px;
}

#propertiesTable #operationsTable label
input {
    width: 13px;
}

#propertiesTable .upZOrderButton,
#propertiesTable .downZOrderButton {
    background-color: #E8E8E8;
    border-color: #FFFFFF #808080
#808080 #FFFFFF;
```

```
   border-style: solid;
   border-width: 1px;
   cursor: pointer;
   height: 11px;
   width: 11px;
   margin: 0 1px;
}

#propertiesTable
.upZOrderButton:hover,
#propertiesTable
.downZOrderButton:hover {
   background-color: #fbcbab;
}

/*********************************
*********************************
*************/

#collaboratorsAndCurrentUsersBoxTabs
{
   display: block;
   height: 15px;
   margin: 5px;
   overflow: hidden;
   padding: 0;
   position: relative;
}

#collaboratorsAndCurrentUsersBoxTabs
> ul {
   margin: 0;
   padding: 0;
}

#collaboratorsAndCurrentUsersBoxTabs
> ul > li {
   display: inline;
}

#collaboratorsAndCurrentUsersBoxTabs
> ul > li > a {
   background: none repeat-x scroll left
bottom #FFFFFF;
   border: 1px solid black;
   color: black;
   display: block;
   float: left;
   margin: 0;
   padding: 0 3px;
   text-align: center;
   text-decoration: none;
   width: auto;
   font: 10px normal;
}

#collaboratorsAndCurrentUsersBoxTabs
> ul > li > a:hover {
   background: #fbcbab center center
repeat-x;
}

#collaboratorsAndCurrentUsersBoxTabs
> ul > li > a.active {
   background-color: #f8ad7c;
}

#collaboratorsAndCurrentUsersTable {
   display: block;
   height: 183px;
   overflow: auto;
   position: relative;
   border-radius: 6px 6px 6px 6px;
}
/*********************************
*********************************
*************/

#currentusersSection,
#collaboratorsSection {
   margin: 5px;
}

#collaborators {
   border-collapse: collapse;
}

.onlineSign tr:nth-child(1), .onlineSign
tr:nth-child(4) {
   font-weight: bold;
   color: #FF6600;
}

.offlineSign tr:nth-child(4) {
   font-weight: bold;
}

#collaborators tbody {
   border-top: 15px solid transparent;
}

#addCollaboratorSection {
   display: block;
   position: relative;
   margin: 5px;
   padding: 5px;
}

#addCollaboratorSection * {
   position: relative;
   margin-right: auto;
   margin-left: auto;
}

.removeCollaboratorButton {

}

.userinfo {
   background: none repeat scroll 0 0
silver;
   border: 4px solid;
   display: none;
   position: fixed;
   width: 400px;
   height: 200px;
   left: 50%;
   top: 50%;
   margin-left: -200px;
   margin-top: -100px;
}

.userinfo > table {
   margin: auto;
}

/*********************************
*********************************
*************/

#propertiesBox {
   display: block;
}

#collaboratorsAndCurrentUsers {
   display: none;
}

#appliedOpsTable {
   display: none;
   height: 210px;
   overflow-x: hidden;
   overflow-y: scroll;
   width: 230px;
}

#appliedOpsTable > table {
   width: 220px;
   margin: auto;
}

.displayPicContainer {
   border: 1px solid;
   height: 100px;
   width: 100px;
   margin: auto;
}

.displayPic {
   background: white;
   height: 100px;
   width: 100px;
}

/*********************************
*********************************
*************/

#deleteButton, #saveAsButton {
   background-color: #E8E8E8;
   border-color: #FFFFFF #808080
#808080 #FFFFFF;
   /*border-radius: 5px 5px 5px 5px;*/
   border-style: solid;
   border-width: 1px;
   cursor: pointer;
   display: block;
   float: left;
   height: 15px;
   line-height: 2em;
   margin: 2px;
   padding: 2px;
   text-align: center;
   text-decoration: none;
   width: 15px;
}
```

```
#deleteButton:hover,
#saveAsButton:hover {
   background-color: #fbcbab;
}

.displayPicContainer2 {
   border: 1px solid;
   margin: auto;
}

.displayPic2 {
   background: white;
}

.displayPicContainer2, .displayPic2 {
   height: 45px;
   width: 45px;
}

/*********************************
*********************************
*************/

.color, #propertiesTable input.color {
   width: 45px;
}

#diagramTitle {
   font-size: 11px;
   font-weight: bold;
   text-align: center;
   color: #FF6600;
}

#textAreaPopUp {
   visibility: hidden;
   position: absolute;
   width: 200px;
   height: 80px;
}

#textValueInput {
   resize: none;
   background: transparent;
   border: 1px solid;
}
```

diagrams.jsp

```
<%@page
import="org.apache.commons.lang3.Str
ingEscapeUtils"%>
<%@page contentType="text/html"
pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
   <head>
      <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
      <title>Cordie - Diagrams</title>
      <link rel="shortcut icon"
type="image/x-icon"
href="../images/icon.gif">
      <script type="text/javascript"
src="diagramspage.js"></script>
      <link rel="stylesheet"
type="text/css"
href="../css/cordie.css"/>
   </head>
   <body onLoad="init('<%=
StringEscapeUtils.escapeEcmaScript((Str
ing) session.getAttribute("USERNAME"))
%>')">
      <div id="main-header">
        <img src="../images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
        <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
      </div>
      <div id="menu">
        <ul id="menulist">
          <li class="home"><a
href="../index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
          <li><a href="../login.jsp">Log
In</a></li>
          <li><a
href="../signup.jsp">Register</a></li>
          <li id="empty"></li>
<%  } else { %>
          <li><a
href="profile.jsp">Profile</a></li>
          <li><a
href="diagrams.jsp">Diagrams</a></li>
          <li><a href="../logout.jsp">Log
Out</a></li>
          <li id="empty2"></li>
<%  } %>
        </ul>
      </div>
      <div id="main-content">
        <br/>
        <div>
          <input type="button"
value="Refresh List"
onclick="javascript:getDiagramList()"/>
          <input id="newDiagram"
type="button" value="New Diagram"
onclick="javascript:newDiagram()"/>
          <br/><br/>

          <table id="diagramListTable"
border="1" cellspacing="0"
cellpadding="2">
            <thead>
              <tr>
                <th
width="175px">Title</th>
                <th
width="200px">Description</th>
                <th
width="150px">Creator</th>
                <th width="175px">Date
Created</th>
                <th
width="175px">Collaborators</th>
                <th width="45px"></th>
                <th width="45px"></th>
                <th width="45px"></th>
              </tr>
            </thead>
            <tbody>
            </tbody>
          </table>
        </div>

        <br/>
        <br/>

        <div id="main-footer">
          All Rights Reserved. Copyright ©
2012 University of the Philippines
Manila
        </div>
      </div>
   </body>
</html>
```

diagramspage.js

```
var USERNAME;
var dlist = [];
var selectedId = null;
var URL = "CordieEditor";

function init(username) {
   USERNAME = username;
   getDiagramList();
}

function createXMLHttpRequest() {
  try {return new XMLHttpRequest();}
catch(e) {}
  try {return new
ActiveXObject("Msxml2.XMLHTTP");}
catch (e) {}
  try {return new
ActiveXObject("Microsoft.XMLHTTP");}
catch (e) {}

  alert("Sorry, you're browser does not
support XMLHttpRequest");
  return null;
}

function getDiagramList() {
   var xhReq = new
createXMLHttpRequest();
   xhReq.open("POST", 'DiagramList',
false);
   xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
```

```
   xhReq.send("");

   //alert(xhReq.responseText);

   try {
     dlist =
JSON.parse(xhReq.responseText);
   } catch(e) {
     alert("Error getting diagram list.");
   }

   showList();
   cancelAddDiagram();
   cancelCopyDiagram();
}

function showList() {
   var dltable =
document.getElementById("diagramList
Table");
   var oldtbody =
dltable.getElementsByTagName("tbody"
)[0];

   var dltbody =
document.createElement("tbody");
   for(var i = 0; i < dlist.length; i++) {
     var dltr =
document.createElement("tr");

     for(var j = 1; j < dlist[i].length; j++) {
       var dltd =
document.createElement("td");
       if(j == 5) {
         var dltdUl1 =
document.createElement("ul");
         for(var k = 0; k <
dlist[i][j].length; k++) {
           var dltdUlLi =
document.createElement("li")

dltdUlLi.appendChild(document.createT
extNode(dlist[i][j][k]));


dltdUl1.appendChild(dltdUlLi);
           dltd.appendChild(dltdUl1);
         }

         if(dlist[i][3] == USERNAME) {
           var dltdUl2 =
document.createElement("ul");
           for(var k = 0; k <
dlist[i][j].length; k++) {
             if(dlist[i][3] == dlist[i][j][k])
{

             } else {
               dltdUlLi =
document.createElement("li")

               var dltdUlLiA =
document.createElement("a");

dltdUlLiA.setAttribute("href",
"javascript:removeCollaboratorReq("
                     + dlist[i][0] + ", '" +
dlist[i][j][k] + "')");

dltdUlLiA.setAttribute("class",
"removeCollaboratorButton");

dltdUlLiA.appendChild(document.create
TextNode("Remove"));

dltdUlLi.appendChild(dltdUlLiA);


dltdUl2.appendChild(dltdUlLi);

dltd.appendChild(dltdUl2);
             }
           }


//dltd.appendChild(document.createEle
ment("br"));

           var dltrAddButton =
document.createElement("input");
           dltrAddButton.type =
"button";
           dltrAddButton.value = "Add
Collaborator";

dltrAddButton.setAttribute("onclick",
"javascript:enableAddReqCollaborator("
+ i + ");");

dltrAddButton.setAttribute("style",
"margin: 5px;");

dltd.appendChild(dltrAddButton);

           var dltrAddCollaboratorDiv =
document.createElement("div");

dltrAddCollaboratorDiv.setAttribute("cla
ss", "addCollaboratorSection");

dltrAddCollaboratorDiv.setAttribute("id"
, "addCollaboratorSection" + i);


dltrAddCollaboratorDiv.appendChild(do
cument.createTextNode("Username:
"));

           var dltrAddText =
document.createElement("input");
           dltrAddText.type = "text";
           dltrAddText.size = "20";
           dltrAddText.value = "";

dltrAddCollaboratorDiv.appendChild(dlt
rAddText);

           var dltrAddColButton =
document.createElement("input");
           dltrAddColButton.type =
"button";
           dltrAddColButton.value =
"Add";

dltrAddColButton.setAttribute("onclick"
, "javascript:addCollaboratorReq("
                + dlist[i][0] + ", " + i + ");");

dltrAddCollaboratorDiv.appendChild(dlt
rAddColButton);

           var dltrCancelAddColButton
= document.createElement("input");
           dltrCancelAddColButton.type
= "button";

dltrCancelAddColButton.value =
"Cancel";

dltrCancelAddColButton.setAttribute("o
nclick",
"javascript:cancelAddCollaborator(" + i +
");");

dltrAddCollaboratorDiv.appendChild(dlt
rCancelAddColButton);


dltd.appendChild(dltrAddCollaboratorDi
v);
         }
       } else {
         var dltdContent =
document.createTextNode(dlist[i][j]);
         dltd.appendChild(dltdContent);
       }

       dltr.appendChild(dltd);
     }

     var removetd =
document.createElement("td");
     var removeButton =
document.createElement("a");
     if(dlist[i][3] == USERNAME) {

removeButton.setAttribute("href",
"javascript:deleteDiagram(" + i + ")");

removeButton.appendChild(document.c
reateTextNode("Delete"));
     } else {

removeButton.setAttribute("href",
"javascript:leaveDiagram(" + i + ")");

removeButton.appendChild(document.c
reateTextNode("Leave"));
     }

removetd.appendChild(removeButton);
     dltr.appendChild(removetd);

     var opentd =
document.createElement("td");
     var openButton =
document.createElement("a");
```

```
      openButton.setAttribute("href",
"javascript:openDiagram(" + dlist[i][0] +
")");

openButton.appendChild(document.cre
ateTextNode("Open"));
      opentd.appendChild(openButton);
      dltr.appendChild(opentd);

      var copytd =
document.createElement("td");
      var copyButton =
document.createElement("a");
      copyButton.setAttribute("href",
"javascript:enableCopyDiagram(" +
         dlist[i][0] + ", '" + dlist[i][1] + "')");

copyButton.appendChild(document.cre
ateTextNode("Copy"));
      copytd.appendChild(copyButton);
      dltr.appendChild(copytd);

      dltbody.appendChild(dltr);
   }

   // hidden row for adding a new
diagram
   var newDiagramRow =
document.createElement("tr");
   newDiagramRow.setAttribute("id",
"newDiagramRow");
   newDiagramRow.setAttribute("style",
"display: none;");
   var newDiagramC1 =
document.createElement("td");
   var newDiagramC2 =
document.createElement("td");
   var newDiagramC3 =
document.createElement("td");
   var newDiagramC4 =
document.createElement("td");
   var newDiagramC5 =
document.createElement("td");
   var newDiagramC6 =
document.createElement("td");

   var newDiagramTitleInput =
document.createElement("input");

newDiagramTitleInput.setAttribute("id",
"addTitle");

newDiagramTitleInput.setAttribute("typ
e", "text");

newDiagramTitleInput.setAttribute("val
ue", "New Diagram");

newDiagramTitleInput.setAttribute("siz
e", "20");

newDiagramTitleInput.setAttribute("dis
abled", "true");

newDiagramC1.appendChild(newDiagra
mTitleInput);

   var newDiagramDescriptionInput =
document.createElement("textarea");

newDiagramDescriptionInput.setAttribu
te("id", "addDescription");

newDiagramDescriptionInput.setAttribu
te("rows", "1");

newDiagramDescriptionInput.setAttribu
te("cols", "20");

newDiagramDescriptionInput.setAttribu
te("disabled", "true");

newDiagramC2.appendChild(newDiagra
mDescriptionInput);


newDiagramC3.appendChild(document.
createTextNode(USERNAME));

   var newDiagramAddButton =
document.createElement("input");

newDiagramAddButton.setAttribute("id
", "addButton");

newDiagramAddButton.setAttribute("ty
pe", "button");

newDiagramAddButton.setAttribute("va
lue", "Add");

newDiagramAddButton.setAttribute("o
nclick", "javascript:addNew()");

newDiagramAddButton.setAttribute("di
sabled", "true");

newDiagramC4.appendChild(newDiagra
mAddButton);

   var newDiagramCancelButton =
document.createElement("input");

newDiagramCancelButton.setAttribute(
"id", "cancelAddButton");

newDiagramCancelButton.setAttribute(
"type", "button");

newDiagramCancelButton.setAttribute(
"value", "Cancel");

newDiagramCancelButton.setAttribute(
"onclick",
"javascript:cancelAddDiagram()");

newDiagramCancelButton.setAttribute(
"disabled", "true");

newDiagramC5.appendChild(newDiagra
mCancelButton);

newDiagramC6.setAttribute("colspan",
"3");


newDiagramRow.appendChild(newDiag
ramC1);

newDiagramRow.appendChild(newDiag
ramC2);

newDiagramRow.appendChild(newDiag
ramC3);

newDiagramRow.appendChild(newDiag
ramC4);

newDiagramRow.appendChild(newDiag
ramC5);

newDiagramRow.appendChild(newDiag
ramC6);

dltbody.appendChild(newDiagramRow);

   // hidden row for copying diagram
   var copyDiagramRow =
document.createElement("tr");
   copyDiagramRow.setAttribute("id",
"copyDiagramRow");
   copyDiagramRow.setAttribute("style",
"display: none;");
   var copyDiagramC1 =
document.createElement("td");
   var copyDiagramC2 =
document.createElement("td");
   var copyDiagramC3 =
document.createElement("td");
   var copyDiagramC4 =
document.createElement("td");
   var copyDiagramC5 =
document.createElement("td");
   var copyDiagramC6 =
document.createElement("td");

   var copyDiagramTitleInput =
document.createElement("input");

copyDiagramTitleInput.setAttribute("id"
, "copyTitle");

copyDiagramTitleInput.setAttribute("typ
e", "text");

copyDiagramTitleInput.setAttribute("val
ue", "");

copyDiagramTitleInput.setAttribute("siz
e", "20");

copyDiagramTitleInput.setAttribute("dis
abled", "true");

copyDiagramC1.appendChild(copyDiagr
amTitleInput);
```

```
   var copyDiagramDescriptionInput =
document.createElement("textarea");

copyDiagramDescriptionInput.setAttrib
ute("id", "copyDescription");

copyDiagramDescriptionInput.setAttrib
ute("rows", "1");

copyDiagramDescriptionInput.setAttrib
ute("cols", "20");

copyDiagramDescriptionInput.setAttrib
ute("disabled", "true");

copyDiagramC2.appendChild(copyDiagr
amDescriptionInput);


copyDiagramC3.appendChild(document.
createTextNode(USERNAME));

   var copyDiagramButton =
document.createElement("input");
   copyDiagramButton.setAttribute("id",
"copyButton");

copyDiagramButton.setAttribute("type",
"button");

copyDiagramButton.setAttribute("value
", "OK");

copyDiagramButton.setAttribute("onclic
k", "javascript:copyDiagram()");

copyDiagramButton.setAttribute("disabl
ed", "true");

copyDiagramC4.appendChild(copyDiagr
amButton);

   var copyDiagramCancelButton =
document.createElement("input");

copyDiagramCancelButton.setAttribute(
"id", "cancelCopyButton");

copyDiagramCancelButton.setAttribute(
"type", "button");

copyDiagramCancelButton.setAttribute(
"value", "Cancel");

copyDiagramCancelButton.setAttribute(
"onclick",
"javascript:cancelCopyDiagram()");

copyDiagramCancelButton.setAttribute(
"disabled", "true");

copyDiagramC5.appendChild(copyDiagr
amCancelButton);

copyDiagramC6.setAttribute("colspan",
"3");


copyDiagramRow.appendChild(copyDia
gramC1);

copyDiagramRow.appendChild(copyDia
gramC2);

copyDiagramRow.appendChild(copyDia
gramC3);

copyDiagramRow.appendChild(copyDia
gramC4);

copyDiagramRow.appendChild(copyDia
gramC5);

copyDiagramRow.appendChild(copyDia
gramC6);

dltbody.appendChild(copyDiagramRow)
;

   dltable.replaceChild(dltbody,
oldtbody);
}

function newDiagram() {

document.getElementById("newDiagra
mRow").setAttribute("style", "display:
row;");

document.getElementById("newDiagra
m").disabled = true;

document.getElementById("addTitle").d
isabled = false;

document.getElementById("addTitle").f
ocus();

document.getElementById("addDescript
ion").disabled = false;

document.getElementById("addButton"
).disabled = false;

document.getElementById("cancelAddB
utton").disabled = false;
}

function addNew() {
   var xhReq = new
createXMLHttpRequest();
   xhReq.open("POST", 'DiagramList',
false);
   xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
   xhReq.send("action=add&title=" +
encodeURIComponent(document.getEle
mentById("addTitle").value)
      + "&description=" +
encodeURIComponent(document.getEle
mentById("addDescription").value));

   try {
      dlist =
JSON.parse(xhReq.responseText);
   } catch(e) {
      alert("Error getting diagram list.");
   }

   showList();
   cancelAddDiagram();
}

function cancelAddDiagram() {

document.getElementById("newDiagra
mRow").setAttribute("style", "display:
none;");

document.getElementById("addTitle").d
isabled = true;

document.getElementById("addTitle").v
alue = 'New Diagram';

document.getElementById("addDescript
ion").value = "";

document.getElementById("addDescript
ion").disabled = true;

document.getElementById("addButton"
).disabled = true;

document.getElementById("cancelAddB
utton").disabled = true;

document.getElementById("newDiagra
m").disabled = false;
}

function deleteDiagram(i) {
   var confirmDelete =
window.confirm("Are you sure you
want to delete " + dlist[i][1] + "?");
   if(!confirmDelete) return;

   var xhReq = new
createXMLHttpRequest();
   xhReq.open("POST", 'DiagramList',
false);
   xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
   xhReq.send("action=delete&id=" +
dlist[i][0]);

   try {
      dlist =
JSON.parse(xhReq.responseText);
   } catch(e) {
      alert("Error getting diagram list.");
   }
```

```
   showList();
}

function leaveDiagram(i) {
   var confirmLeave =
window.confirm("You will no longer be
able to edit this diagram."
      + " Do you wish to continue?");
   if(!confirmLeave) return;

   var xhReq = new
createXMLHttpRequest();
   xhReq.open("POST", 'DiagramList',
false);
   xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
   xhReq.send("action=leave&id=" +
dlist[i][0]);

   try {
      dlist =
JSON.parse(xhReq.responseText);
   } catch(e) {
      alert("Error getting diagram list.");
   }

   showList();
}

function openDiagram(i) {

window.open("editor.jsp?diagram_id="
+ i);
}

function enableCopyDiagram(id, title) {
   selectedId = id;

document.getElementById("copyDiagra
mRow").setAttribute("style", "display:
row;");

document.getElementById("copyTitle").
disabled = false;

document.getElementById("copyDescri
ption").disabled = false;

document.getElementById("copyButton
").disabled = false;

document.getElementById("cancelCopy
Button").disabled = false;

document.getElementById("copyTitle").
value = "Copy of " + title;

document.getElementById("copyDescri
ption").value = "";

document.getElementById("copyTitle").
focus();
}

function copyDiagram() {
   if(selectedId == null) return;

   var xhReq = new
createXMLHttpRequest();
   xhReq.open("POST", 'DiagramList',
false);
   xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
   xhReq.send("action=copy&from=" +
selectedId + "&title=" +
encodeURIComponent(document.getEle
mentById("copyTitle").value)
      + "&description=" +
encodeURIComponent(document.getEle
mentById("copyDescription").value));

   try {
      dlist =
JSON.parse(xhReq.responseText);
   } catch(e) {
      alert("Error getting diagram list.");
   }

   showList();
   cancelCopyDiagram();
}

function cancelCopyDiagram() {

document.getElementById("copyDiagra
mRow").setAttribute("style", "display:
none;");

document.getElementById("copyTitle").
disabled = true;

document.getElementById("copyDescri
ption").disabled = true;

document.getElementById("copyButton
").disabled = true;

document.getElementById("cancelCopy
Button").disabled = true;

document.getElementById("copyTitle").
value = "";

document.getElementById("copyDescri
ption").value = "";
   selectedId = null;
}

function
enableAddReqCollaborator(rowNo) {
   var addCollaboratorSection =
document.getElementById("addCollabor
atorSection" + rowNo);

addCollaboratorSection.setAttribute("st
yle", "display: inline;");

addCollaboratorSection.getElementsByT
agName('input')[0].focus();
}

function cancelAddCollaborator(rowNo)
{
   var addCollaboratorSection =
document.getElementById("addCollabor
atorSection" + rowNo);

addCollaboratorSection.setAttribute("st
yle", "display: none;");

addCollaboratorSection.getElementsByT
agName('input')[0].value = '';
}

function addCollaboratorReq(id, rowNo)
{
   var addCollaboratorSection =
document.getElementById("addCollabor
atorSection" + rowNo);
   var collaborator =
addCollaboratorSection.getElementsByT
agName('input')[0].value;
   //alert("ID: " + id);
   var xhReq = new
createXMLHttpRequest();
   xhReq.onreadystatechange =
addCollaboratorReqResult;
   xhReq.open("POST", URL, true);
   xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");

xhReq.send("action=addcollaborator&id
=" + id + "&collaborator="
      +
encodeURIComponent(collaborator));

   //getDiagramList();
   cancelAddCollaborator(rowNo);
}

function addCollaboratorReqResult() {
   if(this.readyState == 4 && this.status
== 200) {
      var result =
JSON.parse(this.responseText);
      if(result.error)
         alert(result.error);
      else if(result.response) {
         getDiagramList();
         //alert(result.response);
      }
   }
}

function removeCollaboratorReq(id,
toRemove) {
   if(!window.confirm("Remove " +
toRemove + " from list of
collaborators?"))
      return;

   var xhReq = new
createXMLHttpRequest();
   xhReq.onreadystatechange =
removeCollaboratorReqResult;
```

```
   xhReq.open("POST", URL, true);
   xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");

xhReq.send("action=removecollaborato
r&id=" + id + "&collaborator="
      +
encodeURIComponent(toRemove));
}

function
removeCollaboratorReqResult() {
   if(this.readyState == 4 && this.status
== 200) {
      var result =
JSON.parse(this.responseText);
      if(result.error)
         alert(result.error);
      else if(result.response) {
         getDiagramList();
         //alert(result.response);
      }
   }
}
```

---

editor.jsp

```
<%@page
import="org.apache.commons.lang3.Str
ingEscapeUtils"%>
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
   <head>
      <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
      <title>Cordie Editor</title>
      <link rel="shortcut icon"
type="image/x-icon"
href="../images/icon.gif">
      <link rel="stylesheet"
type="text/css"
href="../css/cordie.css"/>
      <link rel="stylesheet"
type="text/css" href="../css/cordie-
editor.css"/>
      <script type="text/javascript"
src="editor/javascript/defaults.js"></scr
ipt>
      <script type="text/javascript"
src="editor/javascript/editor.js"></scrip
t>
      <script type="text/javascript"
src="editor/javascript/diagram shapes
1.js"></script>
      <script type="text/javascript"
src="editor/javascript/diagram shapes
2.js"></script>
      <script type="text/javascript"
src="editor/javascript/diagram shapes
3.js"></script>
      <script type="text/javascript"
src="editor/javascript/diagram shapes
4.js"></script>
      <script type="text/javascript"
src="editor/javascript/communication.js
"></script>
      <script type="text/javascript"
src="editor/javascript/canvas.js"></scri
pt>
      <script type="text/javascript"
src="editor/javascript/tools.js"></script
>
      <script type="text/javascript"
src="editor/javascript/tools
2.js"></script>
      <script type="text/javascript"
src="editor/javascript/operations.js"></
script>
      <script type="text/javascript"
src="editor/javascript/objects.js"></scri
pt>
      <script type="text/javascript"
src="editor/javascript/properties box
1.js"></script>
      <script type="text/javascript"
src="editor/javascript/properties box
2.js"></script>
      <script type="text/javascript"
src="editor/javascript/collaborators.js">
</script>
      <script type="text/javascript"
src="editor/javascript/developer.js"></s
cript>
      <script type="text/javascript"
src="editor/javascript/jsBezier-0.3-
min.js"></script>
      <script type="text/javascript"
src="editor/javascript/jscolor/jscolor.js"
></script>
   </head>

   <body onload="javascript:init('<%=
StringEscapeUtils.escapeEcmaScript((Str
ing)

session.getAttribute("USERNAME"))
%>', '<%=
request.getParameter("diagram_id")
%>')">
      <div id="main-header">
         <img src="../images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
         <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
      </div>
      <div id="menu">
         <ul id="menulist">
            <li class="home"><a
href="../index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
            <li><a href="../login.jsp">Log
In</a></li>
            <li><a
href="../signup.jsp">Register</a></li>
            <li id="empty"></li>
<%  } else { %>
            <li><a
href="profile.jsp">Profile</a></li>
            <li><a
href="diagrams.jsp">Diagrams</a></li>
            <li><a href="../logout.jsp">Log
Out</a></li>
            <li id="empty2"></li>
<%  } %>
         </ul>
      </div>
      <div id="main-content">
         <br/>
         <div id="header">
         </div>

         <div id="content">
            <div id="canvasArea">
               <canvas id="drawingArea"
width="800px"
height="500px"></canvas>
               <div id="textAreaPopUp">
                  <textarea
id="textValueInput"></textarea>
                  <input type="button"
value="save" id="saveText"
onclick="saveText();">
               </div>
            </div>

            <div id="rightBoxes">
               <div id="rightBox1">
                  <jsp:include
page="editor/toolbox.jsp"/>
               </div>

               <div id="rightBox2">
                  <div id="boxMenu">
                     <ul>
                        <li><a
id="propertiesTabButton"
href="javascript:propertiesBoxTab()">Se
lected Object</a></li>
                        <li><a
id="collaboratorsTabButton"
href="javascript:collaboratorsTab()">Col
laborators</a></li>
                        <li><a
id="recentChangesTabButton"
href="javascript:recentChangesTab()">R
ecent Changes</a></li>
                     </ul>
                  </div>

                  <div id="boxTab">
```

```
            <jsp:include
page="editor/properties box.jsp" />

            <jsp:include
page="editor/developer.jsp" />

            <jsp:include
page="editor/collaborators.jsp" />
          </div>
        </div>
      </div>
    </div>

    <div id="error" style="display:
none;">
      <span style="color: #FF6600;
font-weight: bold;">Error </span>
      <span
id="errorMessage"></span>
    </div>

    <div id="footer">
    </div>

    <iframe id="downloadFrame"
style="display:none;"></iframe>

      <br/>
      <br/>

      <div id="main-footer">
        All Rights Reserved. Copyright
© 2012 University of the Philippines
Manila
      </div>
  </div>
 </body>
</html>
```

---

editpassword.jsp

```
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<%@page import="java.sql.*"%>
<%@page import="java.util.*"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
  <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
    <title>Edit Password</title>
    <link rel="shortcut icon"
type="image/x-icon"
href="../images/icon.gif">
    <link rel="stylesheet"
type="text/css"
href="../css/cordie.css"/>
  </head>
  <body>
   <div id="main-header">
      <img src="../images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
      <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
    </div>
    <div id="menu">
      <ul id="menulist">
        <li class="home"><a
href="../index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
        <li><a href="../login.jsp">Log
In</a></li>
        <li><a
href="../signup.jsp">Register</a></li>
        <li id="empty"></li>
<%  } else { %>
        <li><a
href="profile.jsp">Profile</a></li>
        <li><a
href="diagrams.jsp">Diagrams</a></li>
        <li><a href="../logout.jsp">Log
Out</a></li>
        <li id="empty2"></li>
<%  } %>
      </ul>
    </div>
    <div id="main-content">
     <br/><br/>
<%
  HashMap<String, String> errors = new
HashMap<String, String>();;
  boolean valid = true;
  String password = "";
  String newpassword1 = "";
  String newpassword2 = "";


if(request.getMethod().equals("POST"))
{
    try {
      password =
request.getParameter("password");
      newpassword1 =
request.getParameter("newpassword1"
);
      newpassword2 =
request.getParameter("newpassword2"
);


Class.forName("com.mysql.jdbc.Driver")
;
      Connection connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");
      //Statement statement =
connection.createStatement();
      String sql = "SELECT COUNT(*)
FROM user WHERE username = ? AND
password = md5(?)";
      PreparedStatement stmt =
connection.prepareStatement(sql);
      stmt.setString(1, (String)
session.getAttribute("USERNAME"));
      stmt.setString(2, password);
      ResultSet rs =
stmt.executeQuery();
      /*ResultSet rs =
statement.executeQuery("SELECT
COUNT(*) "
          + "FROM user WHERE
username = '" +
session.getAttribute("USERNAME")
          + "' AND password = md5('" +
password + "')");*/
      rs.next();
      if(rs.getInt(1) == 0) {
        valid = false;
        errors.put("password", "Wrong
Password");
      }
      if(newpassword1.equals("") ) {
        valid = false;
        errors.put("newpassword1",
"Please enter a valid password");
      }
      if(!newpassword1.equals("") &&
(newpassword2.equals("") ||

!newpassword1.equals(newpassword2))
) {
        newpassword2 = "";
        valid = false;
        errors.put("newpassword2",
"Please make sure your passwords
match.");
      }

      if(valid) {
        sql = "UPDATE user SET
password = md5(?) WHERE username =
?";
        stmt =
connection.prepareStatement(sql);
        stmt.setString(1,
newpassword1);
        stmt.setString(2, (String)
session.getAttribute("USERNAME"));
        stmt.executeUpdate();
        //int i =
statement.executeUpdate("UPDATE
user SET password = md5('" +
        //    newpassword1 + "')" + "
WHERE username='" +
session.getAttribute("USERNAME") +
"'");
%>
    <div style="text-align:
center;">Your password has been
saved. Go back to <a
href="profile.jsp">profile</a></div>
```

```
      <br/>
      <br/>

      <div id="main-footer">
        All Rights Reserved. Copyright ©
2012 University of the Philippines
Manila
      </div>
    </div>
  </body>
</html>
<%          return;
        }

    } catch (ClassNotFoundException e) {
        System.err.println("Driver Error");
    } catch (SQLException e) {
        System.err.println("SQLException:
" + e.getMessage());
    }
  }
%>
      <form name="editPasswordForm"
action="editpassword.jsp"
method="POST">
        <table align="center" border="0"
cellpadding="0">
          <tbody>
            <tr>
              <td>Enter Your Password:
</td>
              <td><input id="password"
type="password" name="password"
value="" size="25" />
              <br/><font
color="red"><%=
(errors.get("password") == null) ? "" :
errors.get("password") %></font></td>
            </tr>
            <tr>
              <td>Your New Password:
</td>
              <td><input
id="newpassword1" type="password"
name="newpassword1" value=""
size="25" />
              <br/><font
color="red"><%=
(errors.get("newpassword1") == null) ?
"" : errors.get("newpassword1")
%></font></td>
            </tr>
            <tr>
              <td>Confirm New
Password: </td>
              <td><input
id="newpassword2" type="password"
name="newpassword2" value=""
size="25" />
              <br/><font color="red"
color="red"><%=
(errors.get("newpassword2") == null) ?
"" : errors.get("newpassword2") %>
              </font></td>
            </tr>
            <tr><td colspan="2"><input
type="submit" value="Change
Password" /></td></tr>
          </tbody>
        </table>
      </form>
      <% if (!valid) { %>
      <script type="text/javascript">

(document.getElementById("password")
).value = '<%= password %>';

(document.getElementById("newpassw
ord1")).value = '<%= newpassword1
%>';

(document.getElementById("newpassw
ord2")).value = '<%= newpassword2
%>';
      </script>
      <% } %>

      <br/>
      <br/>

      <div id="main-footer">
        All Rights Reserved. Copyright ©
2012 University of the Philippines
Manila
      </div>
    </div>
  </body>
</html>
```

---

editprofile.jsp

```
<%@page
import="org.apache.commons.lang3.Str
ingEscapeUtils"%>
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<%@page import="java.io.*"%>
<%@page import="java.util.*"%>
<%@page import="javax.servlet.*"%>
<%@page
import="javax.servlet.http.*"%>
<%@page
import="org.apache.commons.fileuploa
d.*"%>
<%@page
import="org.apache.commons.fileuploa
d.disk.*"%>
<%@page
import="org.apache.commons.fileuploa
d.servlet.*"%>
<%@page import="java.sql.*"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
  <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
    <title>Edit Profile</title>
    <link rel="shortcut icon"
type="image/x-icon"
href="../images/icon.gif">
    <link rel="stylesheet"
type="text/css"
href="../css/cordie.css"/>
  </head>
  <body>
   <div id="main-header">
     <img src="../images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
     <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
   </div>
   <div id="menu">
     <ul id="menulist">
       <li class="home"><a
href="../index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
       <li><a href="../login.jsp">Log
In</a></li>
       <li><a
href="../signup.jsp">Register</a></li>
       <li id="empty"></li>
<%  } else { %>
       <li><a
href="profile.jsp">Profile</a></li>
       <li><a
href="diagrams.jsp">Diagrams</a></li>
       <li><a href="../logout.jsp">Log
Out</a></li>
       <li id="empty2"></li>
<%  } %>
     </ul>
   </div>
   <div id="main-content">
    <br/>
<%
if(request.getMethod().equals("POST"))
{
    String firstname = "";
    String lastname = "";
    String email = "";

    try {

Class.forName("com.mysql.jdbc.Driver")
;
      Connection connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");
      //Statement statement =
connection.createStatement();
```

```
        File file ;
        int maxFileSize = 5000 * 1024;
        int maxMemSize = 5000 * 1024;
        String filePath =
"uploaded_images/";
        FileItem uploadedImage = null;
        String extension = "";

        // Verify the content type
        String contentType =
request.getContentType();

if((contentType.indexOf("multipart/for
m-data") >= 0)) {
           DiskFileItemFactory factory =
new DiskFileItemFactory();

factory.setSizeThreshold(maxMemSize);
           factory.setRepository(new
File(""));

           ServletFileUpload upload = new
ServletFileUpload(factory);
           upload.setSizeMax( maxFileSize
);
           try {
              // Parse the request to get
file items.
              List fileItems =
upload.parseRequest(request);
              // Process the uploaded file
items
              Iterator i =
fileItems.iterator();

              while (i.hasNext()) {
                 FileItem item = (FileItem)
i.next();

                 if(item.isFormField()) {

if(item.getFieldName().equals("firstnam
e")) {
                       firstname =
item.getString();
                    } else
if(item.getFieldName().equals("lastnam
e")) {
                       lastname =
item.getString();
                    } else
if(item.getFieldName().equals("email"))
{
                       email =
item.getString();
                    }
                 } else {
                    String fileName =
item.getName();

                    if(!fileName.equals("")) {

if(fileName.lastIndexOf(".") >= 0) {
                          extension =
fileName.substring(fileName.lastIndexO
f("."));
                       }

                       uploadedImage = item;
                    }
                 }
              } // end while
           } catch(Exception ex) {
              System.out.println(ex);
           }
        }

        if(uploadedImage != null) {
           String sql = "UPDATE user SET
firstname = ?, lastname = ?, email = ?"
              + ", displaypic = ? WHERE
username = ?";
           PreparedStatement stmt =
connection.prepareStatement(sql);
           stmt.setString(1, firstname);
           stmt.setString(2, lastname);
           stmt.setString(3, email);
           stmt.setString(4,
session.getAttribute("USERNAME") +
extension);
           stmt.setString(5, (String)
session.getAttribute("USERNAME"));
           stmt.executeUpdate();

           File oldDP = new File(filePath +
session.getAttribute("DISPLAYPIC"));
           oldDP.delete();
           uploadedImage.write(new
File(filePath +
session.getAttribute("USERNAME") +
extension)) ;

session.setAttribute("DISPLAYPIC",
session.getAttribute("USERNAME") +
extension);
        } else {
           String sql = "UPDATE user SET
firstname = ?, lastname = ?, email = ?"
              + " WHERE username = ?";
           PreparedStatement stmt =
connection.prepareStatement(sql);
           stmt.setString(1, firstname);
           stmt.setString(2, lastname);
           stmt.setString(3, email);
           stmt.setString(4, (String)
session.getAttribute("USERNAME"));
           stmt.executeUpdate();
        }


session.setAttribute("FIRSTNAME",
firstname);
        session.setAttribute("LASTNAME",
lastname);
        session.setAttribute("EMAIL",
email);
%>
     <div style="text-align: center;">
       Your changes were saved. <a
href="profile.jsp">View your profile</a>
     </div>
     <br/>
     <br/>

     <div id="main-footer">
       All Rights Reserved. Copyright ©
2012 University of the Philippines
Manila
     </div>
    </div>
   </body>
</html>
<%       return;
        } catch (ClassNotFoundException
e) {
           System.err.println("Driver
Error");
        } catch (SQLException e) {

System.err.println("SQLException: " +
e.getMessage());
        }
  } else { %>
     <form name="editProfileForm"
action="editprofile.jsp" method="POST"
enctype="multipart/form-data">
        <table align="center" border="0"
cellpadding="0">
           <tbody>
              <tr>
                 <td>Current Display
Picture</td>
                 <td>
                    <div
class="displayPicContainer">
                       <img
class="displayPic"
src="DisplayPicture?imgTitle=<%=
session.getAttribute("DISPLAYPIC") %>"
                          alt="display
picture"/>
                    </div>
                 </td>
              </tr>
              <tr>
                 <td>Change</td>
                 <td><input type="file"
name="displaypic" value=""
size="25"/></td>
              </tr>
              <tr>
                 <td>First Name</td>
                 <td><input type="text"
id="firstname" name="firstname"
value="" size="25" maxlength="25"
/></td>
              </tr>
              <tr>
                 <td>Last Name</td>
                 <td><input type="text"
id="lastname" name="lastname"
value="" size="25" maxlength="25"
/></td>
              </tr>
              <tr>
                 <td>Email</td>
```

```
                <td><input type="text"
id="email" name="email" value=""
size="25" maxlength="50" /></td>
              </tr>
              <tr>
                <td><input type="submit"
value="Save" name="submit" /></td>
                <td><input type="reset"
value="Reset" name="reset" /></td>
              </tr>
            </tbody>
          </table>
      </form>
      <script type="text/javascript">

(document.getElementById("firstname")
).value = '<%=
StringEscapeUtils.escapeEcmaScript((Str
ing) session.getAttribute("FIRSTNAME"))
%>';

(document.getElementById("lastname")
).value = '<%=
StringEscapeUtils.escapeEcmaScript((Str
ing) session.getAttribute("LASTNAME"))
%>';

(document.getElementById("email")).va
lue = '<%=
StringEscapeUtils.escapeEcmaScript((Str
ing) session.getAttribute("EMAIL")) %>';
      </script>
<% } %>

      <br/>
      <br/>

      <div id="main-footer">
        All Rights Reserved. Copyright ©
2012 University of the Philippines
Manila
      </div>
    </div>
  </body>
</html>
```

---

profile.jsp

```
<%@page contentType="text/html"
pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dt
d">

<html>
  <head>
    <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
    <title>Profile</title>
    <link rel="shortcut icon"
type="image/x-icon"
href="../images/icon.gif">
    <link rel="stylesheet"
type="text/css"
href="../css/cordie.css"/>
  </head>
  <body>
    <div id="main-header">
      <img src="../images/cordie
banner.png">
<%  if(session != null &&
session.getAttribute("USERNAME") !=
null) { %>
      <div id="usernameDisplay">You
are logged in <%=
session.getAttribute("FIRSTNAME")
%>.</div>
<%  } %>
    </div>
    <div id="menu">
      <ul id="menulist">
        <li class="home"><a
href="../index.jsp">Home</a></li>
<%  if(session == null ||
session.getAttribute("USERNAME") ==
null) { %>
        <li><a href="../login.jsp">Log
In</a></li>
        <li><a
href="../signup.jsp">Register</a></li>
        <li id="empty"></li>
<%  } else { %>
        <li><a
href="profile.jsp">Profile</a></li>
        <li><a
href="diagrams.jsp">Diagrams</a></li>
        <li><a href="../logout.jsp">Log
Out</a></li>
        <li id="empty2"></li>
<%  } %>
      </ul>
    </div>
    <div id="main-content">
      <br/>
      <table width="300" align="center"
border="0" cellpadding="2">
        <thead>
          <tr>
            <th colspan="2">
              <div
class="displayPicContainer">
                <img class="displayPic"
src="DisplayPicture?imgTitle=<%=
session.getAttribute("DISPLAYPIC") %>"
                  alt="display
picture"/>
              </div>
            </th>
            <th></th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>Username: </td>
            <td><%=
session.getAttribute("USERNAME")
%></td>
          </tr>
          <tr>
            <td>First Name: </td>
            <td><%=
session.getAttribute("FIRSTNAME")
%></td>
          </tr>
          <tr>
            <td>Last Name: </td>
            <td><%=
session.getAttribute("LASTNAME")
%></td>
          </tr>
          <tr>
            <td>Email: </td>
            <td><%=
session.getAttribute("EMAIL") %></td>
          </tr>
          <tr>
            <td> </td>
          </tr>
          <tr>
            <td colspan="2"><a
href="editprofile.jsp">Edit Profile
Information</a></td>
          </tr>
          <tr>
            <td colspan="2"><a
href="editpassword.jsp">Edit Your
Password</a></td>
          </tr>
        </tbody>
      </table>

      <br/>
      <br/>

      <div id="main-footer">
        All Rights Reserved. Copyright ©
2012 University of the Philippines
Manila
      </div>
    </div>


  </body>
</html>
```

---

basic buttons.jsp

```
          <input type="image"
src="../images/saveas.png" title="Save
as Image"
              id="saveAsButton"
onclick="javascript:saveAsImage()"/>
          <input type="image"
src="../images/delete.png" title="Delete
Object"
              id="deleteButton"
onclick="javascript:deleteObj()"/>
          <!--<input type="checkbox"
name="realtimeUpdates" value="ON"

id="realtimeUpdatesCheckbox"
onclick="javascript:realtimeToggle()"/>R
eal-time
```

```
            <input type="button"
value="Update"
onclick="javascript:requestForUpdates()
"
                   id="updateButton"/>-->
```

---

collaborators.jsp

```
<div
id="collaboratorsAndCurrentUsers">
   <div
id="collaboratorsAndCurrentUsersBoxT
abs">
      <ul>
         <!--<li id="currentusersSubTab">
            <a id="currentusersSubTab2"
href="javascript:currentusersSubTab()">
Current Users</a>
         </li>-->
         <li id="collaboratorsSubTab">
            <a id="collaboratorsSubTab2"
href="javascript:collaboratorsSubTab()"
>Collaborators</a>
         </li>
         <li id="addCollaboratorSubTab">
            <a
id="addCollaboratorSubTab2"
href="javascript:addCollaboratorSubTab
()">Add Collaborator</a>
         </li>
      </ul>
   </div>

   <div
id="collaboratorsAndCurrentUsersTable
">
      <!--<div id="currentusersSection"
style="display: none;">
         <table border="0"
cellspacing="5" cellpadding="0"
id="currentusers">
            <thead>
               <tr>
                  <th colspan="2">Users
currently editing this diagram:</th>
               </tr>
            </thead>
         </table>
      </div>-->
      <div id="collaboratorsSection"
style="display: none;">
         <table border="0"
cellspacing="0" cellpadding="0"
id="collaborators">
            <thead>
               <tr>
                  <th
colspan="2">Collaborators are the users
who can edit this diagram.</th>
               </tr>
            </thead>
         </table>
      </div>
      <table id="addCollaboratorSection"
border="0" cellspacing="0"
cellpadding="0" style="display: none;">
            <thead>
               <tr>
                  <th colspan="2">You can add
a collaborator by entering their
username below.</th>
               </tr>
            </thead>
            <tbody>
               <tr>
                  <td colspan="2">
                     <input
id="enableAddCollaboratorButton"
type="button" value="Add
Collaborator"

onclick="javascript:enableAddReqCollab
orator()" />
                  </td>
               </tr>
               <tr>
                  <td>Username: </td>
                  <td><input
id="addCollaboratorText" type="text"
value="" size="20" disabled="disabled"
/></td>
               </tr>
               <tr>
                  <td>
                     <input
id="addCollaboratorButton"
type="button" value="Add"
                    disabled="disabled"
onclick="javascript:addCollaboratorReq(
)" />
                  </td>
                  <td>
                     <input
id="cancelAddCollaborator"
type="button" value="Cancel"
disabled="disabled"

onclick="javascript:cancelAddCollaborat
or()" />
                  </td>
               </tr>
            </tbody>
         </table>
      </div>
</div>
```

---

properties box.jsp

```
<%
   String objects[] = {"rectangle",
"ellipse", "text", "line", "path", "class",
                  "note", "bezier",
"polygon", "polyline", "instance",
"node",
                  "statebox",
"expansionregion", "arrowhead",
"activationbar",
                  "lifeline", "deletion",
"initialps", "finalstate", "shallowhistps",
                  "deephistps",
"initialnode", "finalnode", "blackbar",
"diamond",
                  "pin", "flowfinal", "port",
"frame", "package1", "package2",
                  "component", "artifact",
"usecase", "actor", "superstate",
                  "action", "subactivity",
"timesignal", "acceptsignal",
                  "sendsignal", "connector",
"transformation", "part"};
   String objecttype[] = {"Rectangle",
"Ellipse", "Text", "Line", "Path", "Class",
                  "Note", "Bezier",
"Polygon", "Polyline", "Instance
Specification",
                  "Node", "Statebox",
"Expansion Region", "Arrowhead",
"Activation Bar",
                  "Lifeline", "Deletion",
"Initial Pseudostate", "Final State",
                  "Shallow History
Pseudostate", "Deep History
Pseudostate",
                  "Initial Node", "Final
Node", "Black Bar", "Diamond", "Pin",
                  "Flow Final", "Port",
"Frame", "Small Package", "Large
Package",
                  "Component", "Artifact",
"Use Case", "Actor", "Superstate",
                  "Action", "Subactivity",
"Time Signal", "Accept Signal",
                  "Send Signal",
"Connector", "Transformation", "Part"};
   String styles[][] = { {"x", "y", "width",
"height", "linewidth", "linecap",
                        "linejoin",
"strokestyle", "fillcolor", "rotate",
                        "z-order"},
                       {"x", "y", "width",
"height", "linewidth", "linecap",
                        "linejoin",
"strokestyle", "fillcolor", "rotate",
                        "z-order"},
                       {"label", "x", "y", "width",
"height", "textcolor",
                        "fontstyle",
"fontweight", "fontsize", "fontfamily",
                        "rotate", "z-order"},
                       {"arrowstyle1",
"arrowstyle2", "linestyle", "x1",
                        "y1", "x2", "y2",
"linewidth", "linecap", "linejoin",
                        "strokestyle", "z-
order"},
                       {"strokestyle",
"linewidth", "linecap", "linejoin",
                        "z-order"},
                       {"fontsize", "x", "y",
"width", "height", "linewidth",
                        "linecap", "linejoin",
"strokestyle", "fillcolor",
                        "classfontsize",
"textcolor", "rotate", "z-order"},
```

{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"arrowstyle1",
"arrowstyle2", "linestyle", "x1",
"y1", "x2", "y2",
"ctrl1x", "ctrl1y", "ctrl2x",
"ctrl2y", "linewidth",
"linecap", "linejoin",
"strokestyle", "z-
order"},
{"strokestyle",
"linewidth", "linecap", "linejoin",
"fillcolor", "z-order"},
{"arrowstyle1",
"arrowstyle2", "linestyle", "linewidth",
"linecap", "linejoin",
"strokestyle", "z-order"},
{"fontsize", "x", "y",
"width", "height", "linewidth",
"linecap", "linejoin",
"strokestyle", "fillcolor",
"textcolor", "rotate",
"z-order"},
{"fontsize", "x", "y",
"width", "height", "topheight",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "textcolor",
"rotate", "z-order"},
{"fontsize", "x", "y",
"width", "height", "linewidth",
"linecap", "linejoin",
"strokestyle", "fillcolor",
"textcolor", "rotate",
"z-order"},
{"x", "y", "listboxpin1x",
"listboxpin1y", "listboxpin2x",
"listboxpin2y",
"listboxpinsize", "width", "height",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "linewidth", "linecap",
"linejoin",
"strokestyle", "fillcolor", "rotate",
"z-order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",
"fontsize",
"fontfamily", "label", "textcolor",
"linewidth", "linecap",
"linejoin", "strokestyle",
"fillcolor", "rotate", "z-
order"},
{"x", "y", "width",
"height", "fontstyle", "fontweight",

```
                    "fontsize",
"fontfamily", "label", "textcolor",
                    "linewidth", "linecap",
"linejoin", "strokestyle",
                    "fillcolor", "rotate", "z-
order"},
                {"x", "y", "width",
"height", "fontstyle", "fontweight",
                    "fontsize",
"fontfamily", "label", "textcolor",
                    "linewidth", "linecap",
"linejoin", "strokestyle",
                    "fillcolor", "rotate", "z-
order"},
                {"x", "y", "width",
"height", "fontstyle", "fontweight",
                    "fontsize",
"fontfamily", "label", "textcolor",
                    "linewidth", "linecap",
"linejoin", "strokestyle",
                    "fillcolor", "rotate", "z-
order"},
                {"x", "y", "width",
"height", "fontstyle", "fontweight",
                    "fontsize",
"fontfamily", "label", "textcolor",
                    "linewidth", "linecap",
"linejoin", "strokestyle",
                    "fillcolor", "rotate", "z-
order"},
                {"x", "y", "width",
"height", "fontstyle", "fontweight",
                    "fontsize",
"fontfamily", "label", "textcolor",
                    "linewidth", "linecap",
"linejoin", "strokestyle",
                    "fillcolor", "rotate", "z-
order"},
                {"x", "y", "width",
"height", "fontstyle", "fontweight",
                    "fontsize",
"fontfamily", "label", "textcolor",
                    "linewidth", "linecap",
"linejoin", "strokestyle",
                    "fillcolor", "rotate", "z-
order"},
                {"x", "y", "width",
"height", "fontstyle", "fontweight",
                    "fontsize",
"fontfamily", "label", "textcolor",
                    "linewidth", "linecap",
"linejoin", "strokestyle",
                    "fillcolor", "rotate", "z-
order"}
            };
   String properties[][] = { {}, {}, {}, {}, {},
                    {"classname",
"abstractclass", "templateclass",
                         "activeclass",
"qualifiedassociation",
                         "qualifier",
"stereotype", "showattributes",

"showoperations"}, {}, {}, {}, {},
                    {"instancename",
"classname", "showattributes"},
                    {"nodename",
"stereotype", "showartifacts"},
                    {"statename",
"showinternalactivities"}, {}, {},
                    {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {},
                    {}, {}, {}, {}, {}, {}, {}, {},
{}, {}, {}, {},
                    {}, {}, {}, {}, {}, {}
                };
%>

<div id="propertiesBox">
   <div id="propertiesBoxTabs">
      <ul>
         <li id="styleTab" style="display:
none;"><a id="styleTab2"
href="javascript:styleTab()">Style</a></
li>
         <li id="propertiesTab"
style="display: none;"><a
id="propertiesTab2"
href="javascript:propertiesTab()">Prope
rties</a></li>
         <li id="attributesTab"
style="display: none;"><a
id="attributesTab2"
href="javascript:attributesTab()">Attrib
utes</a></li>
         <li id="operationsTab"
style="display: none;"><a
id="operationsTab2"
href="javascript:operationsTab()">Oper
ations</a></li>
         <li id="templatesTab"
style="display: none;"><a
id="templatesTab2"
href="javascript:templatesTab()">Templ
ates</a></li>
         <li id="taggedvaluesTab"
style="display: none;"><a
id="taggedvaluesTab2"
href="javascript:taggedvaluesTab()">Ta
gged Values</a></li>
         <li id="artifactsTab"
style="display: none;"><a
id="artifactsTab2"
href="javascript:artifactsTab()">Artifacts
</a></li>
         <li id="internalactivitiesTab"
style="display: none;"><a
id="internalactivitiesTab2"
href="javascript:internalactivitiesTab()"
>Internal Activities</a></li>
      </ul>
   </div>

   <div id="propertiesTable">
<%  for(int i = 0; i < objects.length; i++) {
%>
      <table id="<%= objects[i]
%>StyleTable" style="display: none;">
         <tr><th
colspan="2">Style</th></tr>
         <tr>
            <td>Object Type</td>
            <td><%= objecttype[i] %></td>
         </tr>
         <%  for(String style : styles[i]) {
%>
         <tr>
            <td><%= style %></td>
            <td>
            <% if(style.equals("x") ||
style.equals("y") ||
               style.equals("x1") ||
style.equals("y1") ||
               style.equals("x2") ||
style.equals("y2") ||
               style.equals("ctrl1x") ||
style.equals("ctrl1y") ||
               style.equals("ctrl2x") ||
style.equals("ctrl2y") ||
               style.equals("listboxpin1x")
|| style.equals("listboxpin1y") ||
               style.equals("listboxpin2x")
|| style.equals("listboxpin2y")) { %>
              <input type="text" id="<%=
objects[i] %>_<%= style %>"

onchange="javascript:changeXY('<%=
style %>', value, '<%= objects[i] %>')"/>
           <% } else
if(style.equals("width") ||
style.equals("height") ||

style.equals("linewidth") ||
style.equals("listboxpinsize") ||

style.equals("topheight") ||
style.equals("fontsize") ||

style.equals("classfontsize") ||
style.equals("rotate")) { %>
              <input type="text" id="<%=
objects[i] %>_<%= style %>"

onchange="javascript:changeFloat('<%=
style %>', value, '<%= objects[i] %>')"/>
           <% } else if(style.equals("z-
order")) { %>
              <input type="text" id="<%=
objects[i] %>_<%= style %>" readonly/>
              <input type="image"
src="../images/up.png" title="Move Up"
                class="upZOrderButton"
onclick="javascript:zOrderUp()"/>
              <input type="image"
src="../images/down.png" title="Move
Down"
                class="downZOrderButton"
onclick="javascript:zOrderDown()"/>
           <% } else
if(style.equals("label")) {
                if(objects[i].equals("note")
|| objects[i].equals("text")) { %>
              <textarea id="<%= objects[i]
%>_<%= style %>" rows="5"
                     cols="15"
onchange="javascript:change('<%= style
%>', value)">
              </textarea>
```

```
            <%     } else { %>
              <input id="<%= objects[i]
%>_<%= style %>" type="text"

onchange="javascript:change('<%= style
%>', value)"/>
            <%       }
             } else
if(style.equals("strokestyle") ||
style.equals("fillcolor") ||
style.equals("textcolor")) { %>
              <!--R: <input type="text"
class="rgbaInput" id="<%= objects[i]
%>_<%= style %>_R"

onchange="javascript:changeColor('<%=
objects[i] %>', '<%= style %>')"/>
              G: <input type="text"
class="rgbaInput" id="<%= objects[i]
%>_<%= style %>_G"

onchange="javascript:changeColor('<%=
objects[i] %>', '<%= style %>')"/>
              B: <input type="text"
class="rgbaInput" id="<%= objects[i]
%>_<%= style %>_B"

onchange="javascript:changeColor('<%=
objects[i] %>', '<%= style %>')"/>-->
              <input class="color" value=""
onchange="javascript:changeColor('<%=
objects[i] %>', '<%= style %>')"
                     id="<%= objects[i]
%>_<%= style %>"/>
              a: <input type="text"
class="rgbaInput" id="<%= objects[i]
%>_<%= style %>_A"

onchange="javascript:changeColor('<%=
objects[i] %>', '<%= style %>')"/>
            <% } else
if(style.equals("fontstyle")) { %>
              <select id="<%= objects[i]
%>_<%= style %>"
onchange="javascript:change('<%= style
%>', value)">
                <option
value="normal">Normal</option>
                <option
value="italic">Italic</option>
              </select>
            <% } else
if(style.equals("fontweight")) { %>
              <select id="<%= objects[i]
%>_<%= style %>"
onchange="javascript:change('<%= style
%>', value)">
                <option
value="normal">Normal</option>
                <option
value="bold">Bold</option>
              </select>
            <% } else
if(style.equals("fontfamily")) { %>
              <select id="<%= objects[i]
%>_<%= style %>"
onchange="javascript:change('<%= style
%>', value)">
                <option value="sans-
serif">Sans-serif</option>
              </select>
            <% } else
if(style.equals("linecap")) { %>
              <select id="<%= objects[i]
%>_<%= style %>"
onchange="javascript:change('<%= style
%>', value)">
                <option
value="butt">Butt</option>
                <option
value="round">Round</option>
                <option
value="square">Square</option>
              </select>
            <% } else
if(style.equals("linejoin")) { %>
              <select id="<%= objects[i]
%>_<%= style %>"
onchange="javascript:change('<%= style
%>', value)">
                <option
value="miter">Miter</option>
                <option
value="round">Round</option>
                <option
value="bevel">Bevel</option>
              </select>
            <% } else
if(style.equals("arrowstyle1") ||
style.equals("arrowstyle2")) { %>
              <select id="<%= objects[i]
%>_<%= style %>"
onchange="javascript:change('<%= style
%>', value)">
                <option
value="none">None</option>
                <option
value="lines">Lines</option>
                <option value="half
head">Half-head</option>
                <option value="hollow
triangle">Hollow Triangle</option>
                <option value="filled
triangle">Filled Triangle</option>
                <option value="hollow
diamond">Hollow Diamond</option>
                <option value="filled
diamond">Filled Diamond</option>
                <option
value="ball">Ball</option>
                <option
value="socket">Socket</option>
              </select>
            <% } else
if(style.equals("linestyle")) { %>
              <select id="<%= objects[i]
%>_<%= style %>"
onchange="javascript:change('<%= style
%>', value)">
                <option
value="solid">Solid</option>
                <option
value="dashed">Dashed</option>
                <option
value="dotted">Dotted</option>
              </select>
            <% } else { %>
              <input type="text" id="<%=
objects[i] %>_<%= style %>"

onchange="javascript:change('<%= style
%>', value)"/>
            <% } %>
            </td>
          </tr>
          <%  } %>
          <%
if(objects[i].equals("polygon") ||
objects[i].equals("polyline")) { %>
          <tr>
            <td colspan="2">
              <input id="<%= objects[i]
%>_addpoint" type="button"
value="Add Point"

onclick="javascript:addPoint()"/>
              <div id="<%= objects[i]
%>_addpoint_message" style="display:
none;">
              Click the canvas to add a
point.</div>
            </td>
          </tr>
          <tr>
            <td colspan="2">
              <input id="<%= objects[i]
%>_removepoint" type="button"
value="Remove Point"

onclick="javascript:removePoint()"/>
              <div id="<%= objects[i]
%>_removepoint_message"
style="display: none;">
                Click the canvas to remove
a point.</div>
            </td>
          </tr>
          <%  } %>
        </table>
        <table id="<%= objects[i]
%>PropertiesTable" style="display:
none;">
          <tr><th
colspan="2">Properties</th></tr>
          <%  for(String property :
properties[i]) { %>
          <tr>
            <td><%= property %></td>
            <%
if(property.equals("qualifiedassociation"
) || property.equals("activeclass") ||

property.equals("templateclass") ||
property.equals("abstractclass") ||

property.equals("showattributes") ||
property.equals("showoperations") ||
```

```
property.equals("showartifacts") ||
property.equals("showinternalactivities"
)) { %>
                <td>
                  <input type="checkbox"
id="<%= objects[i] %>_<%= property
%>"
                         name="" value=""

onclick="javascript:changeBoolean('<%=
property %>', checked)" />
                </td>
            <% } else { %>
              <td> <input type="text"
id="<%= objects[i] %>_<%= property
%>"

onchange="javascript:change('<%=
property %>', value)"/> </td>
            <% } %>
          </tr>
          <%  } %>
        </table>
<%  } %>

      <table id="attributesTable"
style="display: none;">
          <tr><th
colspan="2">Attributes</th></tr>
          <tr>
            <td rowspan="4"> <select
id="attributesList" size="4"

onchange="javascript:itemSelect('attrib
ute')"/>
            </td>
            <td> <input type="button"
value="New"
onclick="javascript:itemNew('attribute')
"/> </td>
          </tr>
          <tr> <td> <input type="button"
id="attributeDelete" value="Delete"
disabled

onclick="javascript:itemDelete('attribut
e')"/> </td> </tr>
          <tr> <td> <input type="button"
id="attributeUp" value="Up" disabled

onclick="javascript:itemUp('attribute')"/
> </td> </tr>
          <tr> <td> <input type="button"
id="attributeDown" value="Down"
disabled

onclick="javascript:itemDown('attribute'
)"/> </td> </tr>
          <tr>
            <td> <input type="text"
id="attributeEdit" disabled

onchange="javascript:itemEdit('attribut
e', 'value', value)"/> </td>
            <td> <label><input
type="checkbox" id="attributeStatic"
value="static"

onclick="javascript:itemEdit('attribute',
'static', checked)"
                         disabled />
Static</label>
            </td>
          </tr>
        </table>
        <table id="operationsTable"
style="display: none;">
          <tr><th
colspan="2">Operations</th></tr>
          <tr>
            <td rowspan="4"> <select
id="operationsList" size="4"

onchange="javascript:itemSelect('opera
tion')"/>
            </td>
            <td> <input type="button"
value="New"
onclick="javascript:itemNew('operation'
)"/> </td>
          </tr>
          <tr> <td> <input type="button"
id="operationDelete" value="Delete"
disabled

onclick="javascript:itemDelete('operatio
n')"/> </td> </tr>
          <tr> <td> <input type="button"
id="operationUp" value="Up" disabled

onclick="javascript:itemUp('operation')"
/> </td> </tr>
          <tr> <td> <input type="button"
id="operationDown" value="Down"
disabled

onclick="javascript:itemDown('operatio
n')"/> </td> </tr>
          <tr>
            <td> <input type="text"
id="operationEdit" disabled

onchange="javascript:itemEdit('operatio
n', 'value', value)"/> </td>
            <td><label><input
type="checkbox" id="operationStatic"
value="static"

onclick="javascript:itemEdit('operation',
'static', checked)"
                         disabled />Static</label>
            </td>
          </tr>
          <tr>
            <td></td>
            <td>
              <label><input
type="checkbox"
id="operationAbstract" value="abstract"

onclick="javascript:itemEdit('operation',
'abstract', checked)"
                         disabled
/>Abstract</label>
            </td>
          </tr>
        </table>
        <table id="templatesTable"
style="display: none;">
          <tr><th
colspan="2">Templates</th></tr>
          <tr>
            <td rowspan="4"> <select
id="templatesList" size="4"

onchange="javascript:itemSelect('templ
ate')"/>
            </td>
            <td> <input type="button"
value="New"
onclick="javascript:itemNew('template')
"/> </td>
          </tr>
          <tr> <td> <input type="button"
id="templateDelete" value="Delete"
disabled

onclick="javascript:itemDelete('templat
e')"/> </td> </tr>
          <tr> <td> <input type="button"
id="templateUp" value="Up" disabled

onclick="javascript:itemUp('template')"/
> </td> </tr>
          <tr> <td> <input type="button"
id="templateDown" value="Down"
disabled

onclick="javascript:itemDown('template
')"/> </td> </tr>
          <tr>
            <td> <input type="text"
id="templateEdit" disabled

onchange="javascript:itemEdit('templat
e', 'value', value)"/> </td>
          </tr>
        </table>
        <table id="taggedvaluesTable"
style="display: none;">
          <tr><th colspan="2">Tagged
Values</th></tr>
          <tr>
            <td rowspan="4"> <select
id="taggedvaluesList" size="4"

onchange="javascript:itemSelect('tagge
dvalue')"/>
            </td>
            <td> <input type="button"
value="New"
onclick="javascript:itemNew('taggedval
ue')"/> </td>
          </tr>
```

```
        <tr> <td> <input type="button"
id="taggedvalueDelete" value="Delete"
disabled

onclick="javascript:itemDelete('taggedv
alue')"/> </td> </tr>
        <tr> <td> <input type="button"
id="taggedvalueUp" value="Up"
disabled

onclick="javascript:itemUp('taggedvalue
')"/> </td> </tr>
        <tr> <td> <input type="button"
id="taggedvalueDown" value="Down"
disabled

onclick="javascript:itemDown('taggedva
lue')"/> </td> </tr>
        <tr>
          <td> <input type="text"
id="taggedvalueEdit" disabled

onchange="javascript:itemEdit('taggedv
alue', 'value', value)"/> </td>
        </tr>
      </table>
      <table id="containedartifactsTable"
style="display: none;">
        <tr><th
colspan="2">Artifacts</th></tr>
        <tr>
          <td rowspan="4"> <select
id="containedartifactsList" size="4"

onchange="javascript:itemSelect('contai
nedartifact')"/>
          </td>
          <td> <input type="button"
value="New"
onclick="javascript:itemNew('contained
artifact')"/> </td>
        </tr>
        <tr> <td> <input type="button"
id="containedartifactDelete"
value="Delete" disabled

onclick="javascript:itemDelete('containe
dartifact')"/> </td> </tr>
        <tr> <td> <input type="button"
id="containedartifactUp" value="Up"
disabled

onclick="javascript:itemUp('containedar
tifact')"/> </td> </tr>
        <tr> <td> <input type="button"
id="containedartifactDown"
value="Down" disabled

onclick="javascript:itemDown('containe
dartifact')"/> </td> </tr>
        <tr>
          <td> <input type="text"
id="containedartifactEdit" disabled

onchange="javascript:itemEdit('contain
edartifact', 'value', value)"/> </td>
        </tr>
      </table>
      <table id="internalactivitiesTable"
style="display: none;">
        <tr><th colspan="2">Internal
Activities</th></tr>
        <tr>
          <td rowspan="4"> <select
id="internalactivitiesList" size="4"

onchange="javascript:itemSelect('intern
alactivity')"/>
          </td>
          <td> <input type="button"
value="New"
onclick="javascript:itemNew('internalac
tivity')"/> </td>
        </tr>
        <tr> <td> <input type="button"
id="internalactivityDelete"
value="Delete" disabled

onclick="javascript:itemDelete('internal
activity')"/> </td> </tr>
        <tr> <td> <input type="button"
id="internalactivityUp" value="Up"
disabled

onclick="javascript:itemUp('internalacti
vity')"/> </td> </tr>
        <tr> <td> <input type="button"
id="internalactivityDown"
value="Down" disabled

onclick="javascript:itemDown('internala
ctivity')"/> </td> </tr>
        <tr>
          <td> <input type="text"
id="internalactivityEdit" disabled

onchange="javascript:itemEdit('internal
activity', 'value', value)"/> </td>
        </tr>
      </table>
    </div>
</div>
```

---

toolbox.jsp

```
<div id="toolBox">
  <table id="toolSettings">
    <tbody>
      <tr>
        <td colspan="2">
          <div
id="diagramTitle"></div>
        </td>
      </tr>
      <tr>
        <td colspan="2">
          <jsp:include page="basic
buttons.jsp" />
        </td>
      </tr>
      <tr>
        <td colspan="2">
          <select
id="defaultArrowhead1"
onchange="javascript:changeSetting('ar
rowhead1', value)">
            <option
value="none">None</option>
            <option
value="lines">Lines</option>
            <option value="half
head">Half-head</option>
            <option value="hollow
triangle">Hollow Triangle</option>
            <option value="filled
triangle">Filled Triangle</option>
            <option value="hollow
diamond">Hollow Diamond</option>
            <option value="filled
diamond">Filled Diamond</option>
            <option
value="ball">Ball</option>
            <option
value="socket">Socket</option>
          </select>
          <select id="defaultLinestyle"
onchange="javascript:changeSetting('lin
estyle', value)">
            <option
value="solid">Solid</option>
            <option
value="dashed">Dashed</option>
            <option
value="dotted">Dotted</option>
          </select>
          <select
id="defaultArrowhead2"
onchange="javascript:changeSetting('ar
rowhead2', value)">
            <option
value="none">None</option>
            <option
value="lines">Lines</option>
            <option value="half
head">Half-head</option>
            <option value="hollow
triangle">Hollow Triangle</option>
            <option value="filled
triangle">Filled Triangle</option>
            <option value="hollow
diamond">Hollow Diamond</option>
            <option value="filled
diamond">Filled Diamond</option>
            <option
value="ball">Ball</option>
            <option
value="socket">Socket</option>
          </select>
        </td>
      </tr>
      <tr>
        <td>Fill Color
          <input class="color" value=""
onchange="javascript:changeColorSetti
ng('Fill')"
            id="defaultFill"/>
          a: <input type="text"
class="rgbaInput" id="defaultFillA"
```

```
onchange="javascript:changeColorSetting('Fill')"/>
          </td>
        </tr>
        <tr>
          <td>Stroke Color
            <input class="color" value=""
onchange="javascript:changeColorSetting('Stroke')"
                id="defaultStroke"/>
            a: <input type="text"
class="rgbaInput" id="defaultStrokeA"

onchange="javascript:changeColorSetting('Stroke')"/>
          </td>
        </tr>
        <tr>
          <td>
            Font Size
            <input type="text"
id="selectedFontsize"

onchange="javascript:changeSetting('fontsize', value)"/>
            Font Family
            <select
id="selectedFontfamily"
onchange="javascript:changeSetting('fontfamily', value)">
              <option value="sans-
serif">Sans-serif</option>
            </select>
          </td>
        </tr>
        <tr>
          <td>Text Color
            <input class="color" value=""
onchange="javascript:changeColorSetting('Text')"
                id="defaultText"/>
            a: <input type="text"
class="rgbaInput" id="defaultTextA"

onchange="javascript:changeColorSetting('Text')"/>
          </td>
        </tr>
      </tbody>
    </table>

    <!--<table id="toolSettings">
      <tbody>
        <tr>
          <td colspan="3">
            <select
id="defaultArrowhead1"
onchange="javascript:changeSetting('arrowhead1', value)">
              <option
value="none">None</option>
              <option
value="lines">Lines</option>
              <option value="half
head">Half-head</option>
              <option value="hollow
triangle">Hollow Triangle</option>
              <option value="filled
triangle">Filled Triangle</option>
              <option value="hollow
diamond">Hollow Diamond</option>
              <option value="filled
diamond">Filled Diamond</option>
              <option
value="ball">Ball</option>
              <option
value="socket">Socket</option>
            </select>
            <select id="defaultLinestyle"
onchange="javascript:changeSetting('linestyle', value)">
              <option
value="solid">Solid</option>
              <option
value="dashed">Dashed</option>
              <option
value="dotted">Dotted</option>
            </select>
            <select
id="defaultArrowhead2"
onchange="javascript:changeSetting('arrowhead2', value)">
              <option
value="none">None</option>
              <option
value="lines">Lines</option>
              <option value="half
head">Half-head</option>
              <option value="hollow
triangle">Hollow Triangle</option>
              <option value="filled
triangle">Filled Triangle</option>
              <option value="hollow
diamond">Hollow Diamond</option>
              <option value="filled
diamond">Filled Diamond</option>
              <option
value="ball">Ball</option>
              <option
value="socket">Socket</option>
            </select>
          </td>
        </tr>
        <tr>
          <td><input class="color"
value="66ff00"></td>
        </tr>
        <tr>
          <td>Fill</td>
          <td><div
id="selectedFillColor"></div></td>
          <td>
            R: <input type="text"
class="rgbaInput" id="defaultFillR"

onchange="javascript:changeColorSetting('Fill')"/>
            G: <input type="text"
class="rgbaInput" id="defaultFillG"

onchange="javascript:changeColorSetting('Fill')"/>
            B: <input type="text"
class="rgbaInput" id="defaultFillB"

onchange="javascript:changeColorSetting('Fill')"/>
            a: <input type="text"
class="rgbaInput" id="defaultFillA"

onchange="javascript:changeColorSetting('Fill')"/>
          </td>
        </tr>
        <tr>
          <td>Stroke</td>
          <td><div
id="selectedStrokeColor"></div></td>
          <td>
            R: <input type="text"
class="rgbaInput" id="defaultStrokeR"

onchange="javascript:changeColorSetting('Stroke')"/>
            G: <input type="text"
class="rgbaInput" id="defaultStrokeG"

onchange="javascript:changeColorSetting('Stroke')"/>
            B: <input type="text"
class="rgbaInput" id="defaultStrokeB"

onchange="javascript:changeColorSetting('Stroke')"/>
            a: <input type="text"
class="rgbaInput" id="defaultStrokeA"

onchange="javascript:changeColorSetting('Stroke')"/>
          </td>
        </tr>
        <tr>
          <td colspan="3">
            Font Size
            <input type="text"
id="selectedFontsize"

onchange="javascript:changeSetting('fontsize', value)"/>
            Font Family
            <select
id="selectedFontfamily"
onchange="javascript:changeSetting('fontfamily', value)">
              <option value="sans-
serif">Sans-serif</option>
            </select>
          </td>
        </tr>
        <tr>
          <td>Text Color</td>
          <td><div
id="selectedTextColor"></div></td>
          <td>
```

```
            R: <input type="text"
class="rgbaInput" id="defaultTextR"

onchange="javascript:changeColorSetti
ng('Text')"/>
            G: <input type="text"
class="rgbaInput" id="defaultTextG"

onchange="javascript:changeColorSetti
ng('Text')"/>
            B: <input type="text"
class="rgbaInput" id="defaultTextB"

onchange="javascript:changeColorSetti
ng('Text')"/>
            a: <input type="text"
class="rgbaInput" id="defaultTextA"

onchange="javascript:changeColorSetti
ng('Text')"/>
          </td>
        </tr>
      </tbody>
    </table>-->

    <div id="shapeButtons">
      <ul>
        <li><input type="image"
src="../images/select.png" title="Select"
value="select" id="select"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/line.png" title="Line"
value="line" id="line"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/rectangle.png"
title="Rectangle" value="rectangle"
id="rectangle"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/ellipse.png"
title="Ellipse" value="ellipse"
id="ellipse"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/text.png" title="Text"
value="text" id="text"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/polygon.png"
title="Polygon" value="polygon"
id="polygon"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/polyline.png"
title="Polyline" value="polyline"
id="polyline"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/bezier.png" title="Bezier
Curve" value="bezier" id="bezier"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/path.png" title="Path"
value="path" id="path"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/arrowhead.png"
title="Arrowhead" value="arrowhead"
id="arrowhead"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/activation bar.png"
title="Activation Bar"
value="activationbar" id="activationbar"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/lifeline.png"
title="Lifeline" value="lifeline"
id="lifeline"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/deletion.png"
title="Deletion" value="deletion"
id="deletion"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/initial pseudostate.png"
title="Initial Pseudostate"
value="initialps" id="initialps"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/final state.png"
title="Final State" value="finalstate"
id="finalstate"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/shallow history
pseudostate.png" title="Shallow History
Pseudostate" value="shallowhistps"
id="shallowhistps"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/deep history
pseudostate.png" title="Deep History
Pseudostate" value="deephistps"
id="deephistps"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/initial node.png"
title="Initial Node" value="initialnode"
id="initialnode"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/final node.png"
title="Final Node" value="finalnode"
id="finalnode"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/black bar.png"
title="Black Bar" value="blackbar"
id="blackbar"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/diamond.png"
title="Diamond" value="diamond"
id="diamond"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/pin.png" title="Pin"
value="pin" id="pin"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/flow final.png"
title="Flow Final" value="flowfinal"
id="flowfinal"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/port.png" title="Port"
value="port" id="port"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/class.png" title="Class"
value="classnotation"
id="classnotation"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/note.png" title="Note"
value="note" id="note"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/instance
specification.png" title="Instance
Specification" value="instance"
id="instance"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/node.png" title="Node"
value="node" id="node"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/state box.png"
title="State Box" value="statebox"
id="statebox"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/expansion region.png"
title="Expansion Region"
```

```
value="expansionregion"
id="expansionregion"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/frame.png"
title="Frame" value="frame" id="frame"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/small package.png"
title="Small Package" value="package1"
id="package1"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/large package.png"
title="Large Package" value="package2"
id="package2"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/component.png"
title="Component" value="component"
id="component"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/artifact.png"
title="Artifact" value="artifact"
id="artifact"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/use case.png" title="Use
Case" value="usecase" id="usecase"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/actor.png" title="Actor"
value="actor" id="actor"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/superstate.png"
title="Superstate" value="superstate"
id="superstate"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/action.png"
title="Action" value="action"
id="action"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/subactivity.png"
title="Subactivity" value="subactivity"
id="subactivity"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/time signal.png"
title="Time Signal" value="timesignal"
id="timesignal"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/accept signal.png"
title="Accept Signal"
value="acceptsignal" id="acceptsignal"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/send signal.png"
title="Send Signal" value="sendsignal"
id="sendsignal"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/connector.png"
title="Connector" value="connector"
id="connector"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/transformation.png"
title="Transformation"
value="transformation"
id="transformation"
onclick="javascript:activateTool(value)"/
></li>
        <li><input type="image"
src="../images/part.png" title="Part"
value="part" id="part"
onclick="javascript:activateTool(value)"/
></li>
      </ul>
    </div>
</div>
```

---

canvas.js

```
var diagramObjects = [];
var noOfDiagramObjects;
var tempObject; //temporary holder for
objects

var canvas;
var context;
var WIDTH;
var HEIGHT;

var canvasValid = false;
var dlFrame;

function canvas_init() {
    canvas =
document.getElementById('drawingAre
a');

    if(!canvas) {
        alert("cannot find canvas");
        return;
    }

    if(!canvas.getContext) {
        alert("no canvas.getContext!");
        return;
    }
    context = canvas.getContext('2d');
    if(!context) {
        alert("failed to get context!");
        return;
    }

    WIDTH = canvas.width;
    HEIGHT = canvas.height;
    noOfDiagramObjects =
diagramObjects.length;

    setInterval(mainDraw,
DRAW_INTERVAL);

    //fixes a problem where double
clicking causes text to get selected on
the canvas
    canvas.onselectstart = function ()
{return false;}

    dlFrame =
document.getElementById("downloadFr
ame");
}

//only draw when canvas gets
invalidated
function mainDraw() {
    if (canvasValid == false) {
        context.clear();
        for(var i = 0; i <
noOfDiagramObjects; i++)
context.draw(diagramObjects[i]);
        canvasValid = true;
    }
}

function invalidate() {
    canvasValid = false;
}

function saveAsImage() {
    var indexTempSelected =
diagramObjects.indexOf(selectedObj);
    selectNone();
    mainDraw();

    var dataURI =
canvas.toDataURL("image/png");
    var dataPart = dataURI.indexOf(",") +
1;

    if(indexTempSelected >= 0) {
        select(indexTempSelected);
        mainDraw();
    }

    var xhReq = new
createXMLHttpRequest();
    xhReq.onreadystatechange =
addCollaboratorReqResult;
    xhReq.open("POST", URL,
false);//true);
    xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
```

```
xhReq.send("action=createImage&data
=" +
encodeURIComponent(dataURI.substr(d
ataPart))
      + "&id=" + DIAGRAM_ID +
"&editorID=" + editorID);
   //+ "&dataurl=" +
encodeURIComponent(canvas.toDataUR
L("image/png")));

//alert(canvas.toDataURL("image/png"))
;

//window.open(canvas.toDataURL("ima
ge/png"));
   //alert(dataURI.substr(dataPart));

   window.location.href =
"../members/file?editorID=" + editorID +
"&id=" + DIAGRAM_ID;
   // + "&data=" +
encodeURIComponent(dataURI);//.subs
tr(dataPart));
   //dlFrame.src = "diagram.png";
}
```

---

collaborators.js

```
var creator;
var collaborators;
var currentusers;
var currentusersList;
var collaboratorsList;
var isCreator = false;

function collaborators_init() {
   isCreator = (creator == USERNAME);

   collaboratorsList =
document.getElementById("collaborato
rs");
   //currentusersList =
document.getElementById("currentuser
s");

   for(var i = 0; i < collaborators.length;
i++) {
      // Display in collaborators table
      var collaboratorTbody =
document.createElement("tbody");
      var collaboratorRow1 =
document.createElement("tr");
      var collaboratorRow2 =
document.createElement("tr");
      var collaboratorRow3 =
document.createElement("tr");
      var collaboratorRow4 =
document.createElement("tr");
      var collaboratorRow5 =
document.createElement("tr");
      var colR1C1 =
document.createElement("td");
      var colR1C2 =
document.createElement("td");
      var colR2C1 =
document.createElement("td");
      var colR3C1 =
document.createElement("td");
      var colR4C1 =
document.createElement("td");

      var userPicContainer =
document.createElement("div");

userPicContainer.setAttribute("class",
"displayPicContainer2");
      var userPic =
document.createElement("img");
      userPic.setAttribute("class",
"displayPic2");
      userPic.setAttribute("src",
"DisplayPicture?imgTitle=" +
collaborators[i].displaypic);

userPicContainer.appendChild(userPic);


colR1C1.appendChild(userPicContainer);
      colR1C1.setAttribute("rowspan",
"5");

colR1C2.appendChild(document.createT
extNode(collaborators[i].username));

colR2C1.appendChild(document.createT
extNode(collaborators[i].firstname +
         " " + collaborators[i].lastname));

colR3C1.appendChild(document.createT
extNode(collaborators[i].email));


if(currentusers.indexOf(collaborators[i].
username) == -1) {

colR4C1.appendChild(document.createT
extNode("Offline"));

collaboratorTbody.setAttribute("class",
"offlineSign");
      } else {

colR4C1.appendChild(document.createT
extNode("Online"));

collaboratorTbody.setAttribute("class",
"onlineSign");
      }

collaboratorRow4.appendChild(colR4C1
);

      if(isCreator &&
(collaborators[i].username != creator)) {
         var colR5C1 =
document.createElement("td");
         var removeButton =
document.createElement("a");

removeButton.setAttribute("href",
"javascript:removeCollaboratorReq('"
            + collaborators[i].username +
"')");

removeButton.setAttribute("class",
"removeCollaboratorButton");

removeButton.appendChild(document.c
reateTextNode("Remove"));

colR5C1.appendChild(removeButton);

collaboratorRow5.appendChild(colR5C1
);
      }


collaboratorRow1.appendChild(colR1C1
);

collaboratorRow1.appendChild(colR1C2
);

collaboratorRow2.appendChild(colR2C1
);

collaboratorRow3.appendChild(colR3C1
);

collaboratorRow4.appendChild(colR4C1
);

collaboratorTbody.appendChild(collabor
atorRow1);

collaboratorTbody.appendChild(collabor
atorRow2);

collaboratorTbody.appendChild(collabor
atorRow3);

collaboratorTbody.appendChild(collabor
atorRow4);

collaboratorTbody.appendChild(collabor
atorRow5);

collaboratorTbody.setAttribute("id",
"collaborator_" +
escape(collaborators[i].username));

collaboratorsList.appendChild(collabora
torTbody);

   }

   if(!isCreator) {

document.getElementById("addCollabor
atorSection").setAttribute("style",
"display: none;");

document.getElementById("addCollabor
```

```
atorSubTab").setAttribute("style",
"display: none;");
   }
}

function addCollaborator(collaborator) {
   var newIndex =
collaborators.push(collaborator) - 1;

   var collaboratorTbody =
document.createElement("tbody");
   var collaboratorRow1 =
document.createElement("tr");
   var collaboratorRow2 =
document.createElement("tr");
   var collaboratorRow3 =
document.createElement("tr");
   var collaboratorRow4 =
document.createElement("tr");
   var collaboratorRow5 =
document.createElement("tr");
   var colR1C1 =
document.createElement("td");
   var colR1C2 =
document.createElement("td");
   var colR2C1 =
document.createElement("td");
   var colR3C1 =
document.createElement("td");
   var colR4C1 =
document.createElement("td");

   var userPicContainer =
document.createElement("div");
   userPicContainer.setAttribute("class",
"displayPicContainer2");
   var userPic =
document.createElement("img");
   userPic.setAttribute("class",
"displayPic2");
   userPic.setAttribute("src",
"DisplayPicture?imgTitle=" +
collaborators[newIndex].displaypic);

userPicContainer.appendChild(userPic);


colR1C1.appendChild(userPicContainer);
   colR1C1.setAttribute("rowspan", "5");

colR1C2.appendChild(document.createT
extNode(collaborators[newIndex].usern
ame));

colR2C1.appendChild(document.createT
extNode(collaborators[newIndex].firstn
ame +
      " " +
collaborators[newIndex].lastname));

colR3C1.appendChild(document.createT
extNode(collaborators[newIndex].email
));

if(currentusers.indexOf(collaborators[ne
wIndex].username) == -1) {

colR4C1.appendChild(document.createT
extNode("Offline"));

collaboratorTbody.setAttribute("class",
"offlineSign");
   } else {

colR4C1.appendChild(document.createT
extNode("Online"));

collaboratorTbody.setAttribute("class",
"onlineSign");
   }

collaboratorRow4.appendChild(colR4C1
);

   if(isCreator) {
      var colR5C1 =
document.createElement("td");
      var removeButton =
document.createElement("a");
      removeButton.setAttribute("href",
"javascript:removeCollaboratorReq('"
         +
collaborators[newIndex].username +
"')");
      removeButton.setAttribute("class",
"removeCollaboratorButton");

removeButton.appendChild(document.c
reateTextNode("Remove"));

colR5C1.appendChild(removeButton);

collaboratorRow5.appendChild(colR5C1
);
   }


collaboratorRow1.appendChild(colR1C1
);

collaboratorRow1.appendChild(colR1C2
);

collaboratorRow2.appendChild(colR2C1
);

collaboratorRow3.appendChild(colR3C1
);

collaboratorRow4.appendChild(colR4C1
);

collaboratorTbody.appendChild(collabor
atorRow1);

collaboratorTbody.appendChild(collabor
atorRow2);

collaboratorTbody.appendChild(collabor
atorRow3);

collaboratorTbody.appendChild(collabor
atorRow4);

collaboratorTbody.appendChild(collabor
atorRow5);
   collaboratorTbody.setAttribute("id",
"collaborator_" +
escape(collaborators[newIndex].userna
me));

collaboratorsList.appendChild(collabora
torTbody);
}

function addUser(collaborator) {
   if(currentusers.indexOf(collaborator)
!= -1) return;
   currentusers.push(collaborator)

   var collaboratorTbody =
document.getElementById("collaborato
r_" + escape(collaborator));

collaboratorTbody.removeAttribute("cla
ss");

collaboratorTbody.setAttribute("class",
"onlineSign");
   var colR4C1=
collaboratorTbody.childNodes[3];

colR4C1.removeChild(colR4C1.childNod
es[0]);

colR4C1.appendChild(document.createT
extNode("Online"));
}

function removeCollaborator(target) {
   var targetPos = -1;

   for(var i = 0; i < collaborators.length;
i++) {
      if(collaborators[i].username ==
target)
         targetPos = i;
   }

   if(targetPos == -1) return;
   collaborators.splice(targetPos, 1);


collaboratorsList.removeChild(documen
t.getElementById("collaborator_" +
escape(target)));
}

function removeCurrentUser(target) {
   var targetPos = -1;

   for(var i = 0; i < currentusers.length;
i++) {
```

```
    if(currentusers[i] == target) {
      targetPos = i;
      break;
    }
  }

  if(targetPos == -1) return;
  currentusers.splice(targetPos, 1);


//currentusersList.removeChild(docume
nt.getElementById("currentuser_" +
escape(target)));
  var collaboratorTbody =
document.getElementById("collaborato
r_" + escape(target));

collaboratorTbody.removeAttribute("cla
ss");

collaboratorTbody.setAttribute("class",
"offlineSign");
  var colR4C1=
collaboratorTbody.childNodes[3];

colR4C1.removeChild(colR4C1.childNod
es[0]);

colR4C1.appendChild(document.createT
extNode("Offline"));
}

function enableAddReqCollaborator() {

document.getElementById('enableAddC
ollaboratorButton').disabled = true;

document.getElementById('addCollabor
atorText').disabled = false;

document.getElementById('addCollabor
atorButton').disabled = false;

document.getElementById('cancelAddC
ollaborator').disabled = false;

document.getElementById('addCollabor
atorText').focus();
}

function addCollaboratorReq() {
  var xhReq = new
createXMLHttpRequest();
  xhReq.onreadystatechange =
addCollaboratorReqResult;
  xhReq.open("POST", URL, true);
  xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");

xhReq.send("action=addcollaborator&id
=" + DIAGRAM_ID + "&collaborator="
    +
encodeURIComponent(document.getEle
mentById('addCollaboratorText').value)
    + "&editorID=" + editorID);
  cancelAddCollaborator();
}

function cancelAddCollaborator() {

document.getElementById('addCollabor
atorText').value = '';

document.getElementById('addCollabor
atorText').disabled = true;

document.getElementById('addCollabor
atorButton').disabled = true;

document.getElementById('cancelAddC
ollaborator').disabled = true;

document.getElementById('enableAddC
ollaboratorButton').disabled = false;
}

function
removeCollaboratorReq(toRemove) {
  if(!window.confirm("Remove " +
toRemove + " from list of
collaborators?"))
    return;

  var xhReq = new
createXMLHttpRequest();
  xhReq.onreadystatechange =
removeCollaboratorReqResult;
  xhReq.open("POST", URL, true);
  xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");

xhReq.send("action=removecollaborato
r&id=" + DIAGRAM_ID +
"&collaborator="
    +
encodeURIComponent(toRemove)+"&e
ditorID=" + editorID);
}

function addCollaboratorReqResult() {
  if(this.readyState == 4 && this.status
== 200) {
    var result =
JSON.parse(this.responseText);
    if(result.error)
      alert(result.error);
    else if(result.response) {
      //alert(result.response);
    }
  }
}

function
removeCollaboratorReqResult() {
  if(this.readyState == 4 && this.status
== 200) {
    var result =
JSON.parse(this.responseText);
    if(result.error)
      alert(result.error);
    else if(result.response) {
      //alert(result.response);
    }
  }
}

/*function currentusersSubTab() {

(document.getElementById("collaborato
rsSection")).setAttribute('style', 'display:
none;');

(document.getElementById("addCollabo
ratorSection")).setAttribute('style',
'display: none;');

(document.getElementById("collaborato
rsSubTab2")).removeAttribute('class');

(document.getElementById("addCollabo
ratorSubTab2")).removeAttribute('class'
);


(document.getElementById("currentuse
rsSubTab2")).setAttribute('class',
'active');

(document.getElementById("currentuse
rsSection")).setAttribute('style', 'display:
block;');
}*/

function collaboratorsSubTab() {

//(document.getElementById("currentu
sersSection")).setAttribute('style',
'display: none;');

(document.getElementById("addCollabo
ratorSection")).setAttribute('style',
'display: none;');

//(document.getElementById("currentu
sersSubTab2")).removeAttribute('class');

(document.getElementById("addCollabo
ratorSubTab2")).removeAttribute('class'
);


(document.getElementById("collaborato
rsSubTab2")).setAttribute('class',
'active');

(document.getElementById("collaborato
rsSection")).setAttribute('style', 'display:
block;');
}

function addCollaboratorSubTab() {
  if(!isCreator) {
    return;
  }
```

```
//(document.getElementById("currentu
sersSection")).setAttribute('style',
'display: none;');

(document.getElementById("collaborato
rsSection")).setAttribute('style', 'display:
none;');

//(document.getElementById("currentu
sersSubTab2")).removeAttribute('class');

(document.getElementById("collaborato
rsSubTab2")).removeAttribute('class');


(document.getElementById("addCollabo
ratorSubTab2")).setAttribute('class',
'active');

(document.getElementById("addCollabo
ratorSection")).setAttribute('style',
'display: block;');
}
```

---

communication.js

```
var myMsgs = 1;
var otherMsgs = 0;
var outgoing = [];

var URL = "CordieEditor";
var receivedUpdate = true;
var requestUpdateIntervalID;
var lastEditTimeoutID;
var editorID;

function communication_init() {

//document.getElementById("realtimeU
pdatesCheckbox").defaultChecked =
true;

//document.getElementById("updateBu
tton").disabled = true;

//document.getElementById("realtimeU
pdatesCheckbox").defaultChecked =
false;

//document.getElementById("updateBu
tton").disabled = false;

//if(document.getElementById("realtim
eUpdatesCheckbox").checked) {
      requestUpdateIntervalID =
setInterval(requestForUpdates,
UPDATE_INTERVAL);
   //}
   lastEditTimeoutID =
setTimeout(editTimeoutReached,
EDIT_TIMEOUT);
}

function createXMLHttpRequest() {
  try {return new XMLHttpRequest();}
catch(e) {}
  try {return new
ActiveXObject("Msxml2.XMLHTTP");}
catch (e) {}
  try {return new
ActiveXObject("Microsoft.XMLHTTP");}
catch (e) {}

  alert("Sorry, you're browser does not
support XMLHttpRequest");
  return null;
}

function getDiagram() {
  var xhReq = new
createXMLHttpRequest();
  xhReq.open("POST", URL, false);
  xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
  xhReq.send("action=newUser&id=" +
DIAGRAM_ID);

  //alert(xhReq.responseText);
  try {
    var receivedDiagram =
JSON.parse(xhReq.responseText);
    if(receivedDiagram.error != null) {

displayError(receivedDiagram.error);

/*document.getElementById('errorMess
age').appendChild(

document.createTextNode(receivedDia
gram.error));


document.getElementById('content').se
tAttribute('style', 'display: none;');

document.getElementById('header').set
Attribute('style', 'display: none;');

document.getElementById('error').setAt
tribute('style', 'display: block;');

        document.getElementById('main-
content').setAttribute('style', 'display:
block;');*/
        return;
    }

    editorID =
receivedDiagram.editorID;
    receivedDiagram =
receivedDiagram.diagram;
  } catch(e) {
    displayError("Error getting
diagram");
    //alert("Error getting diagram");
  }
  diagramObjects =
receivedDiagram.objects;
  collaborators =
receivedDiagram.collaborators;
  currentusers =
receivedDiagram.currentusers;
  creator = receivedDiagram.creator;
  diagramTitle = receivedDiagram.title;

  document.getElementById('main-
content').setAttribute('style', 'display:
block;');
}

function generateOp(op) {
  send(op);
  outgoing.push({"op" : op, "myMsgs" :
myMsgs});

  displayOp(op); //show for illustrating

  myMsgs++;
}

function send(op) {
  var xhReq = new
createXMLHttpRequest();
  xhReq.onreadystatechange = sent;
  xhReq.open("POST", URL, true);
  xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
  xhReq.send("action=send&id=" +
DIAGRAM_ID + translateOp(op, 2) +
"&myMsgs="
    + myMsgs + "&otherMsgs=" +
otherMsgs + "&editorID=" + editorID);

  clearTimeout(lastEditTimeoutID);
  lastEditTimeoutID =
setTimeout(editTimeoutReached,
EDIT_TIMEOUT);

  //alert("action=send&id=" +
DIAGRAM_ID + translateOp(op, 2) +
"&myMsgs="
  //   + myMsgs + "&otherMsgs=" +
otherMsgs + "&editorID=" + editorID);
}

function sent() {
  if(this.readyState == 4 && this.status
== 200) {
    if(this.responseText != "") {
      var res =
JSON.parse(this.responseText);
      if(res.errorType == 1) {
        displayError(res.error);
      } else {
        alert(res.error);
      }
    }
  }
}

function updatesReady() {
  if(this.readyState == 4 && this.status
== 200) {
```

```
      receivedUpdate = true;

      if(this.responseText == "") {
         return;
      }

      var updates =
JSON.parse(this.responseText);
      if(updates.error) {
        if(updates.errorType == 1) {
           displayError(updates.error);
        } else {
           alert(updates.error);
        }
      }

      //alert(this.responseText);

      for(var i = 0; i < updates.length; i++)
{
         for(var j = 0; j < outgoing.length;
j++) {
           if(outgoing[j].myMsgs <
updates[i].otherMsgs) {
              outgoing.splice(j, 1);
              j--;
           }
         }

         //displayOutgoing(updates[i]);
//test

         for(var l = 0; l < outgoing.length;
l++) {
           transform(updates[i].op,
outgoing[l].op);
         }

         apply(updates[i].op)
         otherMsgs++;

         displayApplied(updates[i]); // test
      }
   }
}

function displayError(errorMsg) {

document.getElementById('errorMessa
ge').appendChild(document.createText
Node(errorMsg));


document.getElementById('content').se
tAttribute('style', 'display: none;');

document.getElementById('header').set
Attribute('style', 'display: none;');

document.getElementById('error').setAt
tribute('style', 'display: block;');

   document.getElementById('main-
content').setAttribute('style', 'display:
block;');
```

```
clearInterval(requestUpdateIntervalID);
   clearTimeout(lastEditTimeoutID);
   return;
}

function requestForUpdates() {
  if(!receivedUpdate)
     return;

  var xhReq = new
createXMLHttpRequest();
  xhReq.onreadystatechange =
updatesReady;
  xhReq.open("POST", URL, true);
  xhReq.setRequestHeader("Content-
Type", "application/x-www-form-
urlencoded");
  xhReq.send("action=askUpdate&id="
+ DIAGRAM_ID + "&editorID=" +
editorID);

  receivedUpdate = false;
}

function realtimeToggle() {

document.getElementById("updateButt
on").disabled =

document.getElementById("realtimeUp
datesCheckbox").checked;


if(document.getElementById("realtimeU
pdatesCheckbox").checked) {
      requestUpdateIntervalID =
setInterval(requestForUpdates,
UPDATE_INTERVAL);
   } else {

clearInterval(requestUpdateIntervalID);
   }
}

function editTimeoutReached() {
   generateOp(new noneOperation());
}
```

---

defaults.js

```
var UPDATE_INTERVAL = 1 * 1000;    //
get updates every UPDATE_INTERVAL
milliseconds
var DRAW_INTERVAL = 50;          //
calls the mainDraw every
DRAW_INTERVAL milliseconds
var EDIT_TIMEOUT = 10 * 1000;      //
timeout for the last edit done in
milliseconds

// Default settings for tools
var DEFAULT_TOOL = 'select';
var DEFAULT_FILLCOLOR =
"255,255,255,1";
var DEFAULT_STROKESTYLE = "0,0,0,1";
var DEFAULT_TEXTCOLOR = "0,0,0,1";
var DEFAULT_LINEWIDTH = 2;
var DEFAULT_LINECAP = "round";
var DEFAULT_LINEJOIN = "round";
var DEFAULT_ROTATE = 0;
var DEFAULT_FONTSTYLE = "normal";
var DEFAULT_FONTWEIGHT = "normal";
var DEFAULT_FONTSIZE = 14;
var DEFAULT_FONTFAMILY = "sans-
serif";
var DEFAULT_WIDTH = 150;
var DEFAULT_HEIGHT = 150;

// Selection handle defaults
var SELECTION_HANDLE_COLOR =
"220,20,60,1";
var SELECTION_HANDLE_WIDTH = 2;
var SELECTION_HANDLE_SIZE = 6;

// Arrowstyle defaults
var DEFAULT_ARROWSTYLE1 = "none";
var DEFAULT_ARROWSTYLE2 = "none";
var DEFAULT_LINESTYLE = "solid";
var ARROWHEAD_ANGLE = Math.PI / 8;
// angle of arrowhead
var ARROWHEAD_L = 15;
// length of arrowhead
var ARROWHEAD_H =
Math.abs(ARROWHEAD_L /
Math.cos(ARROWHEAD_ANGLE)); //
hypotenuse of arrowhead
var DIAMOND_ANGLE = Math.PI / 6;
var DIAMOND_L = 15;
var BALL_R = 7.5;
var SOCKET_R = 11.5

// Class notation defaults
var DEFAULT_CFONTSIZE = 14;
var DEFAULT_CNFONTSIZE = 18;
var H_MARGIN = 6; // horizontal margin

// Instance specification defaults
var DEFAULT_ISWIDTH = 200;
var DEFAULT_ISHEIGHT = 120;

// Node defaults
var DEFAULT_NODE_THICKNESS = 20;
var DEFAULT_TOPHEIGHT = 30;

// State box defaults
var DEFAULT_RADIUSRATIO = 0.15;

// Expansion region defaults
var DEFAULT_LISTBOXPINSIZE = 8;

// Others
var DEFAULT_SMALLSIZE = 30;
var DEFAULT_MEDIUMSIZE = 70;
var DEFAULT_XSMALLSIZE = 15;

var DEFAULT_ACTIVATIONBAR_W = 25;
var DEFAULT_ACTIVATIONBAR_H = 90;

var DEFAULT_LIFELINE_W = 25;
var DEFAULT_LIFELINE_H = 160;
```

```
var DEFAULT_DELETION_LINEWIDTH =
6;
var DEFAULT_DELETION_SIZE = 50;

var DEFAULT_FINALSTATE_BORDER =
0.30;

var DEFAULT_BLACKBAR_THICKNESS =
5;

var DEFAULT_FRAME_W = 270;
var DEFAULT_FRAME_H = 180;

var DEFAULT_PACKAGE2_W = 270;
var DEFAULT_PACKAGE2_H = 180;

var DEFAULT_ACTIONRADIUS = 0.4;

var DEFAULT_USECASE_W = 150;
var DEFAULT_USECASE_H = 100;

var DEFAULT_SUPERSTATE_W = 270;
var DEFAULT_SUPERSTATE_H = 180;

var DEFAULT_ACTION_W = 150;
var DEFAULT_ACTION_H = 100;

var DEFAULT_SUBACTIVITY_W = 150;
var DEFAULT_SUBACTIVITY_H = 100;

var DEFAULT_SENDSGN_W = 150;
var DEFAULT_SENDSGN_H = 100;

var DEFAULT_ACCEPTSGN_W = 150;
var DEFAULT_ACCEPTSGN_H = 100;

var DEFAULT_CONNECTOR_LABEL = "A";

var DEFAULT_TRANSFORMATION_W =
150;
var DEFAULT_TRANSFORMATION_H =
100;

var DEFAULT_PART_W = 150;
var DEFAULT_PART_H = 100;
```

---

developer.js

```
function displayOp(op){
   var opStr = "";

   switch(op.optype) {
      case 'insert' :
         opStr = "You inserted a(n) " +
op.object.objecttype;
         break;
      case 'delete' :
         opStr = "You deleted an object";
         break;
      case 'edit' :
         if(op.value == null)
            opStr = "You edited the " +
op.attribute + "s of " +
diagramObjects[op.position].objecttype;
         else
            opStr = "You edited the " +
op.attribute + " of " +
diagramObjects[op.position].objecttype;
         break;
      case 'move' :
         opStr = "You moved the z-order
of " +
diagramObjects[op.position].objecttype
+ " from "
            + op.position + " to " +
op.destination;
         break;
      case 'removecollaborator' :
         opStr = op.collaborator + " is no
longer a collaborator";
         break;
      case 'adduser' :
         opStr = op.collaborator + " has
joined the session";
         break;
      case 'removeuser' :
         opStr = op.collaborator + " has
left the session";
         break;
      case 'addcollaborator' :
         opStr = op.collaborator.username
+ " can now edit the diagram";
         break;
      default :
         return;
   }

   var testcontent2 =
document.createTextNode(opStr);
   var newrow2 =
document.createElement('tr');
   var newd2 =
document.createElement('td');
   newd2.appendChild(testcontent2);
   newrow2.appendChild(newd2);

document.getElementById('appliedOps
Display').appendChild(newrow2);

   /*var testcontent =
document.createTextNode(opStr + " |
myMsgs: " + myMsgs);
   var newrow =
document.createElement('tr');
   var newd =
document.createElement('td');
   newd.appendChild(testcontent);
   newrow.appendChild(newd);
   newrow.setAttribute("myMsgs",
myMsgs)

document.getElementById('outgoingDis
play').appendChild(newrow);*/
}

/*function displayOutgoing(update) {
   var ognodes =
document.getElementById('outgoingDis
play').childNodes;
   for(var k = 0; k < ognodes.length; k++)
{

if(ognodes[k].getAttribute("myMsgs") <
update.otherMsgs) {

document.getElementById('outgoingDis
play').removeChild(ognodes[k]);
         k--;
      }
   }
}*/

function displayApplied(update) {
   var opStr = "";
   var op = update.op;

   switch(op.optype) {
      case 'insert' :
         opStr = update.from + " inserted
a(n) " + op.object.objecttype;
         break;
      case 'delete' :
         opStr = update.from + " deleted
an object";
         break;
      case 'edit' :
         if(op.value == null)
            opStr = update.from + " edited
the " + op.attribute + "s of " +

diagramObjects[op.position].objecttype;
         else
            opStr = update.from + " edited
the " + op.attribute + " of " +

diagramObjects[op.position].objecttype;
         break;
      case 'move' :
         opStr = update.from + " moved
the z-order of " +
diagramObjects[op.position].objecttype
+ " from "
            + op.position + " to " +
op.destination;
         break;
      case 'removecollaborator' :
         opStr = op.collaborator + " is no
longer a collaborator";
         break;
      case 'adduser' :
         opStr = op.collaborator + " has
joined the session";
         break;
      case 'removeuser' :
         opStr = op.collaborator + " has
left the session";
         break;
      case 'addcollaborator' :
         opStr = op.collaborator.username
+ " can now edit the diagram";
         break;
      default :
         return;
   }
```

```
   //alert(JSON.stringify(update));

   var testcontent2 =
document.createTextNode(opStr);
   var newrow2 =
document.createElement('tr');
   var newd2 =
document.createElement('td');
   newd2.setAttribute('style', 'color:
#FF6600;');
   newd2.appendChild(testcontent2);
   newrow2.appendChild(newd2);

document.getElementById('appliedOps
Display').appendChild(newrow2);
}

function showTempObject() {
   alert(JSON.stringify(tempObject));
}

function saveCanvasTest() {
   var dataURL =
document.getElementById("drawingAre
a").toDataURL();

document.getElementById("canvasImg"
).src = dataURL;
}
```

---

diagram shapes 1.js

```
var selectionHandles = [];

CanvasRenderingContext2D.prototype.c
lear = function() {
   this.clearRect(0, 0, WIDTH, HEIGHT);
}

CanvasRenderingContext2D.prototype.s
etFillAndStroke = function(fill, stroke) {
   if (this === ghostContext) {
      this.fillStyle = 'black'; // always
want black for the ghost canvas
      this.strokeStyle = 'black';
   } else {
      this.fillStyle = "rgba(" + (fill ? fill :
stroke) + ")";
      this.strokeStyle = "rgba(" + stroke +
")";
   }
}

CanvasRenderingContext2D.prototype.s
etLineSettings = function(linewidth,
linecap, linejoin) {
   this.lineWidth = linewidth;
   this.lineCap = linecap;
   this.lineJoin = linejoin;
};

CanvasRenderingContext2D.prototype.d
raw = function(obj) {
   this.save();
   switch(obj.objecttype) {
      case "rectangle" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawRectangle(obj.x, obj.y,
obj.width, obj.height, obj.rotate);
         break;
      case "ellipse" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawEllipse(obj.x, obj.y,
obj.width, obj.height, obj.rotate);
         break;
      case "line" :

this.setFillAndStroke(obj.strokestyle,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawLine(obj.x1, obj.y1,
obj.x2, obj.y2, obj.arrowstyle1,
obj.arrowstyle2, obj.linestyle);
         break;
      case "path" :

this.setFillAndStroke(obj.strokestyle,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawPath(obj.points);
         break;
      case "polygon" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawPolygon(obj.points);
         break;
      case "text" :

this.setFillAndStroke(obj.textcolor,
obj.textcolor);

this.setLineSettings(DEFAULT_LINEWIDT
H, DEFAULT_LINECAP,
DEFAULT_LINEJOIN);
         this.drawText(obj);
         break;
      case "class" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawClass(obj);
         break;
      case "note" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawNote(obj);
         break;
      case "bezier" :

this.setFillAndStroke(obj.strokestyle,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawBezier(obj.x1, obj.y1,
obj.x2, obj.y2, obj.ctrl1x, obj.ctrl1y,
            obj.ctrl2x, obj.ctrl2y,
obj.arrowstyle1, obj.arrowstyle2,
obj.linestyle);
         break;
      case "instance" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawInstance(obj);
         break;
      case "node" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawNode(obj);
         break;
      case "statebox" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawStatebox(obj);
         break;
      case "expansionregion" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawExpansionregion(obj);
         break;
      case "polyline" :

this.setFillAndStroke(obj.strokestyle,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawPolyline(obj.points,
obj.arrowstyle1, obj.arrowstyle2,
obj.linestyle);
         break;
      case "arrowhead" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
"square", "miter");

this.drawArrowheadNotation(obj.x,
obj.y, obj.width, obj.height, obj.rotate);
         break;
      case "activationbar" :
         this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
         this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
         this.drawActivationBar(obj.x,
obj.y, obj.width, obj.height, obj.rotate);
```

```
            break;
        case "lifeline" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawLifeline(obj.x, obj.y,
obj.width, obj.height, obj.rotate);
            break;
        case "deletion" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawDeletion(obj.x, obj.y,
obj.width, obj.height, obj.rotate);
            break;
        case "initialps" :
        case "initialnode" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawFilledCircle(obj.x, obj.y,
obj.width, obj.height, obj.rotate);
            break;
        case "finalstate" :
        case "finalnode" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);

this.drawBorderedFilledCircle(obj);
            break;
        case "shallowhistps" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawShallowHistPS(obj);
            break;
        case "deephistps" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawDeepHistPS(obj);
            break;
        case "blackbar" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawBlackBar(obj.x, obj.y,
obj.width, obj.height, obj.rotate);
            break;
        case "diamond" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawDiamond(obj.x, obj.y,
obj.width, obj.height, obj.rotate);
            break;
        case "pin" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawSmallSquare(obj.x,
obj.y, obj.width, obj.height, obj.rotate);
            break;
        case "flowfinal" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawFlowFinal(obj);
            break;
        case "port" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawSmallSquare(obj.x,
obj.y, obj.width, obj.height, obj.rotate);
            break;
        case "frame" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawFrame(obj);
            break;
        case "package1" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawPackage1(obj);
            break;
        case "package2" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawPackage2(obj);
            break;
        case "component" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawComponent(obj);
            break;
        case "artifact" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawArtifact(obj);
            break;
        case "usecase" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawUseCase(obj);
            break;
        case "actor" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawActor(obj);
            break;
        case "superstate" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawSuperstate(obj);
            break;
        case "action" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawAction(obj);
            break;
        case "subactivity" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawSubactivity(obj);
            break;
        case "timesignal" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawTimeSignal(obj);
            break;
        case "acceptsignal" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawAcceptSignal(obj);
            break;
        case "sendsignal" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawSendSignal(obj);
            break;
        case "connector" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawConnector(obj);
            break;
        case "transformation" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
            this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
            this.drawTransformation(obj);
            break;
        case "part" :
            this.setFillAndStroke(obj.fillcolor,
obj.strokestyle);
```

```
        this.setLineSettings(obj.linewidth,
obj.linecap, obj.linejoin);
        this.drawPart(obj);
        break;
   }

   if(selectedObj == obj)
this.drawSelectionHandles(obj);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawSelectionHandles = function(obj) {
   this.save();

this.setFillAndStroke(SELECTION_HANDL
E_COLOR, SELECTION_HANDLE_COLOR);
   this.lineWidth =
SELECTION_HANDLE_WIDTH;
   var half = SELECTION_HANDLE_SIZE /
2;

   selectionHandles = [];

   switch(obj.objecttype) {
      case "ellipse" :
      case "rectangle" :
      case "text" :
      case "class" :
      case "note" :
      case "instance" :
      case "node" :
      case "statebox" :
      case "expansionregion" :
      case "arrowhead" :
      case "activationbar" :
      case "lifeline" :
      case "deletion" :
      case "initialps" :
      case "finalstate" :
      case "shallowhistps" :
      case "deephistps" :
      case "initialnode" :
      case "finalnode" :
      case "blackbar" :
      case "diamond" :
      case "pin" :
      case "flowfinal" :
      case "port" :
      case "frame" :
      case "package1" :
      case "package2" :
      case "component" :
      case "artifact" :
      case "usecase" :
      case "actor" :
      case "superstate" :
      case "action" :
      case "subactivity" :
      case "timesignal" :
      case "acceptsignal" :
      case "sendsignal" :
      case "connector" :
      case "transformation" :
      case "part" :
         // Set up the selection handle
boxes
         var noOfSelectionHandles = 8;
         if(obj.objecttype == "node")
            noOfSelectionHandles = 9;
         else if(obj.objecttype ==
"expansionregion")
            noOfSelectionHandles = 10;

         for (i = 0; i <
noOfSelectionHandles; i ++) {
            var handle = {"x" : 0, "y" : 0};
            selectionHandles.push(handle);
         }

         this.translate(obj.x, obj.y);
         this.rotate((obj.rotate / 180) *
3.1416);
         this.strokeRect(0, 0, obj.width,
obj.height);

         selectionHandles[0].x = -half;
         selectionHandles[0].y = -half;
         selectionHandles[1].x = obj.width
/ 2 - half;
         selectionHandles[1].y = -half;
         selectionHandles[2].x = obj.width
- half;
         selectionHandles[2].y = -half;
         selectionHandles[3].x = -half;
         selectionHandles[3].y =
obj.height / 2 - half;
         selectionHandles[4].x = obj.width
- half;
         selectionHandles[4].y =
obj.height / 2 - half;
         selectionHandles[6].x = obj.width
/ 2 - half;
         selectionHandles[6].y =
obj.height - half;
         selectionHandles[5].x = -half;
         selectionHandles[5].y =
obj.height - half;
         selectionHandles[7].x = obj.width
- half;
         selectionHandles[7].y =
obj.height - half;
         if(obj.objecttype == "node") {
            selectionHandles[8].x =
obj.width / 2 - half;
            selectionHandles[8].y =
obj.topheight - half;
         } else if(obj.objecttype ==
"expansionregion") {
            selectionHandles[8].x =
obj.listboxpin1x - half;
            selectionHandles[8].y =
obj.listboxpin1y - half;
            selectionHandles[9].x =
obj.listboxpin2x - half;
            selectionHandles[9].y =
obj.listboxpin2y - half;
         }
         break;
      case "line" :
         // Set up the selection handle
boxes
         for (i = 0; i < 2; i ++) {
            var handle = {"x" : 0, "y" : 0};
            selectionHandles.push(handle);
         }

         selectionHandles[0].x = obj.x1 -
half;
         selectionHandles[0].y = obj.y1 -
half;
         selectionHandles[1].x = obj.x2 -
half;
         selectionHandles[1].y = obj.y2 -
half;
         break;
      case "bezier" :
         // Set up the selection handle
boxes
         for (i = 0; i < 4; i ++) {
            var handle = {"x" : 0, "y" : 0};
            selectionHandles.push(handle);
         }

         selectionHandles[0].x = obj.x1 -
half;
         selectionHandles[0].y = obj.y1 -
half;
         selectionHandles[1].x = obj.x2 -
half;
         selectionHandles[1].y = obj.y2 -
half;
         selectionHandles[2].x = obj.ctrl1x
- half;
         selectionHandles[2].y = obj.ctrl1y
- half;
         selectionHandles[3].x = obj.ctrl2x
- half;
         selectionHandles[3].y = obj.ctrl2y
- half;

         this.drawDashedLine(obj.x1,
obj.y1, obj.ctrl1x, obj.ctrl1y, [1,
this.lineWidth + 5]);
         this.drawDashedLine(obj.x2,
obj.y2, obj.ctrl2x, obj.ctrl2y, [1,
this.lineWidth + 5]);
         break;
      case "polygon" :
         // Set up the selection handle
boxes
         var noOfPts = obj.points.length;
         for (i = 0; i < noOfPts; i ++) {
            var handle = {"x" : 0, "y" : 0};
            selectionHandles.push(handle);
            selectionHandles[i].x =
obj.points[i].x - half;
            selectionHandles[i].y =
obj.points[i].y - half;


this.drawDashedLine(obj.points[i].x,
obj.points[i].y,
                  obj.points[(i + 1) %
noOfPts].x, obj.points[(i + 1) %
noOfPts].y,
```

```
            [1, this.lineWidth + 5]);
        }
        break;
      case "polyline" :
        // Set up the selection handle
boxes
        var noOfPts = obj.points.length;
        for (i = 0; i < noOfPts; i ++) {
          var handle = {"x" : 0, "y" : 0};
          selectionHandles.push(handle);
          selectionHandles[i].x =
obj.points[i].x - half;
          selectionHandles[i].y =
obj.points[i].y - half;
        }
        break;
      case "path" :
        if(obj.points.length > 0) {
          for (i = 0; i < 8; i ++) {
            var handle = {"x" :
obj.points[0].x, "y" : obj.points[0].y};

selectionHandles.push(handle);
          }

          for(var i = 1; i <
obj.points.length; i++) {
            if(obj.points[i].x <
selectionHandles[0].x) {
              selectionHandles[0].x =
obj.points[i].x;
              selectionHandles[3].x =
obj.points[i].x;
              selectionHandles[5].x =
obj.points[i].x;
              selectionHandles[1].x =
(obj.points[i].x +
                selectionHandles[2].x) /
2;
              selectionHandles[6].x =
(obj.points[i].x +
                selectionHandles[2].x) /
2;
            } else if(obj.points[i].x >
selectionHandles[2].x) {
              selectionHandles[2].x =
obj.points[i].x;
              selectionHandles[4].x =
obj.points[i].x;
              selectionHandles[7].x =
obj.points[i].x;
              selectionHandles[1].x =
(obj.points[i].x +
                selectionHandles[0].x) /
2;
              selectionHandles[6].x =
(obj.points[i].x +
                selectionHandles[0].x) /
2;
            }

            if(obj.points[i].y <
selectionHandles[0].y) {
              selectionHandles[0].y =
obj.points[i].y;
              selectionHandles[1].y =
obj.points[i].y;
              selectionHandles[2].y =
obj.points[i].y;
              selectionHandles[3].y =
(obj.points[i].y +
                selectionHandles[5].y) /
2;
              selectionHandles[4].y =
(obj.points[i].y +
                selectionHandles[2].y) /
2;
            } else if(obj.points[i].y >
selectionHandles[5].y) {
              selectionHandles[5].y =
obj.points[i].y;
              selectionHandles[6].y =
obj.points[i].y;
              selectionHandles[7].y =
obj.points[i].y;
              selectionHandles[3].y =
(obj.points[i].y +
                selectionHandles[0].y) /
2;
              selectionHandles[4].y =
(obj.points[i].y +
                selectionHandles[0].y) /
2;
            }
          }


this.strokeRect(selectionHandles[0].x,
selectionHandles[0].y,
            selectionHandles[7].x -
selectionHandles[0].x,
            selectionHandles[7].y -
selectionHandles[0].y);

          for (i = 0; i < 8; i ++) {
            selectionHandles[i].x -= half;
            selectionHandles[i].y -= half;
          }

        }
        break;
      default :
        break;
    }

    for (var i = 0; i <
selectionHandles.length; i++) {
      var cur = selectionHandles[i];
      this.fillRect(cur.x, cur.y,
SELECTION_HANDLE_SIZE,
SELECTION_HANDLE_SIZE);
    }

    this.restore();
};

function changeHandleCursor() {
    switch(selectedObj.objecttype) {
      case "rectangle" :
      case "ellipse" :
      case "text" :
      case "class" :
      case "note" :
      case "instance" :
      case "node" :
      case "statebox" :
      case "expansionregion" :
      case "arrowhead" :
      case "activationbar" :
      case "lifeline" :
      case "deletion" :
      case "initialps" :
      case "finalstate" :
      case "shallowhistps" :
      case "deephistps" :
      case "initialnode" :
      case "finalnode" :
      case "blackbar" :
      case "diamond" :
      case "pin" :
      case "flowfinal" :
      case "port" :
      case "frame" :
      case "package1" :
      case "package2" :
      case "component" :
      case "artifact" :
      case "usecase" :
      case "actor" :
      case "superstate" :
      case "action" :
      case "subactivity" :
      case "timesignal" :
      case "acceptsignal" :
      case "sendsignal" :
      case "connector" :
      case "transformation" :
      case "part" :
        //for (var i = 0; i <
selectionHandles.length; i++) {
        for (var i =
selectionHandles.length - 1; i >= 0; i--) {
          var cur = selectionHandles[i];
          var rad = selectedObj.rotate /
180 * Math.PI;
          var itp =
inverseTransform(tool.x0,
selectedObj.x, tool.y0, selectedObj.y,
rad);


tempCanvas.style.cursor='move';

          if (itp[0] >= cur.x && itp[0] <=
cur.x + SELECTION_HANDLE_SIZE &&
              itp[1] >= cur.y && itp[1] <=
cur.y + SELECTION_HANDLE_SIZE) {
            expectResize = i;

            return;
          }
        }
        break;
      case "line" :
      case "bezier" :
      case "polygon" :
      case "polyline" :
```

```
        for (var i = 0; i <
selectionHandles.length; i++) {
            var cur = selectionHandles[i];

            if (tool.x0 >= cur.x && tool.x0
<= cur.x + SELECTION_HANDLE_SIZE &&
                tool.y0 >= cur.y && tool.y0
<= cur.y + SELECTION_HANDLE_SIZE) {
                expectResize = i;

tempCanvas.style.cursor='move';
                return;
            }
          }
          break;
      case "path" :
          break;
   }

   isResizeDrag = false;
   expectResize = -1;
   tempCanvas.style.cursor='auto';
}

function resizeSelectedObj(mx, my) {
   switch(selectedObj.objecttype) {
      case "rectangle" :
      case "ellipse" :
      case "text" :
      case "class" :
      case "note" :
      case "instance" :
      case "node" :
      case "statebox" :
      case "arrowhead" :
      case "activationbar" :
      case "lifeline" :
      case "deletion" :
      case "initialps" :
      case "finalstate" :
      case "shallowhistps" :
      case "deephistps" :
      case "initialnode" :
      case "finalnode" :
      case "blackbar" :
      case "diamond" :
      case "pin" :
      case "flowfinal" :
      case "port" :
      case "frame" :
      case "package1" :
      case "package2" :
      case "component" :
      case "artifact" :
      case "usecase" :
      case "actor" :
      case "superstate" :
      case "action" :
      case "subactivity" :
      case "timesignal" :
      case "acceptsignal" :
      case "sendsignal" :
      case "connector" :
      case "transformation" :
      case "part" :
          var angle = selectedObj.rotate /
180 * Math.PI;
          var delta = inverseTransform(mx,
selectedObj.x, my, selectedObj.y, angle);

          switch(expectResize) {
            case 0 :
              delta[0] = (delta[0] <
selectedObj.width) ? delta[0] :
(selectedObj.width - 1);
              delta[1] = (delta[1] <
selectedObj.height) ? delta[1] :
(selectedObj.height - 1);
              selectedObj.width -=
delta[0];
              selectedObj.height -=
delta[1];
              selectedObj.x += delta[0] *
Math.cos(angle) - delta[1] *
Math.sin(angle);
              selectedObj.y += delta[0] *
Math.sin(angle) + delta[1] *
Math.cos(angle);
              break;
            case 1 :
              delta[1] = (delta[1] <
selectedObj.height) ? delta[1] :
(selectedObj.height - 1);
              selectedObj.height -=
delta[1];
              selectedObj.x -= delta[1] *
Math.sin(angle);
              selectedObj.y += delta[1] *
Math.cos(angle);
              break;
            case 2 :
              delta[0] = (delta[0] >= 1) ?
delta[0] : 1;
              delta[1] = (delta[1] <
selectedObj.height) ? delta[1] :
(selectedObj.height - 1);
              selectedObj.x -= delta[1] *
Math.sin(angle);
              selectedObj.y += delta[1] *
Math.cos(angle);
              selectedObj.width = delta[0];
              selectedObj.height -=
delta[1];
              break;
            case 3 :
              delta[0] = (delta[0] <
selectedObj.width) ? delta[0] :
(selectedObj.width - 1);
              selectedObj.width -=
delta[0];
              selectedObj.x += delta[0] *
Math.cos(angle);
              selectedObj.y += delta[0] *
Math.sin(angle);
              break;
            case 4 :
              delta[0] = (delta[0] >= 1) ?
delta[0] : 1;
              selectedObj.width = delta[0];
              break;
            case 5 :
              delta[0] = (delta[0] <
selectedObj.width) ? delta[0] :
(selectedObj.width - 1);
              delta[1] = (delta[1] >= 1) ?
delta[1] : 1;
              selectedObj.x += delta[0] *
Math.cos(angle);
              selectedObj.y += delta[0] *
Math.sin(angle);
              selectedObj.width -=
delta[0];
              selectedObj.height =
delta[1];
              break;
            case 6 :
              delta[1] = (delta[1] >= 1) ?
delta[1] : 1;
              selectedObj.height =
delta[1];
              break;
            case 7 :
              delta[0] = (delta[0] >= 1) ?
delta[0] : 1;
              delta[1] = (delta[1] >= 1) ?
delta[1] : 1;
              selectedObj.width = delta[0];
              selectedObj.height =
delta[1];
              break;
            case 8 :
              if(selectedObj.objecttype ==
"node") {
                delta[1] = (delta[1] >= 1) ?
delta[1] : 1;
                //delta[1] = (delta[1] <=
selectedObj.height) ? delta[1] :
selectedObj.height;
                selectedObj.topheight =
delta[1];
              }
              break;
          }
          break;
      case "expansionregion" :
          var angle = selectedObj.rotate /
180 * Math.PI;
          var delta = inverseTransform(mx,
selectedObj.x, my, selectedObj.y, angle);

          switch(expectResize) {
            case 0 :
              delta[0] = (delta[0] <
selectedObj.width) ? delta[0] :
(selectedObj.width - 1);
              delta[1] = (delta[1] <
selectedObj.height) ? delta[1] :
(selectedObj.height - 1);
              selectedObj.width -=
delta[0];
              selectedObj.height -=
delta[1];
              selectedObj.x += delta[0] *
Math.cos(angle) - delta[1] *
Math.sin(angle);
```

```
            selectedObj.y += delta[0] *
Math.sin(angle) + delta[1] *
Math.cos(angle);
            break;
          case 1 :
            delta[1] = (delta[1] <
selectedObj.height) ? delta[1] :
(selectedObj.height - 1);
            selectedObj.height -=
delta[1];
            selectedObj.x -= delta[1] *
Math.sin(angle);
            selectedObj.y += delta[1] *
Math.cos(angle);
            break;
          case 2 :
            delta[0] = (delta[0] >= 1) ?
delta[0] : 1;
            delta[1] = (delta[1] <
selectedObj.height) ? delta[1] :
(selectedObj.height - 1);
            selectedObj.x -= delta[1] *
Math.sin(angle);
            selectedObj.y += delta[1] *
Math.cos(angle);
            selectedObj.width = delta[0];
            selectedObj.height -=
delta[1];
            break;
          case 3 :
            delta[0] = (delta[0] <
selectedObj.width) ? delta[0] :
(selectedObj.width - 1);
            selectedObj.width -=
delta[0];
            selectedObj.x += delta[0] *
Math.cos(angle);
            selectedObj.y += delta[0] *
Math.sin(angle);
            break;
          case 4 :
            delta[0] = (delta[0] >= 1) ?
delta[0] : 1;
            selectedObj.width = delta[0];
            break;
          case 5 :
            delta[0] = (delta[0] <
selectedObj.width) ? delta[0] :
(selectedObj.width - 1);
            delta[1] = (delta[1] >= 1) ?
delta[1] : 1;
            selectedObj.x += delta[0] *
Math.cos(angle);
            selectedObj.y += delta[0] *
Math.sin(angle);
            selectedObj.width -=
delta[0];
            selectedObj.height =
delta[1];
            break;
          case 6 :
            delta[1] = (delta[1] >= 1) ?
delta[1] : 1;
            selectedObj.height =
delta[1];
            break;
          case 7 :
            delta[0] = (delta[0] >= 1) ?
delta[0] : 1;
            delta[1] = (delta[1] >= 1) ?
delta[1] : 1;
            selectedObj.width = delta[0];
            selectedObj.height =
delta[1];
            break;
          case 8 :
            var below1 =
((selectedObj.height /
selectedObj.width) * delta[0] >
delta[1]);
            var below2 = ((-
selectedObj.height / selectedObj.width)
* delta[0]
              + selectedObj.height >=
delta[1]);

            if(below1 && below2) {
              delta[0] = (delta[0] >= 0) ?
delta[0] : 0;
              delta[0] = (delta[0] <
selectedObj.width - 4 *
selectedObj.listboxpinsize)
                ? delta[0] :
selectedObj.width - 4 *
selectedObj.listboxpinsize;
              delta[1] = 0;
            } else if(below1 && !below2)
{
              delta[0] =
selectedObj.width;
              delta[1] = (delta[1] >= 0) ?
delta[1] : 0;
              delta[1] = (delta[1] <
selectedObj.height - 4 *
selectedObj.listboxpinsize)
                ? delta[1] :
selectedObj.height - 4 *
selectedObj.listboxpinsize;
            } else if(!below1 &&
!below2) {
              delta[0] = (delta[0] >= 0) ?
delta[0] : 0;
              delta[0] = (delta[0] <
selectedObj.width - 4 *
selectedObj.listboxpinsize)
                ? delta[0] :
selectedObj.width - 4 *
selectedObj.listboxpinsize;
              delta[1] =
selectedObj.height;
            } else {
              delta[0] = 0;
              delta[1] = (delta[1] >= 0) ?
delta[1] : 0;
              delta[1] = (delta[1] <
selectedObj.height - 4 *
selectedObj.listboxpinsize)
                ? delta[1] :
selectedObj.height - 4 *
selectedObj.listboxpinsize;
            }
            selectedObj.listboxpin1x =
delta[0];
            selectedObj.listboxpin1y =
delta[1];

            break;
          case 9 :
            var below1 =
((selectedObj.height /
selectedObj.width) * delta[0] >
delta[1]);
            var below2 = ((-
selectedObj.height / selectedObj.width)
* delta[0]
              + selectedObj.height >=
delta[1]);

            if(below1 && below2) {
              delta[0] = (delta[0] >= 0) ?
delta[0] : 0;
              delta[0] = (delta[0] <
selectedObj.width - 4 *
selectedObj.listboxpinsize)
                ? delta[0] :
selectedObj.width - 4 *
selectedObj.listboxpinsize;
              delta[1] = 0;
            } else if(below1 && !below2)
{
              delta[0] =
selectedObj.width;
              delta[1] = (delta[1] >= 0) ?
delta[1] : 0;
              delta[1] = (delta[1] <
selectedObj.height - 4 *
selectedObj.listboxpinsize)
                ? delta[1] :
selectedObj.height - 4 *
selectedObj.listboxpinsize;
            } else if(!below1 &&
!below2) {
              delta[0] = (delta[0] >= 0) ?
delta[0] : 0;
              delta[0] = (delta[0] <
selectedObj.width - 4 *
selectedObj.listboxpinsize)
                ? delta[0] :
selectedObj.width - 4 *
selectedObj.listboxpinsize;
              delta[1] =
selectedObj.height;
            } else {
              delta[0] = 0;
              delta[1] = (delta[1] >= 0) ?
delta[1] : 0;
              delta[1] = (delta[1] <
selectedObj.height - 4 *
selectedObj.listboxpinsize)
                ? delta[1] :
selectedObj.height - 4 *
selectedObj.listboxpinsize;
            }

            selectedObj.listboxpin2x =
delta[0];
```

```
            selectedObj.listboxpin2y =
delta[1];

            break;
        }
        break;
      case "line" :
        switch(expectResize) {
          case 0 :
            selectedObj.x1 = mx;
            selectedObj.y1 = my;
            break;
          case 1 :
            selectedObj.x2 = mx;
            selectedObj.y2 = my;
            break;
        }
        break;
      case "bezier" :
        switch(expectResize) {
          case 0 :
            selectedObj.x1 = mx;
            selectedObj.y1 = my;
            break;
          case 1 :
            selectedObj.x2 = mx;
            selectedObj.y2 = my;
            break;
          case 2 :
            selectedObj.ctrl1x = mx;
            selectedObj.ctrl1y = my;
            break;
          case 3 :
            selectedObj.ctrl2x = mx;
            selectedObj.ctrl2y = my;
            break;
        }
        break;
      case "polygon" :
      case "polyline" :

selectedObj.points[expectResize].x =
mx;

selectedObj.points[expectResize].y =
my;
        break;
      case "path" :
        break;
    }
}

function transformX(x, y, angle, shift) {
   return x * Math.cos(angle) - y *
Math.sin(angle) + shift;
}

function transformY(x, y, angle, shift) {
   return x * Math.sin(angle) + y *
Math.cos(angle) + shift;
}

function inverseTransform(x, h, y, k,
angle) {
   var dx = (x - h) * Math.cos(angle) + (y -
k) * Math.sin(angle);
   var dy = (y - k) * Math.cos(angle) - (x -
h) * Math.sin(angle);
   return [dx, dy];
}
```

---

diagram shapes 2.js

```
CanvasRenderingContext2D.prototype.d
rawRectangle = function (x, y, w, h, r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   this.strokeRect(0, 0, w, h);
   this.fillRect(0, 0, w, h);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawRoundedRectangle = function (x, y,
w, h, r, angle) {
   this.save();
   this.translate(x, y);
   this.rotate(angle / 180 * Math.PI);

   this.beginPath();
   this.moveTo(r, 0);
   this.lineTo(w - r, 0);
   this.quadraticCurveTo(w, 0, w, r);
   this.lineTo(w, h - r);
   this.quadraticCurveTo(w, h, w - r, h);
   this.lineTo(r, h);
   this.quadraticCurveTo(0, h, 0, h - r);
   this.lineTo(0, r);
   this.quadraticCurveTo(0, 0, r, 0);
   this.closePath();
   this.fill();
   this.stroke();

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawDashedRoundedRect = function(x, y,
w, h, r, angle, dashPattern) {
   this.save();
   this.translate(x, y);
   this.rotate(angle / 180 * Math.PI);

   this.beginPath();
   this.moveTo(r, 0);
   this.lineTo(w - r, 0);
   this.quadraticCurveTo(w, 0, w, r);
   this.lineTo(w, h - r);
   this.quadraticCurveTo(w, h, w - r, h);
   this.lineTo(r, h);
   this.quadraticCurveTo(0, h, 0, h - r);
   this.lineTo(0, r);
   this.quadraticCurveTo(0, 0, r, 0);
   this.closePath();
   this.fill();

   this.drawDashedLine(r, 0, w - r, 0,
dashPattern);
   this.drawDashedLine(w, r, w, h - r,
dashPattern);
   this.drawDashedLine(w - r, h, r, h,
dashPattern);
   this.drawDashedLine(0, h - r, 0, r,
dashPattern);

   this.drawDashedQuadratic([{"x" : w -
r, "y" : 0}, {"x" : w, "y" : 0}, {"x" : w, "y" :
r}], dashPattern);
   this.drawDashedQuadratic([{"x" : w,
"y" : h - r}, {"x" : w, "y" : h}, {"x" : w - r,
"y" : h}], dashPattern);
   this.drawDashedQuadratic([{"x" : r,
"y" : h}, {"x" : 0, "y" : h}, {"x" : 0, "y" : h -
r}], dashPattern);
   this.drawDashedQuadratic([{"x" : 0,
"y" : r}, {"x" : 0, "y" : 0}, {"x" : r, "y" : 0}],
dashPattern);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawDashedQuadratic = function(points,
dashPattern) {
   var step = 0.005;

   // Possibly gratuitous helper functions
   var delta = function(p0, p1) {
      return [p1[0] - p0[0], p1[1] - p0[1]];
   };
   var arcLength = function(p0, p1) {
      var d = delta(p0, p1);
      return Math.sqrt(d[0]*d[0] + d[1] *
d[1]);
   };
   var getPt = function(t) {
      var pt =
jsBezier.pointOnCurve(points, t);
      return [pt.x, pt.y];
   };

   var subPaths = [];
   var loc = getPt(0);
   var lastLoc = loc;

   var dashIndex = 0;
   var length = 0;
   var thisPath = [];
   for(var t = step; t <= 1; t += step) {
      loc = getPt(t);
      length += arcLength(lastLoc, loc);
      lastLoc = loc;

      //detect when we come to the end
of a dash or space
      if(length >=
dashPattern[dashIndex]) {

        //if we are on a dash, we need to
record the path.
        if(dashIndex % 2 == 0)
          subPaths.push(thisPath);
```

```
        //go to the next dash or space in
the pattern
        dashIndex = (dashIndex + 1) %
dashPattern.length;

        //clear the arclength and path.
        thisPath = [];
        length = 0;
       }

      //if we are on a dash and not a
space, add a point to the path.
      if(dashIndex % 2 == 0) {
       thisPath.push(loc[0], loc[1]);
      }
   }
   if(thisPath.length > 0)
   subPaths.push(thisPath);

   // Now draw subpaths on the canvas
   this.beginPath();
   for(var i = 0; i < subPaths.length; i++) {
      var part = subPaths[i];
      if(part.length > 0)
       this.moveTo(part[0], part[1]);
      for(var j = 1; j < part.length / 2; j++)
{
       this.lineTo(part[2*j], part[2*j+1]);
      }
   }
   this.stroke();
}

CanvasRenderingContext2D.prototype.d
rawEllipse = function (x, y, width, height,
r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   var kappa = .5522848,
      ox = (width / 2) * kappa, // control
point offset horizontal
      oy = (height / 2) * kappa, // control
point offset vertical
      xe = width,          // x-end
      ye = height,         // y-end
      xm = width / 2,      // x-middle
      ym = height / 2;     // y-middle

   this.beginPath();
   this.moveTo(0, ym);
   this.bezierCurveTo(0, ym - oy, xm - ox,
0, xm, 0);
   this.bezierCurveTo(xm + ox, 0, xe, ym
- oy, xe, ym);
   this.bezierCurveTo(xe, ym + oy, xm +
ox, ye, xm, ye);
   this.bezierCurveTo(xm - ox, ye, 0, ym
+ oy, 0, ym);
   this.closePath();
   this.stroke();
   this.fill();

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawText = function (textObj) {
   this.save();
   this.translate(textObj.x, textObj.y);
   this.rotate(textObj.rotate / 180 *
Math.PI);

   this.textBaseline = "top";
   this.font = textObj.fontstyle + " " +
textObj.fontweight + " " +
textObj.fontsize
      + "px " + textObj.fontfamily;
   this.textAlign = "left";

   this.drawWrappedText(textObj.width,
textObj.height - textObj.fontsize,
textObj.label,
      textObj.fontsize, 0, -
textObj.fontsize);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawSolidLine = function (x1, y1, x2, y2) {
   this.beginPath();
   this.moveTo(x1, y1);
   this.lineTo(x2, y2);
   this.stroke();
}

CanvasRenderingContext2D.prototype.d
rawDashedLine = function (fromX,
fromY, toX, toY, pattern) {
   this.beginPath();

   var lt = function (a, b) {return a <= b;};
   var gt = function (a, b) {return a >=
b;};
   var capmin = function (a, b) {return
Math.min(a, b);};
   var capmax = function (a, b) {return
Math.max(a, b);};

   var checkX = {thereYet: gt, cap:
capmin};
   var checkY = {thereYet: gt, cap:
capmin};

   if (fromY - toY > 0) {
      checkY.thereYet = lt;
      checkY.cap = capmax;
   }
   if (fromX - toX > 0) {
      checkX.thereYet = lt;
      checkX.cap = capmax;
   }

   this.moveTo(fromX, fromY);
   var offsetX = fromX;
   var offsetY = fromY;
   var idx = 0, dash = true;
   while (!(checkX.thereYet(offsetX, toX)
&& checkY.thereYet(offsetY, toY))) {
      var ang = Math.atan2(toY - fromY,
toX - fromX);
      var len = pattern[idx];

      offsetX = checkX.cap(toX, offsetX +
(Math.cos(ang) * len));
      offsetY = checkY.cap(toY, offsetY +
(Math.sin(ang) * len));

      if (dash) this.lineTo(offsetX,
offsetY);
      else this.moveTo(offsetX, offsetY);

      idx = (idx + 1) % pattern.length;
      dash = !dash;
   }

   this.stroke();
};

CanvasRenderingContext2D.prototype.d
rawSolidBezier = function(x1, y1, x2, y2,
ctrl1x, ctrl1y, ctrl2x, ctrl2y) {
   this.beginPath();
   this.moveTo(x1, y1);
   this.bezierCurveTo(ctrl1x, ctrl1y,
ctrl2x, ctrl2y, x2, y2);
   this.stroke();
}

CanvasRenderingContext2D.prototype.d
rawDashedBezier = function(x1, y1, x2,
y2, ctrl1x, ctrl1y, ctrl2x, ctrl2y,
dashPattern) {
   var points = [ {x : x1, y : y1}, {x : ctrl1x,
y : ctrl1y}, {x : ctrl2x, y : ctrl2y}, {x : x2, y
: y2} ];

   var step = 0.0005;  //this really should
be set by an intelligent method,
               //rather than using a
constant, but it serves as an
               //example.

   // Possibly gratuitous helper functions
   var delta = function(p0, p1) {
      return [p1[0] - p0[0], p1[1] - p0[1]];
   };
   var arcLength = function(p0, p1) {
      var d = delta(p0, p1);
      return Math.sqrt(d[0]*d[0] + d[1] *
d[1]);
   };
   var getBezierPt = function(inputCurve,
t) {
      var pt =
jsBezier.pointOnCurve(inputCurve, t);
      return [pt.x, pt.y];
   };

   var subPaths = [];
   var loc = getBezierPt(points, 0);
   var lastLoc = loc;

   var dashIndex = 0;
   var length = 0;
```

```
   var thisPath = [];
   for(var t = step; t <= 1; t += step) {
      loc = getBezierPt(points, t);
      length += arcLength(lastLoc, loc);
      lastLoc = loc;

      //detect when we come to the end
of a dash or space
      if(length >=
dashPattern[dashIndex]) {

        //if we are on a dash, we need to
record the path.
        if(dashIndex % 2 == 0)
          subPaths.push(thisPath);

        //go to the next dash or space in
the pattern
        dashIndex = (dashIndex + 1) %
dashPattern.length;

        //clear the arclength and path.
        thisPath = [];
        length = 0;
      }

      //if we are on a dash and not a
space, add a point to the path.
      if(dashIndex % 2 == 0) {
        thisPath.push(loc[0], loc[1]);
      }
   }
   if(thisPath.length > 0)
   subPaths.push(thisPath);

   // Now draw subpaths on the canvas
   this.beginPath();
   for(var i = 0; i < subPaths.length; i++) {
      var part = subPaths[i];
      if(part.length > 0)
        this.moveTo(part[0], part[1]);
      for(var j = 1; j < part.length / 2; j++)
{
        this.lineTo(part[2*j], part[2*j+1]);
      }
   }
   this.stroke();
}

CanvasRenderingContext2D.prototype.d
rawBezier = function(x1, y1, x2, y2,
ctrl1x, ctrl1y, ctrl2x, ctrl2y, ah1, ah2,
linestyle) {
   var lineangle1 = Math.atan2(ctrl1y -
y1, ctrl1x - x1);
   var lineangle2 = Math.atan2(ctrl2y -
y2, ctrl2x - x2);

   var x12 = x1;
   var y12 = y1;
   var x22 = x2;
   var y22 = y2;

   switch(ah1) {
      case "lines" :
      case "hollow triangle" :
      case "filled triangle" :
         x12 += ARROWHEAD_L *
Math.cos(lineangle1);
         y12 += ARROWHEAD_L *
Math.sin(lineangle1);
         break;
      case "hollow diamond" :
      case "filled diamond" :
         x12 += 2 * DIAMOND_L *
Math.cos(lineangle1);
         y12 += 2 * DIAMOND_L *
Math.sin(lineangle1);
         break;
      case "ball" :
         x12 += 2 * BALL_R *
Math.cos(lineangle1);
         y12 += 2 * BALL_R *
Math.sin(lineangle1);
         break;
      case "socket" :
            x12 += SOCKET_R *
Math.cos(lineangle1);
            y12 += SOCKET_R *
Math.sin(lineangle1);
            break;
      default :
         break;
   }

   switch(ah2) {
      case "lines" :
      case "hollow triangle" :
      case "filled triangle" :
         x22 -= ARROWHEAD_L *
Math.cos(lineangle2 + Math.PI);
         y22 -= ARROWHEAD_L *
Math.sin(lineangle2 + Math.PI);
         break;
      case "hollow diamond" :
      case "filled diamond" :
         x22 -= 2 * DIAMOND_L *
Math.cos(lineangle2 + Math.PI);
         y22 -= 2 * DIAMOND_L *
Math.sin(lineangle2 + Math.PI);
         break;
      case "ball" :
         x22 -= 2 * BALL_R *
Math.cos(lineangle2 + Math.PI);
         y22 -= 2 * BALL_R *
Math.sin(lineangle2 + Math.PI);
         break;
      case "socket" :
         x22 -= SOCKET_R *
Math.cos(lineangle2 + Math.PI);
         y22 -= SOCKET_R *
Math.sin(lineangle2 + Math.PI);
         break;
      default :
         break;
   }

   if(linestyle == "dashed")
      this.drawDashedBezier(x12, y12,
x22, y22, ctrl1x, ctrl1y, ctrl2x, ctrl2y, [7,
this.lineWidth + 7]);
   else if(linestyle == "dotted")
      this.drawDashedBezier(x12, y12,
x22, y22, ctrl1x, ctrl1y, ctrl2x, ctrl2y, [2,
this.lineWidth + 5]);
   else
      this.drawSolidBezier(x12, y12, x22,
y22, ctrl1x, ctrl1y, ctrl2x, ctrl2y);

   this.drawArrowhead(ah1, x1, y1,
lineangle1);
   this.drawArrowhead(ah2, x2, y2,
lineangle2);
}

CanvasRenderingContext2D.prototype.d
rawLine = function (x1, y1, x2, y2, ah1,
ah2, linestyle) {
   var lineangle = Math.atan2(y2 - y1, x2
- x1);

   var x12 = x1;
   var y12 = y1;
   var x22 = x2;
   var y22 = y2;

   if(ah1 == "socket") {
      x12 += SOCKET_R *
Math.cos(lineangle);
      y12 += SOCKET_R *
Math.sin(lineangle);
   }
   if(ah2 == "socket") {
      x22 -= SOCKET_R *
Math.cos(lineangle);
      y22 -= SOCKET_R *
Math.sin(lineangle);
   }

   if(linestyle == "dashed")
      this.drawDashedLine(x12, y12, x22,
y22, [7, this.lineWidth + 7]);
   else if(linestyle == "dotted")
      this.drawDashedLine(x12, y12, x22,
y22, [1, this.lineWidth + 5]);
   else
      this.drawSolidLine(x12, y12, x22,
y22);

   this.drawArrowhead(ah1, x1, y1,
lineangle);
   this.drawArrowhead(ah2, x2, y2,
lineangle + Math.PI);
};

CanvasRenderingContext2D.prototype.d
rawArrowhead = function (arrowstyle, x,
y, lineangle) {
   switch(arrowstyle) {
      case "lines" :
         var angle1 = lineangle +
ARROWHEAD_ANGLE;
         var topx = x + Math.cos(angle1) *
ARROWHEAD_H;
         var topy = y + Math.sin(angle1) *
ARROWHEAD_H;
```

```
        var midx = x +
Math.cos(lineangle) * ARROWHEAD_H;
        var midy = y + Math.sin(lineangle)
* ARROWHEAD_H;

        var angle2 = lineangle -
ARROWHEAD_ANGLE;
        var botx = x + Math.cos(angle2) *
ARROWHEAD_H;
        var boty = y + Math.sin(angle2) *
ARROWHEAD_H;

        this.beginPath();
        this.moveTo(topx, topy);
        this.lineTo(x, y);
        this.lineTo(botx, boty);
        this.moveTo(midx, midy);
        this.lineTo(x, y);
        this.stroke();

        break;
      case "hollow triangle" :
      case "filled triangle" :
        var angle1 = lineangle +
ARROWHEAD_ANGLE;
        var topx = x + Math.cos(angle1) *
ARROWHEAD_H;
        var topy = y + Math.sin(angle1) *
ARROWHEAD_H;

        var angle2 = lineangle -
ARROWHEAD_ANGLE;
        var botx = x + Math.cos(angle2) *
ARROWHEAD_H;
        var boty = y + Math.sin(angle2) *
ARROWHEAD_H;

        this.beginPath();
        this.moveTo(topx, topy);
        this.lineTo(x, y);
        this.lineTo(botx, boty);

        if(arrowstyle == "hollow
triangle") {
          this.save();
          this.fillStyle =
"rgba(255,255,255,1)";
          this.closePath();
          this.fill();
          this.restore();
        } else if(arrowstyle == "filled
triangle") {
          this.closePath();
          this.fill();
        }

        this.stroke();

        break;
      case "half head" :
        var angle1 = lineangle +
ARROWHEAD_ANGLE;
        var topx = x + Math.cos(angle1) *
ARROWHEAD_H;
        var topy = y + Math.sin(angle1) *
ARROWHEAD_H;

        this.beginPath();
        this.moveTo(topx, topy);
        this.lineTo(x, y);

        this.stroke();
        break;
      case "hollow diamond" :
      case "filled diamond" :
        var angle1 = lineangle +
DIAMOND_ANGLE;//ARROWHEAD_ANG
LE;
        var topx = x + Math.cos(angle1) *
ARROWHEAD_H;
        var topy = y + Math.sin(angle1) *
ARROWHEAD_H;

        var angle2 = lineangle -
DIAMOND_ANGLE;//ARROWHEAD_ANG
LE;
        var botx = x + Math.cos(angle2) *
ARROWHEAD_H;
        var boty = y + Math.sin(angle2) *
ARROWHEAD_H;

        this.beginPath();
        this.moveTo(topx, topy);
        this.lineTo(x, y);
        this.lineTo(botx, boty);
        this.lineTo(x + 2 * DIAMOND_L *
Math.cos(lineangle), y + 2 *
DIAMOND_L * Math.sin(lineangle));

        this.closePath();

        if(arrowstyle == "filled diamond")
{
          this.fill();
        } else if(arrowstyle == "hollow
diamond") {
          this.save();
          this.fillStyle =
"rgba(255,255,255,1)";
          this.fill();
          this.restore();
        }

        this.stroke();
        break;
      case "ball" :
        this.beginPath();
        this.arc(x + BALL_R *
Math.cos(lineangle), y + BALL_R  *
Math.sin(lineangle),
          BALL_R, 0, 2 * Math.PI);
        this.closePath();
        this.save();
        this.fillStyle =
"rgba(255,255,255,1)";
        this.fill();
        this.restore();
        this.stroke();
        break;
      case "socket" :
        this.beginPath();
        this.arc(x, y, SOCKET_R, lineangle
- Math.PI / 2, lineangle + Math.PI / 2);
        this.stroke();
        break;
      case "none" :
      default :break;
    }
}

CanvasRenderingContext2D.prototype.d
rawPath = function (points) {
  this.beginPath();
  this.moveTo(points[0].x, points[0].y);
  for(var i = 0; i < points.length; i++)
    this.lineTo(points[i].x, points[i].y);
  this.stroke();
};

CanvasRenderingContext2D.prototype.d
rawPolygon = function (points) {
  this.beginPath();
  this.moveTo(points[0].x, points[0].y);
  for(var i = 0; i < points.length; i++)
    this.lineTo(points[i].x, points[i].y);
  this.closePath();
  this.stroke();
  this.fill();
};

CanvasRenderingContext2D.prototype.d
rawPolyline = function (points, ah1, ah2,
linestyle) {
  var noOfPts = points.length;

  if(noOfPts < 2) return;

  var x0 = points[0].x;
  var y0 = points[0].y;
  var lineangle1 =
Math.atan2(points[1].y - points[0].y,
points[1].x - points[0].x);
  if(ah1 == "socket") {
    x0 += SOCKET_R *
Math.cos(lineangle1);
    y0 += SOCKET_R *
Math.sin(lineangle1);
  }

  var xn = points[noOfPts - 1].x;
  var yn = points[noOfPts - 1].y;
  var lineangle2 =
Math.atan2(points[noOfPts - 1].y -
points[noOfPts - 2].y,
    points[noOfPts - 1].x -
points[noOfPts - 2].x);
  if(ah2 == "socket") {
    xn -= SOCKET_R *
Math.cos(lineangle2);
    yn -= SOCKET_R *
Math.sin(lineangle2);
  }

  if(linestyle == "dashed") {
    for(var i = 0; i < noOfPts - 1; i++) {
      var x1 = points[i].x;
      var y1 = points[i].y;
```

```
      var x2 = points[i + 1].x;
      var y2 = points[i + 1].y;

      if(i == 0) {
        x1 = x0;
        y1 = y0;
      }
      if(i + 2 == noOfPts) {
        x2 = xn;
        y2 = yn;
      }

      this.drawDashedLine(x1, y1, x2,
y2, [7, this.lineWidth + 7]);
    }
  } else if(linestyle == "dotted") {
    for(var i = 0; i < noOfPts - 1; i++) {
      var x1 = points[i].x;
      var y1 = points[i].y;
      var x2 = points[i + 1].x;
      var y2 = points[i + 1].y;

      if(i == 0) {
        x1 = x0;
        y1 = y0;
      }
      if(i + 2 == noOfPts) {
        x2 = xn;
        y2 = yn;
      }

      this.drawDashedLine(x1, y1, x2,
y2, [1, this.lineWidth + 5]);
    }
  } else {
    for(var i = 0; i < noOfPts - 1; i++) {
      var x1 = points[i].x;
      var y1 = points[i].y;
      var x2 = points[i + 1].x;
      var y2 = points[i + 1].y;

      if(i == 0) {
        x1 = x0;
        y1 = y0;
      }
      if(i + 2 == noOfPts) {
        x2 = xn;
        y2 = yn;
      }

      this.drawSolidLine(x1, y1, x2, y2);
    }
  }

  this.drawArrowhead(ah1, points[0].x,
points[0].y, lineangle1);
  this.drawArrowhead(ah2,
points[noOfPts - 1].x, points[noOfPts -
1].y, lineangle2 + Math.PI);
};

CanvasRenderingContext2D.prototype.d
rawWrappedText = function(maxwidth,
maxheight, text, yjump, x, y) {
  var textline = text.split("\n");

  for(var i = 0; i < textline.length; i++) {

if(this.measureText(textline[i]).width <=
maxwidth) {
      y += yjump;
      if(y > maxheight) return;
      this.fillText(textline[i], x, y);
    } else {
      var startIndex = 0;
      var currIndex = 0;
      var lastSpaceIndex = -1;
      while(currIndex <=
textline[i].length) {

if(this.measureText(textline[i].substring(
startIndex, currIndex)).width >
maxwidth) {
          y += yjump;
          if(y > maxheight) return;

          if(lastSpaceIndex == -1) {

if(this.measureText(textline[i].substring(
startIndex, currIndex - 1)).width <=
maxwidth) {

this.fillText(textline[i].substring(startInd
ex, currIndex - 1), x, y);
            }
            startIndex = currIndex - 1;
          } else {

if(this.measureText(textline[i].substring(
startIndex, lastSpaceIndex)).width <=
maxwidth) {

this.fillText(textline[i].substring(startInd
ex, lastSpaceIndex + 1), x, y);
            }
            startIndex =
lastSpaceIndex + 1;
            lastSpaceIndex = -1;
          }
        }

        if(textline[i].charAt(currIndex)
== " ") {
          lastSpaceIndex = currIndex;
        }

        currIndex++;
      }

if(textline[i].substring(startIndex).length
!= 0) {

if(this.measureText(textline[i].substring(
startIndex)).width < maxwidth) {
          y += yjump;
          if(y > maxheight) return;

if(this.measureText(textline[i].substring(
startIndex)).width <= maxwidth) {

this.fillText(textline[i].substring(startInd
ex), x, y);
          }
        }
      }
    }
  }
}

CanvasRenderingContext2D.prototype.g
etWrappedTextHeight =
function(maxwidth, maxheight, text,
yjump, x, y) {
  var textline = text.split("\n");

  var totalHeight = 0;

  for(var i = 0; i < textline.length; i++) {

if(this.measureText(textline[i]).width <=
maxwidth) {
      y += yjump;
      if(y > maxheight) return
totalHeight;
      totalHeight += yjump;
    } else {
      var startIndex = 0;
      var currIndex = 0;
      var lastSpaceIndex = -1;
      while(currIndex <=
textline[i].length) {

if(this.measureText(textline[i].substring(
startIndex, currIndex)).width >
maxwidth) {
          y += yjump;
          if(y > maxheight) return
totalHeight;

          if(lastSpaceIndex == -1) {

if(this.measureText(textline[i].substring(
startIndex, currIndex - 1)).width <=
maxwidth) {
              totalHeight += yjump;
            }
            startIndex = currIndex - 1;
          } else {

if(this.measureText(textline[i].substring(
startIndex, lastSpaceIndex)).width <=
maxwidth) {
              totalHeight += yjump;
            }
            startIndex =
lastSpaceIndex + 1;
            lastSpaceIndex = -1;
          }
        }

        if(textline[i].charAt(currIndex)
== " ") {
          lastSpaceIndex = currIndex;
        }

        currIndex++;
      }
```

```
if(textline[i].substring(startIndex).length
!= 0) {

if(this.measureText(textline[i].substring(
startIndex)).width < maxwidth) {
            y += yjump;
            if(y > maxheight) return
totalHeight;

if(this.measureText(textline[i].substring(
startIndex)).width <= maxwidth) {
              totalHeight += yjump;
            }
          }
        }
      }
  }

  return totalHeight;
}
```

---

diagram shapes 3.js

```
CanvasRenderingContext2D.prototype.d
rawClass = function (classObj) {
  this.save();
  this.translate(classObj.x, classObj.y);
  this.rotate(classObj.rotate / 180 *
Math.PI);

  var attLen = classObj.showattributes ?
classObj.attributes.length : 0;
  var opLen = classObj.showoperations
? classObj.operations.length : 0;
  var tempLen = classObj.templateclass
? classObj.templates.length : 0;
  var hasStereotype =
(classObj.stereotype != "");
  var V_MARGIN = (classObj.height -
classObj.fontsize * (attLen + opLen +
    (hasStereotype ? 1 : 0)) -
classObj.classfontsize - 3 *
classObj.linewidth) /
    (attLen + opLen + 4);
  V_MARGIN = (V_MARGIN < 0) ? 0 :
V_MARGIN;
  var V_INC = V_MARGIN +
classObj.fontsize;

  var yTemp0;
  if(!(classObj.showattributes ||
classObj.showoperations) ||
      classObj.activeclass ||
classObj.qualifiedassociation) {
    yTemp0 = 0.5 * (classObj.height +
classObj.classfontsize);
  } else {
    yTemp0 = classObj.fontsize *
(hasStereotype ? 1 : 0) + V_MARGIN +
classObj.classfontsize;
  }

  var yTemp1 = yTemp0 + V_MARGIN +
classObj.linewidth;
  var yTemp2 = yTemp1 + attLen *
V_INC + V_MARGIN +
classObj.linewidth;
  var i;

  this.fillRect(0, 0, classObj.width,
classObj.height);
  this.strokeRect(0, 0, classObj.width,
classObj.height);


this.setFillAndStroke(classObj.textcolor,
classObj.textcolor);
  this.textBaseline = "bottom";
  this.textAlign = "center";

  // Draw class name
  this.font = (classObj.abstractclass ?
"italic" : "normal") + " bold " +
    classObj.classfontsize + "px sans-
serif";
  //this.fillText(classObj.classname, 0.5
* classObj.width, yTemp0);

this.drawWrappedText(classObj.width,
Math.min(yTemp0, classObj.height),
    classObj.classname,
classObj.classfontsize, 0.5 *
classObj.width,
    yTemp0 - classObj.classfontsize);


  this.font = "normal normal " +
classObj.fontsize + "px sans-serif";
  if(hasStereotype) {

this.drawWrappedText(classObj.width,
Math.min(yTemp0 -
classObj.classfontsize, classObj.height),
      "<<" + classObj.stereotype + ">>",
classObj.fontsize, 0.5 * classObj.width,
      yTemp0 - classObj.classfontsize -
classObj.fontsize);
    //this.fillText("<<" +
classObj.stereotype + ">>", 0.5 *
classObj.width,
    //   yTemp0 -
classObj.classfontsize);
  }

  if(classObj.activeclass) {

this.setFillAndStroke(classObj.fillcolor,
classObj.strokestyle);
    this.drawSolidLine(H_MARGIN, 0,
H_MARGIN, classObj.height);
    this.drawSolidLine(classObj.width -
H_MARGIN, 0, classObj.width -
H_MARGIN, classObj.height);

    if(!classObj.qualifiedassociation) {
      this.restore();
      return;
    }
  }

  if(classObj.qualifiedassociation) {

this.setFillAndStroke(classObj.fillcolor,
classObj.strokestyle);
    this.fillRect(classObj.width, 0.25 *
classObj.height, 0.75 * classObj.width,
0.5 * classObj.height);
    this.strokeRect(classObj.width, 0.25
* classObj.height, 0.75 * classObj.width,
0.5 * classObj.height);


this.setFillAndStroke(classObj.textcolor,
classObj.strokestyle);
    this.drawWrappedText(0.75 *
classObj.width, 0.75 * classObj.height,
      classObj.qualifier,
classObj.fontsize, classObj.width + 0.375
*
      classObj.width, 0.5 *
(classObj.height - classObj.fontsize));
    //this.fillText(classObj.qualifier,
classObj.width + 0.375 * classObj.width,
    //   0.5 * (classObj.height +
classObj.fontsize));

    this.restore();
    return;
  }

  if(classObj.showattributes ||
classObj.showoperations) {

this.setFillAndStroke(classObj.fillcolor,
classObj.strokestyle);
    if(yTemp1 - classObj.linewidth <
classObj.height)
      this.drawSolidLine(0, yTemp1 -
classObj.linewidth, classObj.width,
yTemp1 - classObj.linewidth);
    if(yTemp2 - classObj.linewidth <
classObj.height)
      this.drawSolidLine(0, yTemp2 -
classObj.linewidth, classObj.width,
yTemp2 - classObj.linewidth);
  }


this.setFillAndStroke(classObj.textcolor,
classObj.textcolor);
  this.textAlign = "start";

  // Draw attributes
  this.font = "normal normal " +
classObj.fontsize + "px sans-serif";
  i = -1;
  while(++i < attLen) {
    yTemp1 += V_INC;

this.drawWrappedText(classObj.width -
H_MARGIN, Math.min(yTemp1,
classObj.height),
      classObj.attributes[i].value,
classObj.fontsize, H_MARGIN,
      yTemp1 - classObj.fontsize);
```

```
//this.fillText(classObj.attributes[i].value
, H_MARGIN, yTemp1);
      if(classObj.attributes[i].isstatic &&
classObj.width > H_MARGIN &&
yTemp1 <= classObj.height) {
         var attrWidth =
Math.min(classObj.width, H_MARGIN +

this.measureText(classObj.attributes[i].v
alue).width);
         this.drawSolidLine(H_MARGIN,
yTemp1, attrWidth, yTemp1);
      }
   }

   // Draw operations
   i = -1;
   while(++i < opLen) {
      if(classObj.operations[i].isabstract)
{
         this.font = "italic normal " +
classObj.fontsize + "px sans-serif";
      } else {
         this.font = "normal normal " +
classObj.fontsize + "px sans-serif";
      }
      yTemp2 += V_INC;

this.drawWrappedText(classObj.width -
H_MARGIN, Math.min(yTemp2,
classObj.height),
         classObj.operations[i].value,
classObj.fontsize, H_MARGIN,
         yTemp2 - classObj.fontsize);

//this.fillText(classObj.operations[i].valu
e, H_MARGIN, yTemp2);
      if(classObj.operations[i].isstatic &&
classObj.width > H_MARGIN &&
yTemp2 <= classObj.height) {
         var opWidth =
Math.min(classObj.width, H_MARGIN +

this.measureText(classObj.operations[i].
value).width);
         this.drawSolidLine(H_MARGIN,
yTemp2, opWidth, yTemp2);
      }
   }

   // For the template classes
   this.font = "normal normal " +
classObj.fontsize + "px sans-serif";
   if(classObj.templateclass) {
      var tV_INC = 0;
      var templateWidth = 0;
      var tx1 = 0;
      var ty1 = 0;
      var tx2 = 0;
      var ty2 = 0;

      tV_INC = H_MARGIN +
classObj.fontsize;
      var templateHeight = tV_INC *
classObj.templates.length + H_MARGIN;
      var temp;
      for(i = 0; i <
classObj.templates.length; i++) {
         temp =
this.measureText(classObj.templates[i]).
width;
         if(temp > templateWidth)
templateWidth = temp;
      }
      templateWidth += 2 * H_MARGIN;

      if((classObj.width -
this.measureText(classObj.classname).w
idth) < templateWidth) {
         ty2 = 0.75 * V_MARGIN;
         if (classObj.width < 1.5 *
templateWidth)
            tx1 = 0.5 * classObj.width;
         else
            tx1 = classObj.width - 0.75 *
templateWidth;
      } else {
         if ((classObj.width -
this.measureText(classObj.classname).w
idth) < 1.5 * templateWidth)
            tx1 = 0.5 * classObj.width +
H_MARGIN + 0.5 *
this.measureText(classObj.classname).w
idth;
         else
            tx1 = classObj.width - 0.75 *
templateWidth;
         if((classObj.fontsize + 2 *
(V_MARGIN + classObj.linewidth)) < 1.5
* templateHeight)
            ty2 = 0.5 * classObj.fontsize +
V_MARGIN;
         else
            ty2 = 0.75 * templateHeight;
      }

      ty1 = ty2 - templateHeight;
      tx2 = tx1 + templateWidth;


this.setFillAndStroke(classObj.fillcolor,
classObj.strokestyle);
      this.fillRect(tx1, ty1,
templateWidth, templateHeight);
      this.drawDashedLine(tx1, ty1, tx1,
ty2, [5,5]);
      this.drawDashedLine(tx1, ty1, tx2,
ty1, [5,5]);
      this.drawDashedLine(tx2, ty1, tx2,
ty2, [5,5]);
      this.drawDashedLine(tx1, ty2, tx2,
ty2, [5,5]);

      // Draw templates

this.setFillAndStroke(classObj.textcolor,
classObj.strokestyle);
      var yTemp3 = ty1;
      i = -1;
      while(++i < tempLen) {
         yTemp3 += tV_INC;
         this.textAlign = "center";

this.fillText(classObj.templates[i].value,
tx1 + 0.5 * templateWidth, yTemp3);
      }
   }

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawNote = function (noteObj) {
   this.save();
   this.translate(noteObj.x, noteObj.y);
   this.rotate(noteObj.rotate / 180 *
Math.PI);

   var foldlen = (noteObj.height >
noteObj.width) ? noteObj.width :
noteObj.height;
   foldlen *= 0.25;

   var x1 = 0;
   var x2 = x1 + noteObj.width;
   var x3 = x2 - foldlen;
   var y1 = 0;
   var y2 = y1 + noteObj.height;
   var y3 = y1 + foldlen;

   this.beginPath();
   this.moveTo(x3, y1);
   this.lineTo(x1, y1);
   this.lineTo(x1, y2);
   this.lineTo(x2, y2);
   this.lineTo(x2, y3);
   this.lineTo(x3, y1);
   this.fill();
   this.moveTo(x3, y1);
   this.lineTo(x3, y3);
   this.lineTo(x2, y3);
   this.stroke();

   // Write the textual content
   this.textBaseline = "top";
   this.font = noteObj.fontstyle + " " +
noteObj.fontweight + " " +
noteObj.fontsize
      + "px " + noteObj.fontfamily;

this.setFillAndStroke(noteObj.textcolor,
noteObj.strokestyle);
   var maxwidth = noteObj.width - 2 *
H_MARGIN;
   var maxheight = y2 - H_MARGIN -
noteObj.fontsize;
   this.drawWrappedText(maxwidth,
maxheight, noteObj.label,
noteObj.fontsize, x1 + H_MARGIN, y3);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawInstance = function (instanceObj) {
   this.save();
```

```
   this.translate(instanceObj.x,
instanceObj.y);
   this.rotate(instanceObj.rotate / 180 *
Math.PI);

   var attLen =
instanceObj.showattributes ?
instanceObj.attributes.length : 0;
   var V_MARGIN = (instanceObj.height -
instanceObj.fontsize * (attLen + 1)) /
(attLen + 3);
   V_MARGIN = (V_MARGIN < 0) ? 0 :
V_MARGIN;
   var V_INC = V_MARGIN +
instanceObj.fontsize;

   var yTemp0;
   if(!instanceObj.showattributes) {
      yTemp0 = 0.5 * (instanceObj.height
+ instanceObj.fontsize);
   } else {
      yTemp0 = V_MARGIN +
instanceObj.fontsize;
   }

   var yTemp1 = yTemp0 + V_MARGIN;

   this.fillRect(0, 0, instanceObj.width,
instanceObj.height);
   this.strokeRect(0, 0,
instanceObj.width, instanceObj.height);

   this.font = "normal" + " normal " +
instanceObj.fontsize + "px sans-serif";

this.setFillAndStroke(instanceObj.textco
lor, instanceObj.textcolor);
   this.textBaseline = "bottom";
   this.textAlign = "center";

   // Draw instance name and class
name
   var completeName =
instanceObj.instancename +
      ((instanceObj.classname != "") ? " :
" + instanceObj.classname : "");

this.drawWrappedText(instanceObj.wid
th, Math.min(yTemp0,
instanceObj.height),
      completeName,
instanceObj.fontsize, 0.5 *
instanceObj.width,
      yTemp0 - instanceObj.fontsize);
   //this.fillText(completeName, 0.5 *
instanceObj.width, yTemp0);
   var xTemp = Math.max(0, 0.5 *
instanceObj.width - 0.5 *
this.measureText(completeName).width
);
   if(yTemp0 <= instanceObj.height)
      this.drawSolidLine(xTemp, yTemp0,
xTemp +
Math.min(this.measureText(completeN
ame).width, instanceObj.width),
yTemp0);

   if(instanceObj.showattributes &&
yTemp1 < instanceObj.height) {

this.setFillAndStroke(instanceObj.fillcolo
r, instanceObj.strokestyle);
      this.drawSolidLine(0, yTemp1,
instanceObj.width, yTemp1);


this.setFillAndStroke(instanceObj.textco
lor, instanceObj.textcolor);
      this.textAlign = "start";

      // Draw attributes
      var i = 0;
      while(i < attLen) {
         yTemp1 += V_INC;

this.drawWrappedText(instanceObj.wid
th - H_MARGIN, Math.min(yTemp1,
instanceObj.height),
            instanceObj.attributes[i].value,
instanceObj.fontsize, H_MARGIN,
            yTemp1 -
instanceObj.fontsize);

//this.fillText(instanceObj.attributes[i].v
alue, H_MARGIN, yTemp1);

//if(instanceObj.attributes[i].isstatic) {

if(instanceObj.attributes[i].isstatic &&
instanceObj.width > H_MARGIN &&
yTemp1 <= instanceObj.height) {
            var attrWidth =
Math.min(instanceObj.width,
H_MARGIN +

this.measureText(instanceObj.attributes
[i].value).width);
            this.drawSolidLine(H_MARGIN,
yTemp1, attrWidth, yTemp1);

//this.drawSolidLine(H_MARGIN,
yTemp1,
            //    H_MARGIN +
this.measureText(instanceObj.attributes
[i].value).width, yTemp1);
         }
         i++;
      }
   }

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawNode = function (nodeObj) {
   this.save();
   this.translate(nodeObj.x, nodeObj.y);
   this.rotate(nodeObj.rotate / 180 *
Math.PI);

   // Draw box

   this.fillRect(0, 0, nodeObj.width,
nodeObj.height);
   this.strokeRect(0, 0, nodeObj.width,
nodeObj.height);
   this.beginPath();
   this.moveTo(nodeObj.width +
DEFAULT_NODE_THICKNESS, -
DEFAULT_NODE_THICKNESS);

this.lineTo(DEFAULT_NODE_THICKNESS,
-DEFAULT_NODE_THICKNESS);
   this.lineTo(0, 0);
   this.lineTo(nodeObj.width, 0);
   this.lineTo(nodeObj.width +
DEFAULT_NODE_THICKNESS, -
DEFAULT_NODE_THICKNESS);
   this.lineTo(nodeObj.width +
DEFAULT_NODE_THICKNESS,
nodeObj.height -
DEFAULT_NODE_THICKNESS);
   this.lineTo(nodeObj.width,
nodeObj.height);
   this.lineTo(nodeObj.width, 0);
   this.fill();
   this.stroke();

   var tggdvalLen =
nodeObj.taggedvalues.length;
   var artfLen = nodeObj.showartifacts ?
nodeObj.containedartifacts.length : 0;
   var hasStereotype =
(nodeObj.stereotype == "" ? false :
true);
   var V_MARGIN =
(Math.min(nodeObj.height,
nodeObj.topheight) - nodeObj.fontsize *
(tggdvalLen + (hasStereotype ? 2 : 1))) /
(tggdvalLen + 2);
   V_MARGIN = (V_MARGIN < 0) ? 0 :
V_MARGIN;
   var V_INC = V_MARGIN +
nodeObj.fontsize;

   var yTemp0 = V_MARGIN +
nodeObj.fontsize * (nodeObj.stereotype
== "" ? 1 : 2);

this.setFillAndStroke(nodeObj.textcolor,
nodeObj.textcolor);
   this.textBaseline = "bottom";
   this.textAlign = "center";

   // Draw top compartment
   if(nodeObj.stereotype != "") {
      this.font = "normal normal " +
nodeObj.fontsize + "px sans-serif";

this.drawWrappedText(nodeObj.width,
Math.min(yTemp0 - nodeObj.fontsize,
      nodeObj.topheight,
nodeObj.height), "<<" +
nodeObj.stereotype + ">>",
      nodeObj.fontsize, 0.5 *
nodeObj.width, yTemp0 - 2 *
nodeObj.fontsize);
```

```
      //this.fillText("<<" +
nodeObj.stereotype + ">>", 0.5 *
nodeObj.width,
      //    yTemp0 - nodeObj.fontsize);
    }
    // Draw Node name
    this.font = "normal" + " bold " +
nodeObj.fontsize + "px sans-serif";

this.drawWrappedText(nodeObj.width,
Math.min(yTemp0, nodeObj.topheight,
nodeObj.height),
      nodeObj.nodename,
nodeObj.fontsize, 0.5 * nodeObj.width,
yTemp0 - nodeObj.fontsize);
    //this.fillText(nodeObj.nodename, 0.5
* nodeObj.width, yTemp0);

    // Draw tagged values
    this.font = "normal" + " normal " +
nodeObj.fontsize + "px sans-serif";
    for(var i = 0; i < tggdvalLen; i++) {
      yTemp0 += V_INC;

this.drawWrappedText(nodeObj.width,
Math.min(yTemp0, nodeObj.topheight,
        nodeObj.height), "{" +
nodeObj.taggedvalues[i] + "}",
nodeObj.fontsize,
        0.5 * nodeObj.width, yTemp0 -
nodeObj.fontsize);
      //this.fillText("{" +
nodeObj.taggedvalues[i] + "}", 0.5 *
nodeObj.width, yTemp0);
    }

    if(nodeObj.topheight <=
nodeObj.height) {

this.setFillAndStroke(nodeObj.fillcolor,
nodeObj.strokestyle);
      this.drawSolidLine(0,
nodeObj.topheight, nodeObj.width,
nodeObj.topheight);

      // Draw contained artifacts
      if(nodeObj.showartifacts) {

this.setFillAndStroke(nodeObj.textcolor,
nodeObj.textcolor);
        this.textAlign = "start";

        var yTemp1 = nodeObj.topheight;
        for(var i = 0; i < artfLen; i++) {
          yTemp1 += H_MARGIN +
nodeObj.fontsize;

this.drawWrappedText(nodeObj.width -
H_MARGIN, Math.min(yTemp1,
nodeObj.height),

nodeObj.containedartifacts[i],
nodeObj.fontsize, H_MARGIN,
              yTemp1 - nodeObj.fontsize);

//this.fillText(nodeObj.containedartifact
s[i], H_MARGIN, yTemp1);
        }
      }
    }

    this.restore();
};

CanvasRenderingContext2D.prototype.d
rawStatebox = function (sbObj) {
    this.save();
    this.translate(sbObj.x, sbObj.y);
    this.rotate(sbObj.rotate / 180 *
Math.PI);

    // Draw rounded rectangle box
    var rectRad = DEFAULT_RADIUSRATIO
* ((sbObj.width < sbObj.height) ?
sbObj.width : sbObj.height);
    this.drawRoundedRectangle(0, 0,
sbObj.width, sbObj.height, rectRad, 0);

    var ialen =
sbObj.showinternalactivities ?
sbObj.internalactivitys.length : 0;
    var V_MARGIN = (sbObj.height -
sbObj.fontsize * (ialen + 1) - 2 * rectRad)
/ (ialen + 1);
    V_MARGIN = (V_MARGIN < 0) ? 0 :
V_MARGIN;
    var V_INC = V_MARGIN +
sbObj.fontsize;

    var yTemp0;
    if(!sbObj.showinternalactivities ||
sbObj.internalactivitys.length == 0) {
      yTemp0 = 0.5 * (sbObj.height +
sbObj.fontsize);
    } else {
      yTemp0 = V_INC;
    }

    this.setFillAndStroke(sbObj.textcolor,
sbObj.textcolor);
    this.textBaseline = "bottom";
    this.textAlign = "center";

    // Draw state name
    this.font = "normal" + " bold " +
sbObj.fontsize + "px sans-serif";
    this.drawWrappedText(sbObj.width,
Math.min(yTemp0, sbObj.height),
sbObj.statename, sbObj.fontsize,
      0.5 * sbObj.width, yTemp0 -
sbObj.fontsize);
    //this.fillText(sbObj.statename, 0.5 *
sbObj.width, yTemp0);

    // Draw internal activities
    var yTemp1 = yTemp0 + V_MARGIN;
    if(sbObj.showinternalactivities &&
sbObj.internalactivitys.length != 0 &&
yTemp1 <= sbObj.height) {
      this.font = "normal" + " normal " +
sbObj.fontsize + "px sans-serif";
      this.setFillAndStroke(sbObj.fillcolor,
sbObj.strokestyle);
      this.drawSolidLine(0, yTemp1,
sbObj.width, yTemp1);
      yTemp1 += rectRad +
sbObj.fontsize;


this.setFillAndStroke(sbObj.textcolor,
sbObj.textcolor);
      this.textAlign = "start";

      for(var i = 0; i < ialen; i++) {

this.drawWrappedText(sbObj.width -
rectRad, Math.min(yTemp1,
sbObj.height),
            sbObj.internalactivitys[i],
sbObj.fontsize, rectRad, yTemp1-
sbObj.fontsize);

//this.fillText(sbObj.internalactivitys[i],
rectRad, yTemp1);
        yTemp1 += V_INC;
      }
    }

    this.restore();
};

CanvasRenderingContext2D.prototype.d
rawListBoxPin = function(x, y, lbpSize,
horizontal) {
    if(horizontal) {
      this.beginPath();
      this.moveTo(x, y - 0.5 * lbpSize);
      this.lineTo(x + 4 * lbpSize, y - 0.5 *
lbpSize);
      this.lineTo(x + 4 * lbpSize, y + 0.5 *
lbpSize);
      this.lineTo(x, y + 0.5 * lbpSize);
      this.closePath();
      this.moveTo(x + lbpSize, y - 0.5 *
lbpSize);
      this.lineTo(x + lbpSize, y + 0.5 *
lbpSize);
      this.moveTo(x + 2 * lbpSize, y - 0.5
* lbpSize);
      this.lineTo(x + 2 * lbpSize, y + 0.5 *
lbpSize);
      this.moveTo(x + 3 * lbpSize, y - 0.5
* lbpSize);
      this.lineTo(x + 3 * lbpSize, y + 0.5 *
lbpSize);
    } else {
      this.beginPath();
      this.moveTo(x - 0.5 * lbpSize, y);
      this.lineTo(x - 0.5 * lbpSize, y + 4 *
lbpSize);
      this.lineTo(x + 0.5 * lbpSize, y + 4 *
lbpSize);
      this.lineTo(x + 0.5 * lbpSize, y);
      this.closePath();
```

```
      this.moveTo(x - 0.5 * lbpSize, y +
lbpSize);
      this.lineTo(x + 0.5 * lbpSize, y +
lbpSize);
      this.moveTo(x - 0.5 * lbpSize, y + 2
* lbpSize);
      this.lineTo(x + 0.5 * lbpSize, y + 2 *
lbpSize);
      this.moveTo(x - 0.5 * lbpSize, y + 3
* lbpSize);
      this.lineTo(x + 0.5 * lbpSize, y + 3 *
lbpSize);
   }
   this.fill();
   this.stroke();
};

CanvasRenderingContext2D.prototype.d
rawExpansionregion = function (erObj) {
   this.save();
   this.translate(erObj.x, erObj.y);
   this.rotate(erObj.rotate / 180 *
Math.PI);

   var rectRad = DEFAULT_RADIUSRATIO
* ((erObj.width < erObj.height) ?
erObj.width : erObj.height);
   this.drawDashedRoundedRect(0, 0,
erObj.width, erObj.height, rectRad, 0,
[7, this.lineWidth + 7]);

   // Draw list box pins

this.drawListBoxPin(erObj.listboxpin1x,
erObj.listboxpin1y, erObj.listboxpinsize,
      ((erObj.listboxpin1y == 0 &&
erObj.listboxpin1x < erObj.width) ||
       erObj.listboxpin1y ==
erObj.height));

this.drawListBoxPin(erObj.listboxpin2x,
erObj.listboxpin2y, erObj.listboxpinsize,
      ((erObj.listboxpin2y == 0 &&
erObj.listboxpin2x < erObj.width) ||
       erObj.listboxpin2y ==
erObj.height));

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawArrowheadNotation = function (x, y,
w, h, r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   this.beginPath();
   this.moveTo(0, 0);
   this.lineTo(0, h);
   this.lineTo(w, 0.5 * h);
   this.closePath();
   this.fill();
   this.stroke();

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawActivationBar = function (x, y, w, h,
r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   this.strokeRect(0, 0, w, h);
   this.fillRect(0, 0, w, h);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawLifeline = function (x, y, w, h, r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   this.drawDashedLine(w / 2, 0, w / 2,
h, [7, this.lineWidth + 7]);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawDeletion = function (x, y, w, h, r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   this.drawSolidLine(0, 0, w, h);
   this.drawSolidLine(w, 0, 0, h);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawFilledCircle = function (x, y, w, h, r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   var rad = Math.min(w, h) / 2;
   this.beginPath();
   this.arc(w / 2, h - rad, rad, 0, Math.PI
* 2, false);
   this.closePath();
   this.fill();
   this.stroke();

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawBorderedFilledCircle = function
(fsObj) {
   this.save();
   this.translate(fsObj.x, fsObj.y);
   this.rotate(fsObj.rotate / 180 *
Math.PI);

   var rad = Math.min(fsObj.width,
fsObj.height) / 2;
   this.beginPath();
   this.arc(fsObj.width / 2, fsObj.height -
rad, rad, 0, Math.PI * 2, false);
   this.closePath();
   this.fill();
   this.stroke();


this.setFillAndStroke(fsObj.strokestyle,
fsObj.strokestyle);
   var rad2 = rad * (1 -
DEFAULT_FINALSTATE_BORDER);
   this.beginPath();
   this.arc(fsObj.width / 2, fsObj.height -
rad, rad2, 0, Math.PI * 2, false);
   this.closePath();
   this.fill();

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawShallowHistPS = function (shObj) {
   this.save();
   this.translate(shObj.x, shObj.y);
   this.rotate(shObj.rotate / 180 *
Math.PI);

   var rad = Math.min(shObj.width,
shObj.height) / 2;
   this.beginPath();
   this.arc(shObj.width / 2, shObj.height
- rad, rad, 0, Math.PI * 2, false);
   this.closePath();
   this.fill();
   this.stroke();


this.setFillAndStroke(shObj.strokestyle,
shObj.strokestyle);
   this.textBaseline = "middle";
   this.font = "normal bold " + rad + "px
sans-serif";
   this.textAlign = "center";
   this.fillText("H", shObj.width / 2,
shObj.height - rad);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawDeepHistPS = function (dhObj) {
   this.save();
   this.translate(dhObj.x, dhObj.y);
   this.rotate(dhObj.rotate / 180 *
Math.PI);

   var rad = Math.min(dhObj.width,
dhObj.height) / 2;
   this.beginPath();
   this.arc(dhObj.width / 2, dhObj.height
- rad, rad, 0, Math.PI * 2, false);
   this.closePath();
   this.fill();
   this.stroke();
```

```
this.setFillAndStroke(dhObj.strokestyle,
dhObj.strokestyle);
   this.textBaseline = "middle";
   this.font = "normal bold " + rad + "px
sans-serif";
   this.textAlign = "center";
   this.fillText("H*", dhObj.width / 2,
dhObj.height - rad);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawBlackBar = function (x, y, w, h, r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   this.fillRect(0, (h -
DEFAULT_BLACKBAR_THICKNESS) / 2,
w, DEFAULT_BLACKBAR_THICKNESS);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawDiamond = function (x, y, w, h, r) {
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   this.beginPath();
   this.moveTo(w / 2, 0);
   this.lineTo(w, h / 2);
   this.lineTo(w / 2, h);
   this.lineTo(0, h / 2);
   this.closePath();
   this.fill();
   this.stroke();

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawSmallSquare = function (x, y, w, h, r)
{
   this.save();
   this.translate(x, y);
   this.rotate(r / 180 * Math.PI);

   var side = Math.min(w, h);
   this.fillRect(0, 0, side, side);
   this.strokeRect(0, 0, side, side);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawFlowFinal = function (shObj) {
   this.save();
   this.translate(shObj.x, shObj.y);
   this.rotate(shObj.rotate / 180 *
Math.PI);

   var rad = Math.min(shObj.width,
shObj.height) / 2;
   this.beginPath();
   this.moveTo(shObj.width / 2,
shObj.height - rad, rad);
   this.arc(shObj.width / 2, shObj.height
- rad, rad, Math.PI * 0.25, Math.PI *
0.75, false);
   this.lineTo(shObj.width / 2,
shObj.height - rad, rad);
   this.arc(shObj.width / 2, shObj.height
- rad, rad, Math.PI * 0.75, Math.PI *
1.25, false);
   this.lineTo(shObj.width / 2,
shObj.height - rad, rad);
   this.arc(shObj.width / 2, shObj.height
- rad, rad, Math.PI * 1.25, Math.PI *
1.75, false);
   this.lineTo(shObj.width / 2,
shObj.height - rad, rad);
   this.arc(shObj.width / 2, shObj.height
- rad, rad, Math.PI * 1.75, Math.PI *
2.25, false);
   this.closePath();
   this.fill();
   this.stroke();

   this.restore();
};
```

---

diagram shapes 4.js

```
CanvasRenderingContext2D.prototype.d
rawFrame = function (frameObj) {
   this.save();
   this.translate(frameObj.x,
frameObj.y);
   this.rotate(frameObj.rotate / 180 *
Math.PI);

   this.font = frameObj.fontstyle + " " +
frameObj.fontweight + " " +
frameObj.fontsize
      + "px " + frameObj.fontfamily;
   var labelWidth =
Math.min(this.measureText(frameObj.la
bel).width + H_MARGIN * 3,
frameObj.width / 2);
   var labelThickness =
Math.min(H_MARGIN * 2 +
frameObj.fontsize, frameObj.height / 5);

   this.beginPath();
   this.moveTo(labelWidth, 0);
   this.lineTo(0, 0);
   this.lineTo(0, frameObj.height);
   this.lineTo(frameObj.width,
frameObj.height);
   this.lineTo(frameObj.width, 0);
   this.lineTo(labelWidth, 0);
   this.fill();
   this.lineTo(labelWidth, labelThickness
/ 2);
   this.lineTo(labelWidth -
labelThickness / 2, labelThickness);
   this.lineTo(0, labelThickness);
   this.stroke();

   // Write the label
   this.textBaseline = "bottom";

this.setFillAndStroke(frameObj.textcolor
, frameObj.strokestyle);
   var maxwidth = labelWidth - 2 *
H_MARGIN;
   this.drawWrappedText(maxwidth,
labelThickness, frameObj.label,
frameObj.fontsize, H_MARGIN,
H_MARGIN);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawPackage1 = function (pkg1Obj) {
   this.save();
   this.translate(pkg1Obj.x, pkg1Obj.y);
   this.rotate(pkg1Obj.rotate / 180 *
Math.PI);

   this.font = pkg1Obj.fontstyle + " " +
pkg1Obj.fontweight + " " +
pkg1Obj.fontsize
      + "px " + pkg1Obj.fontfamily;
   var labelWidth = pkg1Obj.width / 3;
   var labelThickness =
Math.min(H_MARGIN * 2 +
pkg1Obj.fontsize, pkg1Obj.height / 5);

   this.beginPath();
   this.moveTo(0, 0);
   this.lineTo(0, pkg1Obj.height);
   this.lineTo(pkg1Obj.width,
pkg1Obj.height);
   this.lineTo(pkg1Obj.width,
labelThickness);
   this.lineTo(labelWidth,
labelThickness);
   this.lineTo(labelWidth, 0);
   this.closePath();
   this.fill();
   this.moveTo(0, labelThickness);
   this.lineTo(labelWidth,
labelThickness);
   this.stroke();

   // Write the label
   this.textBaseline = "bottom";
   this.textAlign = "center";

this.setFillAndStroke(pkg1Obj.textcolor,
pkg1Obj.strokestyle);
   var nameHeight =
this.getWrappedTextHeight(pkg1Obj.wi
dth - 2 * H_MARGIN,
      pkg1Obj.height - labelThickness,
pkg1Obj.label, pkg1Obj.fontsize,
      pkg1Obj.width / 2, labelThickness +
H_MARGIN);

this.drawWrappedText(pkg1Obj.width -
```

```
2 * H_MARGIN, pkg1Obj.height -
labelThickness, pkg1Obj.label,
      pkg1Obj.fontsize, pkg1Obj.width /
2,
      (pkg1Obj.height + labelThickness -
nameHeight) / 2)

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawPackage2 = function (pkg2Obj) {
   this.save();
   this.translate(pkg2Obj.x, pkg2Obj.y);
   this.rotate(pkg2Obj.rotate / 180 *
Math.PI);

   this.font = pkg2Obj.fontstyle + " " +
pkg2Obj.fontweight + " " +
pkg2Obj.fontsize
      + "px " + pkg2Obj.fontfamily;
   var labelWidth =
Math.min(this.measureText(pkg2Obj.la
bel).width + H_MARGIN * 3,
pkg2Obj.width / 2);
   var labelThickness =
Math.min(H_MARGIN * 2 +
pkg2Obj.fontsize, pkg2Obj.height / 5);

   this.beginPath();
   this.moveTo(0, 0);
   this.lineTo(0, pkg2Obj.height);
   this.lineTo(pkg2Obj.width,
pkg2Obj.height);
   this.lineTo(pkg2Obj.width,
labelThickness);
   this.lineTo(labelWidth,
labelThickness);
   this.lineTo(labelWidth, 0);
   this.closePath();
   this.fill();
   this.moveTo(0, labelThickness);
   this.lineTo(labelWidth,
labelThickness);
   this.stroke();

   // Write the label
   this.textBaseline = "bottom";

this.setFillAndStroke(pkg2Obj.textcolor,
pkg2Obj.strokestyle);
   var maxwidth = labelWidth - 2 *
H_MARGIN;
   this.drawWrappedText(maxwidth,
labelThickness, pkg2Obj.label,
pkg2Obj.fontsize, H_MARGIN,
H_MARGIN);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawComponent = function (compObj) {
   this.save();
   this.translate(compObj.x, compObj.y);
   this.rotate(compObj.rotate / 180 *
Math.PI);

   this.fillRect(0, 0, compObj.width,
compObj.height);
   this.strokeRect(0, 0, compObj.width,
compObj.height);

   // Write the label
   this.font = compObj.fontstyle + " " +
compObj.fontweight + " " +
compObj.fontsize
      + "px " + compObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";

this.setFillAndStroke(compObj.textcolor
, compObj.strokestyle);

   var nameHeight =
this.getWrappedTextHeight(compObj.wi
dth - 2 * H_MARGIN,
      compObj.height - 2 * H_MARGIN,
compObj.label, compObj.fontsize,
      H_MARGIN, H_MARGIN);

this.drawWrappedText(compObj.width -
2 * H_MARGIN, compObj.height - 2 *
H_MARGIN,
      compObj.label, compObj.fontsize,
compObj.width / 2, (compObj.height -
nameHeight) / 2)

   // Draw icon

this.setFillAndStroke(compObj.fillcolor,
compObj.strokestyle);
   var compIcon =
Math.min(compObj.width,
compObj.height) * 0.20;
   var iconX = compObj.width * 0.95 -
compIcon * 1.5;//- H_MARGIN -
compIcon * 1.5;
   var iconY = compObj.height * 0.06;
   this.strokeRect(iconX, iconY,
compIcon * 1.5, compIcon);
   var tempH = compIcon * 0.20;
   var tempW = compIcon * 0.40;
   this.fillRect(iconX - tempW / 2, iconY +
compIcon * 0.2, tempW, tempH);
   this.strokeRect(iconX - tempW / 2,
iconY + compIcon * 0.2, tempW,
tempH);
   this.fillRect(iconX - tempW / 2, iconY +
compIcon * 0.6, tempW, tempH);
   this.strokeRect(iconX - tempW / 2,
iconY + compIcon * 0.6, tempW,
tempH);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawArtifact = function (artfObj) {
   this.save();
   this.translate(artfObj.x, artfObj.y);
   this.rotate(artfObj.rotate / 180 *
Math.PI);

   this.fillRect(0, 0, artfObj.width,
artfObj.height);
   this.strokeRect(0, 0, artfObj.width,
artfObj.height);

   // Write the label
   this.font = artfObj.fontstyle + " " +
artfObj.fontweight + " " +
artfObj.fontsize
      + "px " + artfObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";

this.setFillAndStroke(artfObj.textcolor,
artfObj.strokestyle);

   var nameHeight =
this.getWrappedTextHeight(artfObj.wid
th - 2 * H_MARGIN,
      artfObj.height - 2 * H_MARGIN,
artfObj.label, artfObj.fontsize,
      H_MARGIN, H_MARGIN);
   this.drawWrappedText(artfObj.width
- 2 * H_MARGIN, artfObj.height - 2 *
H_MARGIN,
      artfObj.label, artfObj.fontsize,
artfObj.width / 2, (artfObj.height -
nameHeight) / 2);

   // Draw icon
   this.setFillAndStroke(artfObj.fillcolor,
artfObj.strokestyle);
   var artfIcon = Math.min(artfObj.width
* 0.20, artfObj.height * 0.20);

   var x1 = artfObj.width * 0.95 - artfIcon
* 1.5;
   var x2 = x1 + artfIcon * 1.5;
   var x3 = x2 - artfIcon * 0.33;
   var y1 = artfObj.height * 0.06;
   var y2 = y1 + artfIcon;
   var y3 = y1 + artfIcon * 0.33;

   this.beginPath();
   this.moveTo(x3, y1);
   this.lineTo(x1, y1);
   this.lineTo(x1, y2);
   this.lineTo(x2, y2);
   this.lineTo(x2, y3);
   this.lineTo(x3, y1);
   this.fill();
   this.moveTo(x3, y1);
   this.lineTo(x3, y3);
   this.lineTo(x2, y3);
   this.stroke();

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawUseCase = function (ucObj) {
   this.save();
   this.translate(ucObj.x, ucObj.y);
```

```
   this.rotate(ucObj.rotate / 180 *
Math.PI);

   this.drawEllipse(0, 0, ucObj.width,
ucObj.height, 0)

   // Write the label
   this.font = ucObj.fontstyle + " " +
ucObj.fontweight + " " + ucObj.fontsize
      + "px " + ucObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";
   this.setFillAndStroke(ucObj.textcolor,
ucObj.strokestyle);

   var nameHeight =
this.getWrappedTextHeight(ucObj.widt
h - 2 * H_MARGIN,
      ucObj.height - 2 * H_MARGIN,
ucObj.label, ucObj.fontsize,
      H_MARGIN, H_MARGIN);
   this.drawWrappedText(ucObj.width -
2 * H_MARGIN, ucObj.height - 2 *
H_MARGIN,
      ucObj.label, ucObj.fontsize,
ucObj.width / 2, (ucObj.height -
nameHeight) / 2);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawActor = function (actorObj) {
   this.save();
   this.translate(actorObj.x, actorObj.y);
   this.rotate(actorObj.rotate / 180 *
Math.PI);

   var headRad =
Math.min(actorObj.height / 12,
actorObj.width / 4);

   this.beginPath();
   this.moveTo(actorObj.width / 2,
actorObj.height / 6);
   this.arc(actorObj.width / 2, headRad,
headRad, Math.PI * 0.5, Math.PI * 2.5,
false);
   this.fill();
   this.lineTo(actorObj.width / 2,
actorObj.height * 4 / 9);
   this.moveTo(0, actorObj.height * 2 /
9);
   this.lineTo(actorObj.width,
actorObj.height * 2 / 9);
   this.moveTo(actorObj.width / 2,
actorObj.height * 4 / 9);
   this.lineTo(0, actorObj.height * 2 / 3);
   this.moveTo(actorObj.width / 2,
actorObj.height * 4 / 9);
   this.lineTo(actorObj.width,
actorObj.height * 2 / 3);
   this.stroke();

   // Write the label
   this.font = actorObj.fontstyle + " " +
actorObj.fontweight + " " +
actorObj.fontsize
      + "px " + actorObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";

this.setFillAndStroke(actorObj.textcolor,
actorObj.strokestyle);


this.drawWrappedText(actorObj.width,
actorObj.height, actorObj.label,
      actorObj.fontsize, actorObj.width /
2, actorObj.height * 2 / 3 + H_MARGIN);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawSuperstate = function (ssObj) {
   this.save();
   this.translate(ssObj.x, ssObj.y);
   this.rotate(ssObj.rotate / 180 *
Math.PI);

   // Draw rounded rectangle box
   var rectRad = DEFAULT_RADIUSRATIO
* ((ssObj.width < ssObj.height) ?
ssObj.width : ssObj.height);
   this.drawRoundedRectangle(0, 0,
ssObj.width, ssObj.height, rectRad, 0);

   // Write label
   this.setFillAndStroke(ssObj.textcolor,
ssObj.textcolor);
   this.font = ssObj.fontstyle + " " +
ssObj.fontweight + " " + ssObj.fontsize
         + "px " + ssObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";

   var nameHeight =
this.getWrappedTextHeight(ssObj.width
- 2 * H_MARGIN, ssObj.height -
H_MARGIN,
      ssObj.label, ssObj.fontsize,
ssObj.width / 2, H_MARGIN);
   this.drawWrappedText(ssObj.width -
2 * H_MARGIN, ssObj.height -
H_MARGIN,
      ssObj.label, ssObj.fontsize,
ssObj.width / 2, H_MARGIN);

   var lineY = nameHeight + 0.5 *
ssObj.fontsize;
   if(lineY <= ssObj.height) {
      this.setFillAndStroke(ssObj.fillcolor,
ssObj.strokestyle);
      this.drawSolidLine(0, lineY,
ssObj.width, lineY);
   }

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawAction = function (actionObj) {
   this.save();
   this.translate(actionObj.x,
actionObj.y);
   this.rotate(actionObj.rotate / 180 *
Math.PI);

   // Draw rounded rectangle box
   var rectRad =
DEFAULT_ACTIONRADIUS *
((actionObj.width < actionObj.height) ?
      actionObj.width : actionObj.height);
   this.drawRoundedRectangle(0, 0,
actionObj.width, actionObj.height,
rectRad, 0);

   // Write the label
   this.font = actionObj.fontstyle + " " +
actionObj.fontweight + " " +
actionObj.fontsize
      + "px " + actionObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";

this.setFillAndStroke(actionObj.textcolo
r, actionObj.strokestyle);

   var nameHeight =
this.getWrappedTextHeight(actionObj.w
idth - 2 * H_MARGIN,
      actionObj.height - 2 * H_MARGIN,
actionObj.label, actionObj.fontsize,
      H_MARGIN, H_MARGIN);

this.drawWrappedText(actionObj.width
- 2 * H_MARGIN, actionObj.height - 2 *
H_MARGIN,
      actionObj.label, actionObj.fontsize,
actionObj.width / 2, (actionObj.height -
nameHeight) / 2);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawSubactivity = function (saObj) {
   this.save();
   this.translate(saObj.x, saObj.y);
   this.rotate(saObj.rotate / 180 *
Math.PI);

   // Draw rounded rectangle box
   var rectRad =
DEFAULT_ACTIONRADIUS *
((saObj.width < saObj.height) ?
      saObj.width : saObj.height);
   this.drawRoundedRectangle(0, 0,
saObj.width, saObj.height, rectRad, 0);

   // Write the label
   this.font = saObj.fontstyle + " " +
saObj.fontweight + " " + saObj.fontsize
      + "px " + saObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";
```

```
   this.setFillAndStroke(saObj.textcolor,
saObj.strokestyle);

   var nameHeight =
this.getWrappedTextHeight(saObj.width
- 2 * H_MARGIN,
      saObj.height - 2 * H_MARGIN,
saObj.label, saObj.fontsize,
      H_MARGIN, H_MARGIN);
   this.drawWrappedText(saObj.width -
2 * H_MARGIN, saObj.height - 2 *
H_MARGIN,
      saObj.label, saObj.fontsize,
saObj.width / 2, (saObj.height -
nameHeight) / 2);

   // Draw subactivity indicator
   this.setFillAndStroke(saObj.fillcolor,
saObj.strokestyle);
   var saIcon = Math.min(saObj.width,
saObj.height) * 0.15;
   var iconX = saObj.width * 0.75;
   var iconY = saObj.height * 0.25 -
saIcon * 0.5;
   this.beginPath();
   this.moveTo(iconX, iconY);
   this.lineTo(iconX, iconY + saIcon *
0.5);
   this.lineTo(iconX - saIcon * 0.75, iconY
+ saIcon * 0.5);
   this.lineTo(iconX - saIcon * 0.75, iconY
+ saIcon);
   this.moveTo(iconX, iconY + saIcon);
   this.lineTo(iconX, iconY + saIcon *
0.5);
   this.lineTo(iconX + saIcon * 0.75,
iconY + saIcon * 0.5);
   this.lineTo(iconX + saIcon * 0.75,
iconY + saIcon);
   //this.lineTo(iconX - saIcon * 0.75,
iconY + saIcon);
   this.stroke();

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawTimeSignal = function (tsObj) {
   this.save();
   this.translate(tsObj.x, tsObj.y);
   this.rotate(tsObj.rotate / 180 *
Math.PI);

   this.beginPath();
   this.moveTo(0, 0);
   this.lineTo(tsObj.width, 0);
   this.lineTo(0, tsObj.height * 0.75);
   this.lineTo(tsObj.width, tsObj.height *
0.75);
   this.closePath();
   this.fill();
   this.stroke();

   // Write the label
   this.font = tsObj.fontstyle + " " +
tsObj.fontweight + " " + tsObj.fontsize
      + "px " + tsObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";
   this.setFillAndStroke(tsObj.textcolor,
tsObj.strokestyle);

   this.drawWrappedText(tsObj.width -
2 * H_MARGIN, tsObj.height,
tsObj.label,
      tsObj.fontsize, tsObj.width / 2,
tsObj.height * 0.75 + H_MARGIN);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawAcceptSignal = function (asObj) {
   this.save();
   this.translate(asObj.x, asObj.y);
   this.rotate(asObj.rotate / 180 *
Math.PI);

   this.beginPath();
   this.moveTo(0, 0);
   this.lineTo(asObj.width * 0.20,
asObj.height / 2);
   this.lineTo(0, asObj.height);
   this.lineTo(asObj.width, asObj.height);
   this.lineTo(asObj.width, 0);
   this.closePath();
   this.fill();
   this.stroke();

   // Write the label
   this.font = asObj.fontstyle + " " +
asObj.fontweight + " " + asObj.fontsize
      + "px " + asObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";
   this.setFillAndStroke(asObj.textcolor,
asObj.strokestyle);

   var nameHeight =
this.getWrappedTextHeight(asObj.width
* 0.8 - 2 * H_MARGIN,
      asObj.height - 2 * H_MARGIN,
asObj.label, asObj.fontsize,
       H_MARGIN, H_MARGIN);
   this.drawWrappedText(asObj.width *
0.8 - 2 * H_MARGIN, asObj.height - 2 *
H_MARGIN,
      asObj.label, asObj.fontsize,
asObj.width * 0.6, (asObj.height -
nameHeight) / 2);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawSendSignal = function (ssObj) {
   this.save();
   this.translate(ssObj.x, ssObj.y);
   this.rotate(ssObj.rotate / 180 *
Math.PI);

   this.beginPath();
   this.moveTo(0, 0);
   this.lineTo(ssObj.width * 0.8, 0);
   this.lineTo(ssObj.width, ssObj.height *
0.5);
   this.lineTo(ssObj.width * 0.8,
ssObj.height);
   this.lineTo(0, ssObj.height);
   this.closePath();
   this.fill();
   this.stroke();

   // Write the label
   this.font = ssObj.fontstyle + " " +
ssObj.fontweight + " " + ssObj.fontsize
      + "px " + ssObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";
   this.setFillAndStroke(ssObj.textcolor,
ssObj.strokestyle);

   var nameHeight =
this.getWrappedTextHeight(ssObj.width
* 0.8 - 2 * H_MARGIN,
      ssObj.height - 2 * H_MARGIN,
ssObj.label, ssObj.fontsize,
       + H_MARGIN, H_MARGIN);
   this.drawWrappedText(ssObj.width *
0.8 - 2 * H_MARGIN, ssObj.height - 2 *
H_MARGIN,
      ssObj.label, ssObj.fontsize,
ssObj.width * 0.4, (ssObj.height -
nameHeight) / 2);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawConnector = function
(connectorObj) {
   this.save();
   this.translate(connectorObj.x,
connectorObj.y);
   this.rotate(connectorObj.rotate / 180
* Math.PI);

   var rad =
Math.min(connectorObj.width,
connectorObj.height) / 2;
   this.beginPath();
   this.arc(connectorObj.width / 2,
connectorObj.height - rad, rad, 0,
Math.PI * 2, false);
   this.closePath();
   this.fill();
   this.stroke();

   // Write the label
   this.font = connectorObj.fontstyle + "
" + connectorObj.fontweight + " " +
connectorObj.fontsize
      + "px " + connectorObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";

this.setFillAndStroke(connectorObj.textc
olor, connectorObj.strokestyle);
```

```
   var nameHeight =
this.getWrappedTextHeight(connectorO
bj.width / 2 + rad, connectorObj.height,
      connectorObj.label,
connectorObj.fontsize,
connectorObj.width / 2,
      connectorObj.height - 2 * rad);

this.drawWrappedText(connectorObj.wi
dth / 2 + rad, connectorObj.height,
      connectorObj.label,
connectorObj.fontsize,
connectorObj.width / 2,
      connectorObj.height - rad -
nameHeight / 2);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawTransformation = function
(transfObj) {
   this.save();
   this.translate(transfObj.x,
transfObj.y);
   this.rotate(transfObj.rotate / 180 *
Math.PI);

   var foldlen = (transfObj.height >
transfObj.width) ? transfObj.width :
transfObj.height;
   foldlen *= 0.25;

   var x1 = 0;
   var x2 = x1 + transfObj.width;
   var x3 = x2 - foldlen;
   var y1 = 0;
   var y2 = y1 + transfObj.height;
   var y3 = y1 + foldlen;

   this.beginPath();
   this.moveTo(x3, y1);
   this.lineTo(x1, y1);
   this.lineTo(x1, y2);
   this.lineTo(x2, y2);
   this.lineTo(x2, y3);
   this.lineTo(x3, y1);
   this.fill();
   this.moveTo(x3, y1);
   this.lineTo(x3, y3);
   this.lineTo(x2, y3);
   this.stroke();

   // Write the textual content
   this.textBaseline = "top";
   this.font = transfObj.fontstyle + " " +
transfObj.fontweight + " " +
transfObj.fontsize
      + "px " + transfObj.fontfamily;

this.setFillAndStroke(transfObj.textcolor
, transfObj.strokestyle);
   var maxwidth = transfObj.width - 2 *
H_MARGIN;
   var maxheight = y2 - H_MARGIN -
transfObj.fontsize;
   this.textAlign = "center";
   var labelStart =
this.getWrappedTextHeight(maxwidth,
maxheight, "<<transformation>>",
      transfObj.fontsize, transfObj.width
/ 2, y3) + H_MARGIN;
   this.drawWrappedText(maxwidth,
maxheight, "<<transformation>>",
transfObj.fontsize,
      transfObj.width / 2, y3);
   this.textAlign = "start";
   this.drawWrappedText(maxwidth,
maxheight, transfObj.label,
transfObj.fontsize,
      x1 + H_MARGIN, y3 + labelStart);

   this.restore();
};

CanvasRenderingContext2D.prototype.d
rawPart = function (partObj) {
   this.save();
   this.translate(partObj.x, partObj.y);
   this.rotate(partObj.rotate / 180 *
Math.PI);

   this.fillRect(0, 0, partObj.width,
partObj.height);
   this.strokeRect(0, 0, partObj.width,
partObj.height);

   // Write the label
   this.font = partObj.fontstyle + " " +
partObj.fontweight + " " +
partObj.fontsize
      + "px " + partObj.fontfamily;
   this.textBaseline = "bottom";
   this.textAlign = "center";

this.setFillAndStroke(partObj.textcolor,
partObj.strokestyle);

   var nameHeight =
this.getWrappedTextHeight(partObj.wid
th - 2 * H_MARGIN,
      partObj.height, partObj.label,
partObj.fontsize, H_MARGIN, 0);
   this.drawWrappedText(partObj.width
- 2 * H_MARGIN, partObj.height,
      partObj.label, partObj.fontsize,
partObj.width / 2, (partObj.height -
nameHeight) / 2);

   this.restore();
};
```

editor.js

```
var USERNAME;
var DIAGRAM_ID;

function init(iUsername, iDiagramID) {
   USERNAME = iUsername;
   DIAGRAM_ID = iDiagramID;
   getDiagram();

   canvas_init();
   tools_init();
   collaborators_init();
   propertiesBox_init();
   communication_init();
}
```

objects.js

```
function Rectangle(iX, iY, w, h) {
   this.objecttype = "rectangle";
   this.x = iX;
   this.y = iY;
   this.width = w;
   this.height = h;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function Ellipse(iX, iY, w, h) {
   this.objecttype = "ellipse";
   this.x = iX;
   this.y = iY;
   this.width = w;
   this.height = h;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function Arrowhead(iX, iY) {
   this.objecttype = "arrowhead";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_SMALLSIZE;
   this.height = DEFAULT_SMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = "0,0,0,1";
   this.rotate = DEFAULT_ROTATE;
}

function ActivationBar(iX, iY) {
   this.objecttype = "activationbar";
   this.x = iX;
   this.y = iY;
   this.width =
DEFAULT_ACTIVATIONBAR_W;
```

```
   this.height =
DEFAULT_ACTIVATIONBAR_H;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function Lifeline(iX, iY) {
   this.objecttype = "lifeline";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_LIFELINE_W;
   this.height = DEFAULT_LIFELINE_H;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function Deletion(iX, iY) {
   this.objecttype = "deletion";
   this.x = iX;
   this.y = iY;
   this.width =
DEFAULT_DELETION_SIZE;
   this.height =
DEFAULT_DELETION_SIZE;
   this.linewidth =
DEFAULT_DELETION_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function InitialPS(iX, iY) {
   this.objecttype = "initialps";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_SMALLSIZE;
   this.height = DEFAULT_SMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor =
DEFAULT_STROKESTYLE;
   this.rotate = DEFAULT_ROTATE;
}

function FinalState(iX, iY) {
   this.objecttype = "finalstate";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_SMALLSIZE;
   this.height = DEFAULT_SMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function ShallowHistPS(iX, iY) {
   this.objecttype = "shallowhistps";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_SMALLSIZE;
   this.height = DEFAULT_SMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function DeepHistPS(iX, iY) {
   this.objecttype = "deephistps";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_SMALLSIZE;
   this.height = DEFAULT_SMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function InitialNode(iX, iY) {
   this.objecttype = "initialnode";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_SMALLSIZE;
   this.height = DEFAULT_SMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor =
DEFAULT_STROKESTYLE;
   this.rotate = DEFAULT_ROTATE;
}

function FinalNode(iX, iY) {
   this.objecttype = "finalnode";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_SMALLSIZE;
   this.height = DEFAULT_SMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function BlackBar(iX, iY) {
   this.objecttype = "blackbar";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_WIDTH;
   this.height = DEFAULT_HEIGHT;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor =
DEFAULT_STROKESTYLE;
   this.rotate = DEFAULT_ROTATE;
}

function Diamond(iX, iY) {
   this.objecttype = "diamond";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_MEDIUMSIZE;
   this.height = DEFAULT_MEDIUMSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function Pin(iX, iY) {
   this.objecttype = "pin";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_XSMALLSIZE;
   this.height = DEFAULT_XSMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function FlowFinal(iX, iY) {
   this.objecttype = "flowfinal";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_SMALLSIZE;
```

```
   this.height = DEFAULT_SMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function Port(iX, iY) {
   this.objecttype = "port";
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_XSMALLSIZE;
   this.height = DEFAULT_XSMALLSIZE;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.rotate = DEFAULT_ROTATE;
}

function Note(iX, iY, iLabel) {
   this.objecttype = "note";
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.fontsize = DEFAULT_FONTSIZE;
   this.fontfamily =
DEFAULT_FONTFAMILY;
   this.fontstyle = DEFAULT_FONTSTYLE;
   this.fontweight =
DEFAULT_FONTWEIGHT;
   this.height = DEFAULT_HEIGHT;
   this.label = iLabel;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.rotate = DEFAULT_ROTATE;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.width = DEFAULT_WIDTH;
   this.x = iX;
   this.y = iY;
}

function Frame(iX, iY) {
   this.objecttype = "frame";
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.fontsize = DEFAULT_FONTSIZE;
   this.fontfamily =
DEFAULT_FONTFAMILY;
   this.fontstyle = DEFAULT_FONTSTYLE;
   this.fontweight =
DEFAULT_FONTWEIGHT;
   this.height = DEFAULT_FRAME_H;
   this.label = "Frame Name";
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.rotate = DEFAULT_ROTATE;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.width = DEFAULT_FRAME_W;
   this.x = iX;
   this.y = iY;
}

function Package1(iX, iY) {
   this.objecttype = "package1";
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.fontsize = DEFAULT_FONTSIZE;
   this.fontfamily =
DEFAULT_FONTFAMILY;
   this.fontstyle = DEFAULT_FONTSTYLE;
   this.fontweight =
DEFAULT_FONTWEIGHT;
   this.height = DEFAULT_HEIGHT;
   this.label = "Package";
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.rotate = DEFAULT_ROTATE;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.width = DEFAULT_WIDTH;
   this.x = iX;
   this.y = iY;
}

function Package2(iX, iY) {
   this.objecttype = "package2";
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.fontsize = DEFAULT_FONTSIZE;
   this.fontfamily =
DEFAULT_FONTFAMILY;
   this.fontstyle = DEFAULT_FONTSTYLE;
   this.fontweight =
DEFAULT_FONTWEIGHT;
   this.height = DEFAULT_PACKAGE2_H;
   this.label = "Package";
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.rotate = DEFAULT_ROTATE;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.width = DEFAULT_PACKAGE2_W;
   this.x = iX;
   this.y = iY;
}

function Component(iX, iY) {
   this.objecttype = "component";
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.fontsize = DEFAULT_FONTSIZE;
   this.fontfamily =
DEFAULT_FONTFAMILY;
   this.fontstyle = DEFAULT_FONTSTYLE;
   this.fontweight =
DEFAULT_FONTWEIGHT;
   this.height = DEFAULT_HEIGHT;
   this.label = "Component Name";
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.rotate = DEFAULT_ROTATE;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.width = DEFAULT_WIDTH;
   this.x = iX;
   this.y = iY;
}

function Artifact(iX, iY) {
   this.objecttype = "artifact";
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.fontsize = DEFAULT_FONTSIZE;
   this.fontfamily =
DEFAULT_FONTFAMILY;
   this.fontstyle = DEFAULT_FONTSTYLE;
   this.fontweight =
DEFAULT_FONTWEIGHT;
   this.height = DEFAULT_HEIGHT;
   this.label = "Artifact Name";
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.rotate = DEFAULT_ROTATE;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.width = DEFAULT_WIDTH;
   this.x = iX;
   this.y = iY;
}

function UseCase(iX, iY) {
   this.objecttype = "usecase";
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.fontsize = DEFAULT_FONTSIZE;
   this.fontfamily =
DEFAULT_FONTFAMILY;
   this.fontstyle = DEFAULT_FONTSTYLE;
   this.fontweight =
DEFAULT_FONTWEIGHT;
   this.height = DEFAULT_USECASE_H;
   this.label = "Use Case";
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.rotate = DEFAULT_ROTATE;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.textcolor =
DEFAULT_TEXTCOLOR;
```

```
    this.width = DEFAULT_USECASE_W;
    this.x = iX;
    this.y = iY;
}

function Actor(iX, iY) {
    this.objecttype = "actor";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height = DEFAULT_HEIGHT;
    this.label = "Actor Name";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width = DEFAULT_MEDIUMSIZE;
    this.x = iX;
    this.y = iY;
}

function Superstate(iX, iY) {
    this.objecttype = "superstate";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height =
DEFAULT_SUPERSTATE_H;
    this.label = "Superstate Name";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width =
DEFAULT_SUPERSTATE_W;
    this.x = iX;
    this.y = iY;
}

function Action(iX, iY) {
    this.objecttype = "action";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height = DEFAULT_ACTION_H;
    this.label = "Action Name";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width = DEFAULT_ACTION_W;
    this.x = iX;
    this.y = iY;
}

function Subactivity(iX, iY) {
    this.objecttype = "subactivity";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height =
DEFAULT_SUBACTIVITY_H;
    this.label = "Subactivity Name";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width =
DEFAULT_SUBACTIVITY_W;
    this.x = iX;
    this.y = iY;
}

function TimeSignal(iX, iY) {
    this.objecttype = "timesignal";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height = DEFAULT_HEIGHT;
    this.label = "Time Signal Name";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width = DEFAULT_MEDIUMSIZE;
    this.x = iX;
    this.y = iY;
}

function AcceptSignal(iX, iY) {
    this.objecttype = "acceptsignal";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height =
DEFAULT_ACCEPTSGN_H;
    this.label = "Accept Signal Name";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width =
DEFAULT_ACCEPTSGN_W;
    this.x = iX;
    this.y = iY;
}

function SendSignal(iX, iY) {
    this.objecttype = "sendsignal";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height = DEFAULT_SENDSGN_H;
    this.label = "Send Signal Name";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width = DEFAULT_SENDSGN_W;
    this.x = iX;
    this.y = iY;
}

function Connector(iX, iY) {
    this.objecttype = "connector";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height = DEFAULT_SMALLSIZE;
    this.label =
DEFAULT_CONNECTOR_LABEL;
```

```
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width = DEFAULT_SMALLSIZE;
    this.x = iX;
    this.y = iY;
}

function Transformation(iX, iY) {
    this.objecttype = "transformation";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height =
DEFAULT_TRANSFORMATION_H;
    this.label = "";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width =
DEFAULT_TRANSFORMATION_W;
    this.x = iX;
    this.y = iY;
}

function Part(iX, iY) {
    this.objecttype = "part";
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.height = DEFAULT_PART_H;
    this.label = "Part Name";
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.rotate = DEFAULT_ROTATE;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.width = DEFAULT_PART_W;
    this.x = iX;
    this.y = iY;
}

function Text(iX, iY, iWidth, iHeight,
iLabel) {
    this.objecttype = "text";
    this.x = iX;
    this.y = iY;
    this.width = iWidth;
    this.height = iHeight;
    this.fontstyle = DEFAULT_FONTSTYLE;
    this.fontweight =
DEFAULT_FONTWEIGHT;
    this.fontsize = DEFAULT_FONTSIZE;
    this.fontfamily =
DEFAULT_FONTFAMILY;
    this.label = iLabel;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.rotate = DEFAULT_ROTATE;
}

function Line(iX1, iY1, iX2, iY2) {
    this.objecttype = "line";
    this.x1 = iX1;
    this.y1 = iY1;
    this.x2 = iX2;
    this.y2 = iY2;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.linecap = DEFAULT_LINECAP;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.arrowstyle1 =
DEFAULT_ARROWSTYLE1;
    this.arrowstyle2 =
DEFAULT_ARROWSTYLE2;
    this.linestyle = DEFAULT_LINESTYLE;
}

function Bezier(iX1, iY1, iX2, iY2, iCtrl1x,
iCtrl1y, iCtrl2x, iCtrl2y) {
    this.objecttype = "bezier";
    this.x1 = iX1;
    this.y1 = iY1;
    this.x2 = iX2;
    this.y2 = iY2;
    this.ctrl1x = iCtrl1x;
    this.ctrl1y = iCtrl1y;
    this.ctrl2x = iCtrl2x;
    this.ctrl2y = iCtrl2y;
    this.linejoin = DEFAULT_LINEJOIN;
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.linecap = DEFAULT_LINECAP;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.arrowstyle1 =
DEFAULT_ARROWSTYLE1;
    this.arrowstyle2 =
DEFAULT_ARROWSTYLE2;
    this.linestyle = DEFAULT_LINESTYLE;
}

function Path(iPathPoints) {
    this.objecttype = "path";
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.points = iPathPoints;
}

function ClassNotation(iX, iY) {
    this.objecttype = "class";
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.x = iX;
    this.y = iY;
    this.width = DEFAULT_WIDTH;
    this.height = DEFAULT_HEIGHT;
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.textcolor =
DEFAULT_TEXTCOLOR;
    this.fontsize = DEFAULT_CFONTSIZE;
    this.classfontsize =
DEFAULT_CNFONTSIZE;
    this.classname = "Class Name";
    this.abstractclass = false;
    this.qualifiedassociation = false;
    this.qualifier = "";
    this.stereotype = "";
    this.templateclass = false;
    this.activeclass = false;
    this.attributes = [];
    this.showattributes = true;
    this.operations = [];
    this.showoperations = true;
    this.templates = [];
    this.rotate = DEFAULT_ROTATE;
}

function Polygon(iPoints) {
    this.objecttype = "polygon";
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.fillcolor = DEFAULT_FILLCOLOR;
    this.points = iPoints;
}

function Polyline(iPoints) {
    this.objecttype = "polyline";
    this.linewidth =
DEFAULT_LINEWIDTH;
    this.linecap = DEFAULT_LINECAP;
    this.linejoin = DEFAULT_LINEJOIN;
    this.strokestyle =
DEFAULT_STROKESTYLE;
    this.arrowstyle1 =
DEFAULT_ARROWSTYLE1;
    this.arrowstyle2 =
DEFAULT_ARROWSTYLE2;
    this.linestyle = DEFAULT_LINESTYLE;
    this.points = iPoints;
```

```
}

function Instance(iX, iY) {
   this.objecttype = "instance";
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_ISWIDTH;
   this.height = DEFAULT_ISHEIGHT;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.fontsize = DEFAULT_CFONTSIZE;
   this.classname = "Class Name";
   this.instancename = "Instance Name"
   this.attributes = [];
   this.showattributes = false;
   this.rotate = DEFAULT_ROTATE;
}

function Node(iX, iY) {
   this.objecttype = "node";
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_WIDTH;
   this.height = DEFAULT_HEIGHT;
   this.topheight =
DEFAULT_TOPHEIGHT;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.fontsize = DEFAULT_CFONTSIZE;
   this.nodename = "Node Name";
   this.stereotype = "";
   this.rotate = DEFAULT_ROTATE;
   this.showartifacts = true;
   this.containedartifacts = [];
   this.taggedvalues = [];
}

function Statebox(iX, iY) {
   this.objecttype = "statebox";
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.x = iX;
   this.y = iY;
   this.width = DEFAULT_ISWIDTH;
   this.height = DEFAULT_ISHEIGHT;
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.textcolor =
DEFAULT_TEXTCOLOR;
   this.fontsize = DEFAULT_CFONTSIZE;
   this.statename = "State Name";
   this.internalactivitys = [];
   this.showinternalactivities = true;
   this.rotate = DEFAULT_ROTATE;
}

function ExpansionRegion(iX, iY) {
   this.objecttype = "expansionregion";
   this.fillcolor = DEFAULT_FILLCOLOR;
   this.height = DEFAULT_HEIGHT;
   this.linecap = DEFAULT_LINECAP;
   this.linejoin = DEFAULT_LINEJOIN;
   this.linewidth =
DEFAULT_LINEWIDTH;
   this.rotate = DEFAULT_ROTATE;
   this.strokestyle =
DEFAULT_STROKESTYLE;
   this.width = DEFAULT_WIDTH;
   this.x = iX;
   this.y = iY;
   this.listboxpinsize =
DEFAULT_LISTBOXPINSIZE;
   this.listboxpin1x = 0;
   this.listboxpin1y = this.height * 0.3;
   this.listboxpin2x = this.width * 0.7;
   this.listboxpin2y = this.height;
}

function addNewRectangle(iX, iY, w, h) {
   var temp = new Rectangle(iX, iY, w, h);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewEllipse(iX, iY, w, h) {
   var temp = new Ellipse(iX, iY, w, h);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewText(iX, iY, iWidth,
iHeight, iLabel) {
   var temp = new Text(iX, iY, iWidth,
iHeight, iLabel);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewLine(iX1, iY1, iX2, iY2) {
   var temp = new Line(iX1, iY1, iX2, iY2);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewBezier(iX1, iY1, iX2, iY2,
iCtrl1x, iCtrl1y, iCtrl2x, iCtrl2y) {
   var temp = new Bezier(iX1, iY1, iX2,
iY2, iCtrl1x, iCtrl1y, iCtrl2x, iCtrl2y);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewPath(iPathCoordinates)
{
   var temp = new
Path(iPathCoordinates);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewClass(iX, iY) {
   var temp = new ClassNotation(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewNote(iX, iY, iLabel) {
   var temp = new Note(iX, iY, iLabel);
   diagramObjects.push(temp);
```

```
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewFrame(iX, iY) {
   var temp = new Frame(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewPackage1(iX, iY) {
   var temp = new Package1(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewPackage2(iX, iY) {
   var temp = new Package2(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewComponent(iX, iY) {
   var temp = new Component(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewArtifact(iX, iY) {
   var temp = new Artifact(iX, iY);
   diagramObjects.push(temp);
```

```
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewUseCase(iX, iY) {
   var temp = new UseCase(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewActor(iX, iY) {
   var temp = new Actor(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewSuperstate(iX, iY) {
   var temp = new Superstate(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewAction(iX, iY) {
   var temp = new Action(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewSubactivity(iX, iY) {
   var temp = new Subactivity(iX, iY);
   diagramObjects.push(temp);
```

```
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewTimeSignal(iX, iY) {
   var temp = new TimeSignal(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewAcceptSignal(iX, iY) {
   var temp = new AcceptSignal(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewSendSignal(iX, iY) {
   var temp = new SendSignal(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewConnector(iX, iY) {
   var temp = new Connector(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewTransformation(iX, iY) {
   var temp = new Transformation(iX,
iY);
   diagramObjects.push(temp);
```

```
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewPart(iX, iY) {
    var temp = new Part(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewPolygon(iX, iY) {
    var polygonPts = [];
    polygonPts.push({ "x" : iX, "y" : iY });
    polygonPts.push({ "x" : iX, "y" : iY +
DEFAULT_HEIGHT });
    polygonPts.push({ "x" : iX +
DEFAULT_HEIGHT, "y" : iY +
DEFAULT_HEIGHT  });
    var temp = new Polygon(polygonPts);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewPolyline(iX1, iY1, iX2,
iY2) {
    var polylinePts = [];
    polylinePts.push({ "x" : iX1, "y" : iY1 });
    polylinePts.push({ "x" : iX2, "y" : iY2 });
    var temp = new Polyline(polylinePts);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewInstance(iX, iY) {
    var temp = new Instance(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewNode(iX, iY) {
    var temp = new Node(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewStatebox(iX, iY) {
    var temp = new Statebox(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewExpansionRegion(iX, iY)
{
    var temp = new ExpansionRegion(iX,
iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewArrowhead(iX, iY) {
    var temp = new Arrowhead(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewActivationBar(iX, iY) {
    var temp = new ActivationBar(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewLifeline(iX, iY) {
    var temp = new Lifeline(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewDeletion(iX, iY) {
    var temp = new Deletion(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewInitialPS(iX, iY) {
    var temp = new InitialPS(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewFinalState(iX, iY) {
    var temp = new FinalState(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);

    generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
    generateOp(new noneOperation());
}

function addNewShallowHistPS(iX, iY) {
    var temp = new ShallowHistPS(iX, iY);
    diagramObjects.push(temp);
    noOfDiagramObjects =
diagramObjects.length;
    select(noOfDiagramObjects - 1);
```

```
   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewDeepHistPS(iX, iY) {
   var temp = new DeepHistPS(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewInitialNode(iX, iY) {
   var temp = new InitialNode(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewFinalNode(iX, iY) {
   var temp = new FinalNode(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewBlackBar(iX, iY) {
   var temp = new BlackBar(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewDiamond(iX, iY) {
   var temp = new Diamond(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);
   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewPin(iX, iY) {
   var temp = new Pin(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewFlowFinal(iX, iY) {
   var temp = new FlowFinal(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

function addNewPort(iX, iY) {
   var temp = new Port(iX, iY);
   diagramObjects.push(temp);
   noOfDiagramObjects =
diagramObjects.length;
   select(noOfDiagramObjects - 1);

   generateOp(new
insertOperation(temp,
noOfDiagramObjects - 1));
   generateOp(new noneOperation());
}

// Translate an object into descriptions
or URL code
function translateObj(obj, type) {
   if(type == 1) {
      switch(obj.objecttype) {
         case "rectangle" :
            return "[rectangle (" + obj.x + ",
" + obj.y + ")]";
         case "ellipse" :
            return "[ellipse (" + obj.x + ", "
+ obj.y + ")]";
         case "text" :
            return "[text " + obj.label + "]";
         case "line" :
            return "[line ("+ obj.x1 + ", " +
obj.y1 + ") to (" + obj.x2 +
               ", " + obj.y2 + ")]";
         case "path" :
            return "[path from (" +
obj.points[0].x + ", " + obj.points[0].y
               + ") to (" +
obj.points[obj.points.length-1].x + ", " +
               obj.points[obj.points.length-
1].y + ")]";
         case "class" :
            return "[class]";
         default :
            return "[" + obj.objecttype +
"]";
      }
   }
   else if(type == 2) {
      var temp = "&objecttype=" +
obj.objecttype + "&linewidth=" +
obj.linewidth
         + "&linecap=" + obj.linecap +
"&linejoin=" + obj.linejoin +
"&strokestyle="
         +
encodeURIComponent(obj.strokestyle);

      switch(obj.objecttype) {
         case "rectangle" :
         case "ellipse" :
         case "arrowhead" :
         case "activationbar" :
         case "lifeline" :
         case "deletion" :
         case "initialps" :
         case "finalstate" :
         case "shallowhistps" :
         case "deephistps" :
         case "initialnode" :
         case "finalnode" :
         case "blackbar" :
         case "diamond" :
         case "pin" :
         case "flowfinal" :
         case "port" :
            temp += "&x=" + obj.x + "&y="
+ obj.y + "&width=" + obj.width +
               "&height=" + obj.height +
"&fillcolor=" +
encodeURIComponent(obj.fillcolor)
               + "&rotate=" + obj.rotate;
            break;
         case "text" :
            temp = "&objecttype=text" +
"&fontstyle=" + obj.fontstyle +
               "&fontweight=" +
obj.fontweight + "&fontsize=" +
obj.fontsize
               + "&fontfamily=" +
encodeURIComponent(obj.fontfamily) +
"&x="
               + obj.x + "&y=" + obj.y +
"&width=" + obj.width + "&height="
               + obj.height + "&label=" +
encodeURIComponent(obj.label) +
               "&textcolor=" +
encodeURIComponent(obj.textcolor) +
"&rotate="
               + obj.rotate;
            break;
         case "line" :
```

```
            temp += "&x1=" + obj.x1 +
"&y1=" + obj.y1 + "&x2=" + obj.x2 +
                "&y2=" + obj.y2 +
"&arrowstyle1=" +
encodeURIComponent(obj.arrowstyle1)
                + "&arrowstyle2=" +
encodeURIComponent(obj.arrowstyle2)
+ "&linestyle="
                + obj.linestyle;
            break;
          case "bezier" :
            temp += "&x1=" + obj.x1 +
"&y1=" + obj.y1 + "&x2=" + obj.x2
                + "&y2=" + obj.y2 +
"&ctrl1x=" + obj.ctrl1x + "&ctrl1y=" +
                obj.ctrl1y + "&ctrl2x=" +
obj.ctrl2x + "&ctrl2y=" + obj.ctrl2y
                + "&arrowstyle1=" +
encodeURIComponent(obj.arrowstyle1)
+
                "&arrowstyle2=" +
encodeURIComponent(obj.arrowstyle2)
+
                "&linestyle=" + obj.linestyle;
            break;
          case "path" :
            for(var i = 0; i <
obj.points.length; i++) {
                temp += "&x=" +
obj.points[i].x + "&y=" + obj.points[i].y;
            }
            break;
          case "class" :
            temp += "&x=" + obj.x + "&y="
+ obj.y + "&width=" + obj.width
                + "&height=" + obj.height +
"&fillcolor=" +
encodeURIComponent(obj.fillcolor)
                + "&textcolor=" +
encodeURIComponent(obj.textcolor) +
"&fontsize="
                + obj.fontsize +
"&classfontsize=" + obj.classfontsize +
"&classname="
                +
encodeURIComponent(obj.classname) +
"&abstractclass=" + obj.abstractclass
                + "&qualifiedassociation=" +
obj.qualifiedassociation + "&qualifier="
                +
encodeURIComponent(obj.qualifier) +
"&stereotype=" +

encodeURIComponent(obj.stereotype)
+ "&templateclass=" + obj.templateclass
                + "&activeclass=" +
obj.activeclass + "&showattributes=" +
                obj.showattributes +
"&showoperations=" +
obj.showoperations
                + "&rotate=" + obj.rotate;

            for(var i = 0; i <
obj.attributes.length; i++) {
                temp += "&attributevalue="
+
encodeURIComponent(obj.attributes[i].
value)
                    + "&attributestatic=" +
obj.attributes[i].isstatic;
            }

            for(var i = 0; i <
obj.operations.length; i++) {
                temp += "&operationvalue="
+
encodeURIComponent(obj.operations[i]
.value)
                    + "&operationstatic=" +
obj.operations[i].isstatic +
"&operationabstract"
                    +
obj.operations[i].isabstract;
            }

            for(var i = 0; i <
obj.templates.length; i++) {
                temp += "&templatevalue="
+
encodeURIComponent(obj.templates[i])
;
            }

            break;
          case "note" :
          case "frame" :
          case "package1" :
          case "package2" :
          case "component" :
          case "artifact" :
          case "usecase" :
          case "actor" :
          case "superstate" :
          case "action" :
          case "subactivity" :
          case "timesignal" :
          case "acceptsignal" :
          case "sendsignal" :
          case "connector" :
          case "transformation" :
          case "part" :
            temp += "&fillcolor=" +
encodeURIComponent(obj.fillcolor) +
"&fontsize="
                + obj.fontsize +
"&fontfamily=" + obj.fontfamily +
"&fontstyle="
                + obj.fontstyle +
"&fontweight=" + obj.fontweight +
"&height="
                + obj.height + "&label=" +
encodeURIComponent(obj.label) +
                "&rotate=" + obj.rotate +
"&textcolor=" + obj.textcolor +
                "&width=" + obj.width +
"&x=" + obj.x + "&y=" + obj.y;
            break;
          case "polygon" :
            temp += "&fillcolor=" +
encodeURIComponent(obj.fillcolor);
            for(var i = 0; i <
obj.points.length; i++) {
                temp += "&x=" +
obj.points[i].x + "&y=" + obj.points[i].y;
            }
            break;
          case "polyline" :
            temp += "&arrowstyle1=" +
encodeURIComponent(obj.arrowstyle1)
+
                "&arrowstyle2=" +
encodeURIComponent(obj.arrowstyle2)
+
                "&linestyle=" + obj.linestyle;
            for(var i = 0; i <
obj.points.length; i++) {
                temp += "&x=" +
obj.points[i].x + "&y=" + obj.points[i].y;
            }
            break;
          case "instance" :
            temp += "&x=" + obj.x + "&y="
+ obj.y + "&width=" + obj.width
                + "&height=" + obj.height +
"&fillcolor=" +
encodeURIComponent(obj.fillcolor)
                + "&textcolor=" +
encodeURIComponent(obj.textcolor) +
"&fontsize="
                + obj.fontsize +
"&classname=" +
encodeURIComponent(obj.classname) +
"&instancename="
                +
encodeURIComponent(obj.instancenam
e) + "&showattributes=" +
                obj.showattributes +
"&rotate=" + obj.rotate;

            for(var i = 0; i <
obj.attributes.length; i++) {
                temp += "&attributevalue="
+
encodeURIComponent(obj.attributes[i].
value)
                    + "&attributestatic=" +
obj.attributes[i].isstatic;
            }

            break;
          case "node" :
            temp += "&x=" + obj.x + "&y="
+ obj.y + "&width=" + obj.width
                + "&height=" + obj.height +
"&topheight=" + obj.topheight +
                "&fillcolor=" +
encodeURIComponent(obj.fillcolor) +
"&textcolor="
                +
encodeURIComponent(obj.textcolor) +
"&fontsize=" + obj.fontsize
                + "&nodename=" +
encodeURIComponent(obj.nodename) +
"&stereotype="
                +
encodeURIComponent(obj.stereotype)
+ "&showartifacts=" +
```

```
            obj.showartifacts +
"&rotate=" + obj.rotate;

            for(var i = 0; i <
obj.containedartifacts.length; i++) {
                temp +=
"&containedartifactvalue=" +
encodeURIComponent(obj.containedart
ifacts[i]);
            }

            for(var i = 0; i <
obj.taggedvalues.length; i++) {
                temp +=
"&taggedvaluevalue=" +
encodeURIComponent(obj.taggedvalues
[i]);
            }

            break;
         case "statebox" :
            temp += "&x=" + obj.x + "&y="
+ obj.y + "&width=" + obj.width
                + "&height=" + obj.height +
"&fillcolor=" +
encodeURIComponent(obj.fillcolor)
                + "&textcolor=" +
encodeURIComponent(obj.textcolor) +
"&fontsize="
                + obj.fontsize +
"&statename=" +
encodeURIComponent(obj.statename)
                + "&showinternalactivities="
+ obj.showinternalactivities +
                "&rotate=" + obj.rotate;

            for(var i = 0; i <
obj.internalactivitys.length; i++) {
                temp +=
"&internalactivityvalue=" +
encodeURIComponent(obj.internalactivi
tys[i]);
            }

            break;
         case "expansionregion" :
            temp += "&fillcolor=" +
encodeURIComponent(obj.fillcolor) +
                "&height=" + obj.height +
"&rotate=" + obj.rotate + "&width="
                + obj.width + "&x=" + obj.x +
"&y=" + obj.y + "&listboxpin1x="
                + obj.listboxpin1x +
"&listboxpin1y=" + obj.listboxpin1y +
                "&listboxpin2x=" +
obj.listboxpin2x + "&listboxpin2y=" +
                obj.listboxpin2y +
"&listboxpinsize=" + obj.listboxpinsize;
            break;
      }

      return temp;
   }
}
```

operations.js

```
function insertOperation(obj, pos) {
   this.optype = "insert";
   this.object = obj;
   this.position = pos;
}

function deleteOperation(pos) {
   this.optype = "delete";
   this.position = pos;
}

function editOperation(pos, attr, val,
attrOp) {
   this.optype = "edit";
   this.position = pos;
   this.attribute = attr;
   this.value = val;
   this.attrOp = attrOp;
}

function moveOperation(pos, dest) {
   this.optype = "move";
   this.position = pos;
   this.destination = dest;
}

function noneOperation() {
   this.optype = "none";
}

function apply(op) {
   //alert("applying " + op.optype + "
operation");

   switch(op.optype) {
      case "insert" :

diagramObjects.splice(op.position, 0,
op.object);
         noOfDiagramObjects =
diagramObjects.length;
         break;
      case "delete" :
         if(selectedObj ==
diagramObjects[op.position]) {
            selectedObj = null;
            updatePropertiesBox();
         }

diagramObjects.splice(op.position, 1);
         noOfDiagramObjects =
diagramObjects.length;
         break;
      case "edit" :

if((diagramObjects[op.position].objectty
pe == "path") &&
               (op.attribute == "points")) {
            var temp2 = (op.value).split("
");
            var newPoints = [];
            for(var i = 0; i < temp2.length;
i++) {
               var temp3 =
temp2[i].split(",");
               newPoints.push({"x" :
parseFloat(temp3[0]),
                  "y" :
parseFloat(temp3[1])});
            }

diagramObjects[op.position].points =
newPoints;
         } else if(op.attribute == "x" ||
op.attribute == "y"  ||
               op.attribute == "x1" ||
op.attribute == "y1" ||
               op.attribute == "x2" ||
op.attribute == "y2" ||
               op.attribute == "linewidth"
|| op.attribute == "width" ||
               op.attribute == "height" ||
op.attribute == "topheight" ||
               op.attribute ==
"listboxpin1x" || op.attribute ==
"listboxpin1y" ||
               op.attribute ==
"listboxpin2x" || op.attribute ==
"listboxpin2y" ||
               op.attribute ==
"listboxpinsize" ||op.attribute ==
"fontsize" ||
               op.attribute ==
"classfontsize" || op.attribute ==
"rotate") {

diagramObjects[op.position][op.attribut
e] = parseFloat(op.value);
         } else if(op.attribute ==
"abstractclass" || op.attribute ==
"activeclass"
               || op.attribute ==
"qualifiedassociation" || op.attribute ==
"templateclass"
               || op.attribute ==
"showattributes" || op.attribute ==
"showoperations"
               || op.attribute ==
"showartifacts" || op.attribute ==
"showinternalactivities") {

diagramObjects[op.position][op.attribut
e] = (op.value == "true");
         } else if(op.value == null) {

applyItemOp(diagramObjects[op.positio
n][op.attribute + "s"], op.attrOp,
op.attribute);
         } else {

diagramObjects[op.position][op.attribut
e] = op.value;
         }
         break;
      case "move" :
         tempObject =
diagramObjects.splice(op.position, 1);

diagramObjects.splice(op.destination, 0,
tempObject[0]);
         tempObject = {};
```

```
        break;
      case "removecollaborator" :

removeCollaborator(op.collaborator);

removeCurrentUser(op.collaborator);
        break;
      case "adduser" :
        addUser(op.collaborator);
        break;
      case "removeuser" :
        //alert("removing: " +
op.collaborator);

removeCurrentUser(op.collaborator);
        break;
      case "addcollaborator" :
        addCollaborator(op.collaborator);
        break;
      default:
        break;
   }

   invalidate();
   updatePropertiesBox();
}

function applyItemOp(itemsRoot, op,
item) {
   switch(op.optype) {
      case "insert" :
        itemsRoot.splice(op.position, 0,
op.object);
        break;
      case "delete" :
        itemsRoot.splice(op.position, 1);
        break;
      case "edit" :
        if(op.attribute ==
"attributeValue") {
           if(item == "taggedvalue" ||
item == "containedartifact" || item ==
"internalactivity") {
              itemsRoot[op.position] =
op.value;
           } else {
              itemsRoot[op.position].value
= op.value;
           }
        } else if(op.attribute ==
"attributeStatic") {
           itemsRoot[op.position].isstatic
= eval(op.value);
        } else if(op.attribute ==
"attributeAbstract") {

itemsRoot[op.position].isabstract =
eval(op.value);
        } else if(op.attribute ==
"attributeX") {
           itemsRoot[op.position].x =
parseFloat(op.value);
        } else if(op.attribute ==
"attributeY") {
           itemsRoot[op.position].y =
parseFloat(op.value);
        }
        break;
      case "move" :
        tempObject =
itemsRoot.splice(op.position, 1);
        itemsRoot.splice(op.destination,
0, tempObject[0]);
        tempObject = {};
        break;
      default:
        break;
   }
}

function transform(op1, op2) {
   switch(op1.optype) {
      case "insert" :
        switch(op2.optype) {
           case "insert" :
              if(op1.position <
op2.position)
                 op2.position++;
              else
                 op1.position++;
              break;
           case "delete" :
              if(op1.position >
op2.position)
                 op1.position--;
              else
                 op2.position++;
              break;
           case "edit" :
              if(op1.position <=
op2.position)
                 op2.position++;
              break;
           case "move" :
              if(op1.position >
op2.position) {
                 if(op1.position >
op2.destination) {
                    //do nothing
                 } else {
                    op1.position--;
                    op2.destination++;
                 }
              } else if(op1.position >
op2.destination) {
                 op1.position++;
                 op2.position++;
              } else {
                 op2.position++;
                 op2.destination++;
              }
              break;
           default :
              break;
        }
        break;
      case "delete" :
        switch(op2.optype) {
           case "insert" :
              if(op1.position <
op2.position)
                 op2.position--;
              else
                 op1.position++;
              break;
           case "delete" :
              if(op1.position <
op2.position)
                 op2.position--;
              else if(op1.position >
op2.position)
                 op1.position--;
              else {
                 op1.optype = "none";
                 op2.optype = "none";
              }
              break;
           case "edit" :
              if(op1.position <
op2.position)
                 op2.position--;
              else if(op1.position ==
op2.position)
                 op2.optype = "none";
              break;
           case "move" :
              if(op1.position ==
op2.position) {
                 op1.position =
op2.destination;
                 op2.optype = "none";
              } else if(op1.position <
op2.position) {
                 if(op1.position <
op2.destination) {
                    op2.position--;
                    op2.destination--;
                 } else {
                    op1.position++;
                    op2.position--;
                 }
              } else if(op1.position <=
op2.destination) {
                 op1.position--;
                 op2.destination--;
              } else {
                 //do nothing
              }
              break;
           default :
              break;
        }
        break;
      case "edit" :
        switch(op2.optype) {
           case "insert" :
              if(op1.position >=
op2.position)
                 op1.position++;
              break;
           case "delete" :
              if(op1.position >
op2.position)
                 op1.position--;
              else if(op1.position ==
op2.position)
                 op1.optype = "none";
              break;
```

```
            case "edit" :
               if((op1.position ==
op2.position) && (op1.attribute ==
op2.attribute)) {
                  if(op1.value == null &&
op2.value == null) {
                     transform(op1.attrOp,
op2.attrOp);
                  } else if(op1.value != null
&& op2.value != null) {
                     op1.optype = "none";
                  }
               }
               break;
            case "move" :
               if(op1.position ==
op2.position) {
                  op1.position =
op2.destination;
               } else if(op1.position >=
op2.destination &&
                     op1.position <
op2.position) {
                  op1.position++;
               } else if(op1.position <=
op2.destination &&
                     op1.position >
op2.position) {
                  op1.position--;
               } else {
                  //do nothing
               }
               break;
            default :
               break;
         }
         break;
      case "move" :
         switch(op2.optype) {
            case "insert" :
               if(op2.position >
op1.position) {
                  if(op2.position >
op1.destination) {
                     //do nothing
                  } else {
                     op2.position--;
                     op1.destination++;
                  }
               } else if(op2.position >
op1.destination) {
                  op2.position++;
                  op1.position++;
               } else {
                  op1.position++;
                  op1.destination++;
               }
               break;
            case "delete" :
               if(op2.position ==
op1.position) {
                  op2.position =
op1.destination;
                  op1.optype = "none";
               } else if(op2.position <
op1.position) {
                  if(op2.position <
op1.destination) {
                     op1.position--;
                     op1.destination--;
                  } else {
                     op2.position++;
                     op1.position--;
                  }
               } else if(op2.position <=
op1.destination) {
                  op2.position--;
                  op1.destination--;
               } else {
                  //do nothing
               }
               break;
            case "edit" :
               if(op2.position ==
op1.position) {
                  op2.position =
op1.destination;
               } else if(op2.position >=
op1.destination &&
                     op2.position <
op1.position) {
                  op2.position++;
               } else if(op2.position <=
op1.destination &&
                     op2.position >
op1.position) {
                  op2.position--;
               } else {
                  //do nothing
               }
               break;
            case "move" :
               var p = op1.position;
               var q = op1.destination;
               var r = op2.position;
               var s = op2.destination;

               // 1
               if(p == r) {
                  if(q == s) {
                     op1.optype = "none";
                     op2.optype = "none";
                     return;
                  } else if(p == q) {
                     op1.position =  s;
                     op1.destination = s;
                     return;
                  } else if(r == s) {
                     op2.position = q;
                     op2.destination = q;
                     return;
                  } else {
                     op1.optype = "none";
                     op2.position = q;
                     return;
                  }
               }

               // 2
               if(p > r) {
                  if(p <= s) op1.position--;
                  if(r >= q) op2.position++;
               } else {
                  if(p >= s) op1.position++;
                  if(r <= q) op2.position--;
               }

               // 3
               if(q > r) {
                  if(q <= s) op1.destination--;
               } else if(q < r) {
                  if(q >= s)
op1.destination++;
               } else { // q == r
                  if(p < q && r < s)
op1.destination--;
                  else if(p > q && r > s)
op1.destination++;
               }

               //4
               if(s > p && s < q)
op2.destination--;
               else if(s < p && s > q)
op2.destination++;
               else {
                  if(p == s) {
                     if(r < p && p < q)
op2.destination--;
                     else if(r > p && p > q)
op2.destination++;
                  } else if(q == s) {
                     if(p < q && q <= r)
op2.destination--;
                     else if(p > q && q >= r)
op2.destination++;
                  }
               }

               break;
            default:
               break;
         }
         break;
      default :
         break;
   }
}

function generateResizeOp() {
   switch(selectedObj.objecttype) {
      case "rectangle" :
      case "ellipse" :
      case "text" :
      case "class" :
      case "note" :
      case "instance" :
      case "node" :
      case "statebox" :
      case "expansionregion" :
      case "arrowhead" :
      case "activationbar" :
      case "lifeline" :
      case "deletion" :
      case "initialps" :
      case "finalstate" :
      case "shallowhistps" :
      case "deephistps" :
```

```
    case "initialnode" :
    case "finalnode" :
    case "blackbar" :
    case "diamond" :
    case "pin" :
    case "flowfinal" :
    case "port" :
    case "frame" :
    case "package1" :
    case "package2" :
    case "component" :
    case "artifact" :
    case "usecase" :
    case "actor" :
    case "superstate" :
    case "action" :
    case "subactivity" :
    case "timesignal" :
    case "acceptsignal" :
    case "sendsignal" :
    case "connector" :
    case "transformation" :
    case "part" :
      switch(expectResize) {
        case 0 :
        case 2 :
        case 5 :
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "x", selectedObj.x, null));
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "y", selectedObj.y, null));
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "width",
selectedObj.width, null));
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "height",
selectedObj.height, null));
          break;
        case 1 :
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "x", selectedObj.x, null));
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "y", selectedObj.y, null));
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "height",
selectedObj.height, null));
          break;
        case 3 :
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "x", selectedObj.x, null));
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "y", selectedObj.y, null));
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "width",
selectedObj.width, null));
          break;
        case 4 :
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "width",
selectedObj.width, null));
          break;
        case 6 :
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "height",
selectedObj.height, null));
          break;
        case 7 :
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "width",
selectedObj.width, null));
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "height",
selectedObj.height, null));
          break;
        case 8 :
          if(selectedObj.objecttype ==
"node") {
            generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
              "topheight",
selectedObj.topheight, null));
          } else
if(selectedObj.objecttype ==
"expansionregion") {
            generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
              "listboxpin1x",
selectedObj.listboxpin1x, null));
            generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
              "listboxpin1y",
selectedObj.listboxpin1y, null));
          }
          break;
        case 9 :
          if(selectedObj.objecttype ==
"expansionregion") {
            generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
              "listboxpin2x",
selectedObj.listboxpin2x, null));
            generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
              "listboxpin2y",
selectedObj.listboxpin2y, null));
          }
      }
      break;
    case "line" :
    case "bezier" :
      if(expectResize == 0) {
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "x1", selectedObj.x1,
null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "y1", selectedObj.y1,
null));
      } else if(expectResize == 1) {
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "x2", selectedObj.x2,
null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "y2", selectedObj.y2,
null));
      } else if(expectResize == 2) {
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "ctrl1x",
selectedObj.ctrl1x, null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "ctrl1y",
selectedObj.ctrl1y, null));
      } else if(expectResize == 3) {
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "ctrl2x",
selectedObj.ctrl2x, null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "ctrl2y",
selectedObj.ctrl2y, null));
      }
      break;
    case "polygon" :
    case "polyline" :
      var attrOp = new
editOperation(expectResize,
"attributeX",
selectedObj.points[expectResize].x,
null);
```

```
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), "point", null, attrOp));
        attrOp = new
editOperation(expectResize,
"attributeY",
selectedObj.points[expectResize].y,
null);
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), "point", null, attrOp));
        break;
      case "text" :
        break;
      default :
        break;
    }
    generateOp(new noneOperation());
}

function generateDragOp() {
    switch(selectedObj.objecttype) {
      case "rectangle" :
      case "ellipse" :
      case "text" :
      case "class" :
      case "note" :
      case "instance" :
      case "node" :
      case "statebox" :
      case "expansionregion" :
      case "arrowhead" :
      case "activationbar" :
      case "lifeline" :
      case "deletion" :
      case "initialps" :
      case "finalstate" :
      case "shallowhistps" :
      case "deephistps" :
      case "initialnode" :
      case "finalnode" :
      case "blackbar" :
      case "diamond" :
      case "pin" :
      case "flowfinal" :
      case "port" :
      case "frame" :
      case "package1" :
      case "package2" :
      case "component" :
      case "artifact" :
      case "usecase" :
      case "actor" :
      case "superstate" :
      case "action" :
      case "subactivity" :
      case "timesignal" :
      case "acceptsignal" :
      case "sendsignal" :
      case "connector" :
      case "transformation" :
      case "part" :
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "x", selectedObj.x, null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "y", selectedObj.y, null));
        break;
      case "bezier" :
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "ctrl1x", selectedObj.ctrl1x,
null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "ctrl1y", selectedObj.ctrl1y,
null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "ctrl2x", selectedObj.ctrl2x,
null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "ctrl2y", selectedObj.ctrl2y,
null));
      case "line" :
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "x1", selectedObj.x1, null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "y1", selectedObj.y1, null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "x2", selectedObj.x2, null));
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "y2", selectedObj.y2, null));
        break;
      case "path" :
        var newPoints = "";
        for(var i = 0; i <
selectedObj.points.length; i++) {
          if(i != 0)
            newPoints += " ";

          newPoints = newPoints +
selectedObj.points[i].x
            + "," +
selectedObj.points[i].y;
        }
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
            "points", newPoints, null));
        break;
      case "polygon" :
      case "polyline" :
        for(var i = 0; i <
selectedObj.points.length; i++) {
          var attrOp = new
editOperation(i, "attributeX",
selectedObj.points[i].x, null);
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), "point", null, attrOp));
          attrOp = new editOperation(i,
"attributeY", selectedObj.points[i].y,
null);
          generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), "point", null, attrOp));
        }
        break;
    }

    generateOp(new noneOperation());
}

function generateAddPointOp(x, y) {
    switch(selectedObj.objecttype) {
      case "polygon" :
        // Find out the closest points to
the selected point on each edge
        var noOfPts =
selectedObj.points.length;
        var minDistance;
        var closestCorner;
        for(var i = 0; i < noOfPts; i++) {
          var x1 = selectedObj.points[i].x;
          var y1 = selectedObj.points[i].y;
          var x2 = selectedObj.points[(i +
1) % noOfPts].x;
          var y2 = selectedObj.points[(i +
1) % noOfPts].y;
          var u = ((x - x1) * (x2 - x1) + (y -
y1) * (y2 - y1)) /
                  ((x2 - x1) * (x2 - x1) + (y2
- y1) * (y2 - y1));

          var dist;
          if(u < 0) {
            dist = Math.sqrt((x - x1) * (x -
x1) + (y - y1) * (y - y1));
          } else if(u > 1) {
            dist = Math.sqrt((x - x2) * (x -
x2) + (y - y2) * (y - y2));
          } else {
            var xu = x1 + u * (x2 - x1);
            var yu = y1 + u * (y2 - y1);
            dist = Math.sqrt((x - xu) * (x -
xu) + (y - yu) * (y - yu));
          }

          if(i == 0) {
            minDistance = dist;
            closestCorner = i;
          } else if(dist < minDistance) {
            minDistance = dist;
            closestCorner = i;
          }
        }
        // Insert the new polygon corner
between the endpoints of the closest
edge
        var newPoint = {"x" : x, "y" : y};
```

```
selectedObj.points.splice((closestCorner
+ 1) % noOfPts, 0, newPoint);

        // Generate and send operations
        var attrOp = new
insertOperation(newPoint,
(closestCorner + 1) % noOfPts);
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), "point", null, attrOp));
        generateOp(new
noneOperation());


document.getElementById("polygon_ad
dpoint_message").setAttribute("style",
"display: none;");

document.getElementById("polygon_ad
dpoint").disabled = false;

document.getElementById("polygon_re
movepoint").disabled = false;
        break;
      case "polyline" :
        var noOfPts =
selectedObj.points.length;

        // Find the closest corner from
the clicked point
        var minDistance;
        var closestCorner;
        for(var i = 0; i < noOfPts; i++) {
          var xc = selectedObj.points[i].x;
          var yc = selectedObj.points[i].y;

          var dist = Math.sqrt((x - xc) * (x
- xc) + (y - yc) * (y - yc));

          if(i == 0) {
            minDistance = dist;
            closestCorner = i;
          } else if(dist < minDistance) {
            minDistance = dist;
            closestCorner = i;
          }
        }

        // Insert the new point after the
closest corner
        var newPoint = {"x" : x, "y" : y};

selectedObj.points.splice(closestCorner
+ 1, 0, newPoint);

        // Generate and send operations
        var attrOp = new
insertOperation(newPoint,
closestCorner + 1);
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), "point", null, attrOp));
        generateOp(new
noneOperation());

document.getElementById("polyline_ad
dpoint_message").setAttribute("style",
"display: none;");

document.getElementById("polyline_ad
dpoint").disabled = false;

document.getElementById("polyline_re
movepoint").disabled = false;
        break
      default :
        break;
    }

    pointOperation = "none";
    invalidate();
}

function generateRemovePointOp(x, y) {
    switch(selectedObj.objecttype) {
      case "polygon" :
        if(selectedObj.points.length <= 3)
return;
      case "polyline" :
        var noOfPts =
selectedObj.points.length;
        if(noOfPts <= 2) return;

        // Find the closest corner from
the clicked point
        var minDistance;
        var closestCorner;
        for(var i = 0; i < noOfPts; i++) {
          var xc = selectedObj.points[i].x;
          var yc = selectedObj.points[i].y;

          var dist = Math.sqrt((x - xc) * (x
- xc) + (y - yc) * (y - yc));

          if(i == 0) {
            minDistance = dist;
            closestCorner = i;
          } else if(dist < minDistance) {
            minDistance = dist;
            closestCorner = i;
          }
        }
        // Delete the nearest corner

selectedObj.points.splice(closestCorner,
1);

        // Generate and send operations
        var attrOp = new
deleteOperation(closestCorner);
        generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), "point", null, attrOp));
        generateOp(new
noneOperation());


document.getElementById(selectedObj.
objecttype +
"_removepoint_message").setAttribute(
"style", "display: none;");

document.getElementById(selectedObj.
objecttype + "_addpoint").disabled =
false;

document.getElementById(selectedObj.
objecttype + "_removepoint").disabled
= false;
        break;
      default :
        break;
    }

    pointOperation = "none";
    invalidate();
}

function translateOp(op, type) {
    // For translating operations into
understandable descriptions
    if(type == 1) {
      switch(op.optype) {
        case 'insert' :
          return "You inserted a(n) " +
op.object.objecttype;
        case 'delete' :
          return "delete @" +
op.position;
        case 'edit' :
          if(op.value == null)
            return "edit " + op.attribute
+ " of  @" + op.position + " : "
              + translateOp(op.attrOp,
3);
          else
            return "edit " + op.attribute
+ " to " + op.value + " @" + op.position;
        case 'move' :
          return "move @" + op.position
+ " to " + op.destination;
        case 'removecollaborator' :
          return op.collaborator + " is no
longer a collaborator";
        case 'adduser' :
          return op.collaborator + " has
joined the session";
        case 'removeuser' :
          return op.collaborator + " has
left the session";
        case 'addcollaborator' :
          return
op.collaborator.username + " can now
edit the diagram";
        default :
          return "none";
      }
    }
    // For translating operations on
objects into URL code
    else if(type == 2) {
      switch(op.optype) {
        case 'insert' :
```

```
            return "&optype=insert" +
translateObj(op.object, 2) +
"&position=" + op.position;
         case 'delete' :
            return
"&optype=delete&position=" +
op.position;
         case 'edit' :
            if(op.value == null) {
               return
"&optype=edit&attribute=" +
op.attribute + "&attrOp="
                     +
translateItemOp(op.attrOp,
op.attribute) + "&position=" +
op.position;
            } else {
               return
"&optype=edit&attribute=" +
op.attribute + "&value="
                  +
encodeURIComponent(op.value) +
"&position=" + op.position;
            }
         case 'move' :
            return
"&optype=move&position=" +
op.position + "&destination=" +
op.destination;
         default :
            return "&optype=none";
      }
   }
   // For translating operations on class
attributes / operations / templates into
descriptions
   else if(type == 3) {
      switch(op.optype) {
         case 'insert' :
            return "inserted " +
op.object.value + " on item #" +
op.position;
         case 'delete' :
            return "deleted item #" +
op.position;
         case 'edit' :
            return "edited item #" +
op.position + " to " + op.value;
         case 'move' :
            return "moved item #" +
op.position + " to #" + op.destination;
         default :
            return "";
      }
   }
}

function translateItemOp(op, item) {
   // For translating operations on
polygon points into URL code
   switch(op.optype) {
      case 'insert' :
         if(item == "point") {
            return "insert&attrX=" +
op.object.x + "&attrY=" + op.object.y
               + "&attrPos=" + op.position;
         } else if(item == "attribute" ||
item == "operation" || item ==
"template") {
            return "insert&attrVal=" +
encodeURIComponent(op.object.value)
               + "&attrPos=" + op.position;
         } else if(item ==
"containedartifact" || item ==
"taggedvalue" || item ==
"internalactivity") {
            return "insert&attrVal=" +
encodeURIComponent(op.object)
               + "&attrPos=" + op.position;
         }
         break;
      case 'delete' :
         return "delete&attrPos=" +
op.position;
      case 'edit' :
         return "edit&attrAttr=" +
op.attribute + "&attrVal=" +

encodeURIComponent(op.value) +
"&attrPos=" + op.position;
      case 'move' :
         return "move&attrPos=" +
op.position + "&attrDest=" +
op.destination;
      default :
         return "";
   }
}
```

---

properties box 1.js

```
var OBJ_TYPE = ["rectangle", "ellipse",
"text", "line", "path", "class", "note",
            "bezier", "polygon", "polyline",
"instance", "node", "statebox",
            "expansionregion",
"arrowhead", "activationbar", "lifeline",
"deletion",
            "initialps", "finalstate",
"shallowhistps", "deephistps",
"initialnode",
            "finalnode", "blackbar",
"diamond", "pin", "flowfinal", "port",
            "frame", "package1",
"package2", "component", "artifact",
"usecase",
            "actor", "superstate", "action",
"subactivity", "timesignal",
"acceptsignal",
            "sendsignal", "connector",
"transformation", "part"];
var OBJ_STYLE = {"rectangle" : ["x", "y",
"width", "height", "linewidth",
"linecap",
               "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "ellipse" : ["x", "y", "width",
"height", "linewidth", "linecap",
               "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "text" : ["label", "x", "y",
"width", "height", "textcolor",
"fontstyle",
               "fontweight", "fontsize",
"fontfamily", "rotate"],
            "line" : ["arrowstyle1",
"arrowstyle2", "linestyle", "x1",
               "y1", "x2", "y2", "linewidth",
"linecap", "linejoin",
               "strokestyle"],
            "path" : ["strokestyle",
"linewidth", "linecap", "linejoin"],
            "class" : ["fontsize", "x", "y",
"width", "height", "linewidth",
               "linecap", "linejoin",
"strokestyle", "fillcolor",
               "classfontsize", "textcolor",
"rotate", "showattributes",
               "showoperations"],
            "note" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                  "fontsize",
"fontfamily", "label", "textcolor",
                  "linewidth", "linecap",
"linejoin", "strokestyle",
                  "fillcolor", "rotate"],
            "bezier" : ["arrowstyle1",
"arrowstyle2", "linestyle", "x1",
               "y1", "x2", "y2", "ctrl1x",
"ctrl1y", "ctrl2x", "ctrl2y",
               "linewidth", "linecap",
"linejoin", "strokestyle"],
            "polygon" : ["strokestyle",
"linewidth", "linecap", "linejoin",
"fillcolor"],
            "polyline" : ["arrowstyle1",
"arrowstyle2", "linestyle", "linewidth",
               "linecap", "linejoin",
"strokestyle"],
            "instance" : ["fontsize", "x",
"y", "width", "height", "linewidth",
               "linecap", "linejoin",
"strokestyle", "fillcolor",
               "textcolor", "rotate"],
            "node" : ["fontsize", "x", "y",
"width", "height", "topheight",
               "linewidth", "linecap",
"linejoin", "strokestyle", "fillcolor",
               "textcolor", "rotate"],
            "statebox" : ["fontsize", "x",
"y", "width", "height", "linewidth",
               "linecap", "linejoin",
"strokestyle", "fillcolor",
               "textcolor", "rotate"],
            "expansionregion" : ["x", "y",
"width", "height", "linewidth",
               "linecap", "linejoin",
"strokestyle", "fillcolor", "rotate",
               "listboxpin1x",
"listboxpin1y", "listboxpin2x",
"listboxpin2y",
               "listboxpinsize"],
            "arrowhead" : ["x", "y",
"width", "height", "linewidth",
"linecap",
```

```
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "activationbar" : ["x", "y",
"width", "height", "linewidth",
"linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "lifeline" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "deletion" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "initialps" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "finalstate" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "shallowhistps" : ["x", "y",
"width", "height", "linewidth",
"linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "deephistps" : ["x", "y",
"width", "height", "linewidth",
"linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "initialnode" : ["x", "y",
"width", "height", "linewidth",
"linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "finalnode" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "blackbar" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "diamond" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "pin" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "flowfinal" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "port" : ["x", "y", "width",
"height", "linewidth", "linecap",
            "linejoin", "strokestyle",
"fillcolor", "rotate"],
            "frame" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "package1" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "package2" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "component" : ["x", "y",
"width", "height", "fontstyle",
"fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "artifact" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "usecase" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "actor" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "superstate" : ["x", "y",
"width", "height", "fontstyle",
"fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "action" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "subactivity" : ["x", "y",
"width", "height", "fontstyle",
"fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "timesignal" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "acceptsignal" : ["x", "y",
"width", "height", "fontstyle",
"fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "sendsignal" : ["x", "y",
"width", "height", "fontstyle",
"fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "connector" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "transformation" : ["x", "y",
"width", "height", "fontstyle",
"fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"],
            "part" : ["x", "y", "width",
"height", "fontstyle", "fontweight",
                "fontsize",
"fontfamily", "label", "textcolor",
                "linewidth", "linecap",
"linejoin", "strokestyle",
                "fillcolor", "rotate"]
            };
var OBJ_PROPERTIES = {"rectangle" : [],
"ellipse" : [], "text" : [], "line" : [],
                "path" : [], "class" :
["classname", "abstractclass",
"templateclass",
                "activeclass",
"qualifiedassociation", "qualifier",
"stereotype",
                "showattributes",
"showoperations"], "note" : [], "bezier" :
[],
                "polygon" : [], "polyline" : [],
"instance" : ["instancename",
                "classname",
"showattributes"], "node" :
["nodename", "stereotype",
```

```
                "showartifacts"], "statebox"
: ["statename",
"showinternalactivities"],
                "expansionregion" : [],
"arrowhead" : [], "activationbar" : [],
                "lifeline" : [], "deletion" : [],
"initialps" : [], "finalstate" : [],
                "shallowhistps" : [],
"deephistps" : [], "initialnode" : [],
                "finalnode" : [], "blackbar" :
[], "diamond" : [], "pin" : [],
                "flowfinal" : [], "port" : [],
"frame" : [], "package1" : [],
                "package2" : [],
"component" : [], "artifact" : [],
"usecase" : [],
                "actor" : [], "superstate" : [],
"action" : [], "subactivity" : [],
                "timesignal" : [],
"acceptsignal" : [], "sendsignal" : [],
"connector" : [],
                "transformation" : [], "part"
: []
              };

var selectedIndex; // index of the
currently selected attribute, operation
or template
var itemlist = {};

function propertiesBox_init() {
   itemlist["attribute"] =
(document.getElementById("attributesL
ist"));
   itemlist["operation"] =
(document.getElementById("operations
List"));
   itemlist["template"] =
(document.getElementById("templatesL
ist"));
   itemlist["containedartifact"] =
(document.getElementById("containeda
rtifactsList"));
   itemlist["taggedvalue"] =
(document.getElementById("taggedvalu
esList"));
   itemlist["internalactivity"] =
(document.getElementById("internalact
ivitiesList"));
}

function hideAllPropertiesBoxTable() {
   var i = 0;
   while(OBJ_TYPE[i]) {

(document.getElementById(OBJ_TYPE[i]
+ "StyleTable")).setAttribute(
         'style', 'display: none;');

(document.getElementById(OBJ_TYPE[i]
+ "PropertiesTable")).setAttribute(
         'style', 'display: none;');
      i++;
   }

(document.getElementById("attributesT
able")).setAttribute('style', 'display:
none;');

(document.getElementById("operations
Table")).setAttribute('style', 'display:
none;');

(document.getElementById("templatesT
able")).setAttribute('style', 'display:
none;');

(document.getElementById("containeda
rtifactsTable")).setAttribute('style',
'display: none;');

(document.getElementById("taggedvalu
esTable")).setAttribute('style', 'display:
none;');

(document.getElementById("internalact
ivitiesTable")).setAttribute('style',
'display: none;');


(document.getElementById("styleTab2")
).removeAttribute('class');

(document.getElementById("properties
Tab2")).removeAttribute('class');

(document.getElementById("attributesT
ab2")).removeAttribute('class');

(document.getElementById("operations
Tab2")).removeAttribute('class');

(document.getElementById("templatesT
ab2")).removeAttribute('class');

(document.getElementById("taggedvalu
esTab2")).removeAttribute('class');

(document.getElementById("artifactsTa
b2")).removeAttribute('class');

(document.getElementById("internalact
ivitiesTab2")).removeAttribute('class');
}

function hideAllPropertiesBoxTab() {

(document.getElementById("styleTab"))
.setAttribute('style', 'display: none;');

(document.getElementById("properties
Tab")).setAttribute('style', 'display:
none;');

(document.getElementById("attributesT
ab")).setAttribute('style', 'display:
none;');

(document.getElementById("operations
Tab")).setAttribute('style', 'display:
none;');

(document.getElementById("templatesT
ab")).setAttribute('style', 'display:
none;');

(document.getElementById("taggedvalu
esTab")).setAttribute('style', 'display:
none;');

(document.getElementById("artifactsTa
b")).setAttribute('style', 'display:
none;');

(document.getElementById("internalact
ivitiesTab")).setAttribute('style', 'display:
none;');
}

function updatePropertiesBox() {
   hideAllPropertiesBoxTable();
   hideAllPropertiesBoxTab();

   if(selectedObj == null) return;

   if(OBJ_TYPE.some(function (elt)
{return (elt ==
selectedObj.objecttype);})) {
      //propertiesBoxTab();


(document.getElementById(selectedObj
.objecttype +
         "StyleTable")).setAttribute('style',
'display: block;');


(document.getElementById("styleTab"))
.setAttribute('style', 'display: inline;');
      var style;
      var i = 0;
      while(style =
OBJ_STYLE[selectedObj.objecttype][i++]
) {
         if(style == "strokestyle" || style
== "fillcolor" || style == "textcolor") {
            var rgba =
selectedObj[style].split(",");

/*(document.getElementById(selectedO
bj.objecttype + "_" + style + "_R")).value
=
               rgba[0];

(document.getElementById(selectedObj
.objecttype + "_" + style + "_G")).value =
               rgba[1];

(document.getElementById(selectedObj
.objecttype + "_" + style + "_B")).value =
               rgba[2];*/
            var r =
parseInt(rgba[0]).toString(16);
            var g =
parseInt(rgba[1]).toString(16);
            var b =
parseInt(rgba[2]).toString(16);
```

```
            r = (r.length < 2) ? '0' + r : r;
            g = (g.length < 2) ? '0' + g : g;
            b = (b.length < 2) ? '0' + b : b;


(document.getElementById(selectedObj
.objecttype + "_" +
style)).color.fromString(r + g + b);

(document.getElementById(selectedObj
.objecttype + "_" + style + "_A")).value =
                rgba[3];
          } else {

(document.getElementById(selectedObj
.objecttype + "_" + style)).value =
                selectedObj[style];
          }
       }

       // z-order property

(document.getElementById(selectedObj
.objecttype + "_z-order")).value =

diagramObjects.indexOf(selectedObj);


if(OBJ_PROPERTIES[selectedObj.objectt
ype].length > 0) {

(document.getElementById("properties
Tab")).setAttribute('style', 'display:
inline;');
          var property;
          i = 0;
          while(property =
OBJ_PROPERTIES[selectedObj.objecttyp
e][i++]) {

switch((document.getElementById(selec
tedObj.objecttype + "_" +
property)).type) {
                case "text" :

(document.getElementById(selectedObj
.objecttype + "_" + property)).value =
                     selectedObj[property];
                   break;
                case "checkbox" :

(document.getElementById(selectedObj
.objecttype + "_" + property)).checked =
                     selectedObj[property];
                   break;
                default :
                   break;
             }
          }
       }

       if(selectedObj.objecttype == "class"
|| selectedObj.objecttype ==
"instance") {

(document.getElementById("attributesT
ab")).setAttribute('style', 'display:
inline;');
          itemUnselect("attribute");

itemlist["attribute"].options.length = 0;
          for(i =
selectedObj.attributes.length - 1; i >= 0;
i--) {
             itemlist["attribute"].options[i]
= new
Option(selectedObj.attributes[i].value,
                "", false, false);
          }
       }

       if(selectedObj.objecttype ==
"class") {

(document.getElementById("operations
Tab")).setAttribute('style', 'display:
inline;');
          itemUnselect("operation");

itemlist["operation"].options.length = 0;
          for(i =
selectedObj.operations.length - 1; i >=
0; i--) {
             itemlist["operation"].options[i]
= new
Option(selectedObj.operations[i].value,
                "", false, false);
          }


(document.getElementById("templatesT
ab")).setAttribute('style', 'display:
inline;');
          itemUnselect("template");

itemlist["template"].options.length = 0;
          for(i =
selectedObj.templates.length - 1; i >= 0;
i--) {
             itemlist["template"].options[i]
= new
Option(selectedObj.templates[i].value,
                "", false, false);
          }
       }

       if(selectedObj.objecttype ==
"polygon" || selectedObj.objecttype ==
"polyline") {

document.getElementById(selectedObj.
objecttype +
"_addpoint_message").setAttribute("sty
le", "display: none");

document.getElementById(selectedObj.
objecttype +
"_removepoint_message").setAttribute(
"style", "display: none");

document.getElementById(selectedObj.
objecttype + "_addpoint").disabled =
false;

document.getElementById(selectedObj.
objecttype + "_removepoint").disabled
= false;
          expectingInputClick = false;
          pointOperation = "none";
       }

       if(selectedObj.objecttype ==
"node") {

(document.getElementById("taggedvalu
esTab")).setAttribute('style', 'display:
inline;');
          itemUnselect("taggedvalue");

itemlist["taggedvalue"].options.length =
0;
          for(i =
selectedObj.taggedvalues.length - 1; i >=
0; i--) {

itemlist["taggedvalue"].options[i] = new
Option(selectedObj.taggedvalues[i],
                "", false, false);
          }


(document.getElementById("artifactsTa
b")).setAttribute('style', 'display:
inline;');

itemUnselect("containedartifact");

itemlist["containedartifact"].options.len
gth = 0;
          for(i =
selectedObj.containedartifacts.length -
1; i >= 0; i--) {

itemlist["containedartifact"].options[i] =
new
Option(selectedObj.containedartifacts[i]
,
                "", false, false);
          }
       }

       if(selectedObj.objecttype ==
"statebox") {

(document.getElementById("internalact
ivitiesTab")).setAttribute('style', 'display:
inline;');
          itemUnselect("internalactivity");

itemlist["internalactivity"].options.lengt
h = 0;
          for(i =
selectedObj.internalactivitys.length - 1; i
>= 0; i--) {

itemlist["internalactivity"].options[i] =
```

```
new
Option(selectedObj.internalactivitys[i],
                "", false, false);
          }
       }

       styleTab();
    }
}

// updates only x, y, width, and height in
the properties box
function updatePropertiesBox2() {
   switch(selectedObj.objecttype) {
       case "node" :

(document.getElementById("node_toph
eight")).value = selectedObj.topheight;
       case "rectangle" :
       case "ellipse" :
       case "text" :
       case "note" :
       case "class" :
       case "instance" :
       case "statebox" :
       case "arrowhead" :
       case "activationbar" :
       case "lifeline" :
       case "deletion" :
       case "initialps" :
       case "finalstate" :
       case "shallowhistps" :
       case "deephistps" :
       case "initialnode" :
       case "finalnode" :
       case "blackbar" :
       case "diamond" :
       case "pin" :
       case "flowfinal" :
       case "port" :
       case "frame" :
       case "package1" :
       case "package2" :
       case "component" :
       case "artifact" :
       case "usecase" :
       case "actor" :
       case "superstate" :
       case "action" :
       case "subactivity" :
       case "timesignal" :
       case "acceptsignal" :
       case "sendsignal" :
       case "connector" :
       case "transformation" :
       case "part" :


(document.getElementById(selectedObj
.objecttype + "_x")).value =
selectedObj["x"];

(document.getElementById(selectedObj
.objecttype + "_y")).value =
selectedObj["y"];

(document.getElementById(selectedObj
.objecttype + "_width")).value =
selectedObj["width"];

(document.getElementById(selectedObj
.objecttype + "_height")).value =
selectedObj["height"];

//(document.getElementById(selectedO
bj.objecttype + "_rotate")).value =
selectedObj["rotate"];
          break;
       case "expansionregion" :

(document.getElementById("expansionr
egion_x")).value = selectedObj["x"];

(document.getElementById("expansionr
egion_y")).value = selectedObj["y"];

(document.getElementById("expansionr
egion_width")).value =
selectedObj["width"];

(document.getElementById("expansionr
egion_height")).value =
selectedObj["height"];

(document.getElementById("expansionr
egion_listboxpin1x")).value =
selectedObj["listboxpin1x"];

(document.getElementById("expansionr
egion_listboxpin1y")).value =
selectedObj["listboxpin1y"];

(document.getElementById("expansionr
egion_listboxpin2x")).value =
selectedObj["listboxpin2x"];

(document.getElementById("expansionr
egion_listboxpin2y")).value =
selectedObj["listboxpin2y"];
          break;
       case "line" :

(document.getElementById("line_x1")).v
alue = selectedObj["x1"];

(document.getElementById("line_y1")).
value = selectedObj["y1"];

(document.getElementById("line_x2")).v
alue = selectedObj["x2"];

(document.getElementById("line_y2")).
value = selectedObj["y2"];
          break;
       case "bezier" :

(document.getElementById("bezier_x1"
)).value = selectedObj["x1"];

(document.getElementById("bezier_y1"
)).value = selectedObj["y1"];

(document.getElementById("bezier_x2"
)).value = selectedObj["x2"];

(document.getElementById("bezier_y2"
)).value = selectedObj["y2"];

(document.getElementById("bezier_ctrl
1x")).value = selectedObj["ctrl1x"];

(document.getElementById("bezier_ctrl
1y")).value = selectedObj["ctrl1y"];

(document.getElementById("bezier_ctrl
2x")).value = selectedObj["ctrl2x"];

(document.getElementById("bezier_ctrl
2y")).value = selectedObj["ctrl2y"];
          break;
       case "path" :
          break;
   }
}

function styleTab() {
   hideAllPropertiesBoxTable();

(document.getElementById("styleTab2")
).setAttribute('class', 'active');
   if(selectedObj)

(document.getElementById(selectedObj
.objecttype +
          "StyleTable")).setAttribute('style',
'display: block;');
}

function propertiesTab() {
   hideAllPropertiesBoxTable();

(document.getElementById("properties
Tab2")).setAttribute('class', 'active');

if(OBJ_PROPERTIES[selectedObj.objectt
ype].length > 0)

(document.getElementById(selectedObj
.objecttype +

"PropertiesTable")).setAttribute('style',
'display: block;');
}

function attributesTab() {
   hideAllPropertiesBoxTable();

(document.getElementById("attributesT
ab2")).setAttribute('class', 'active');
   if(selectedObj.objecttype == "class" ||
selectedObj.objecttype == "instance")

(document.getElementById("attributesT
able")).setAttribute('style', 'display:
block;');
}
```

```
function operationsTab() {
   hideAllPropertiesBoxTable();

(document.getElementById("operations
Tab2")).setAttribute('class', 'active');
   if(selectedObj.objecttype == "class")

(document.getElementById("operations
Table")).setAttribute('style', 'display:
block;');
}

function templatesTab() {
   hideAllPropertiesBoxTable();

(document.getElementById("templatesT
ab2")).setAttribute('class', 'active');
   if(selectedObj.objecttype == "class")

(document.getElementById("templatesT
able")).setAttribute('style', 'display:
block;');
}

function artifactsTab() {
   hideAllPropertiesBoxTable();

(document.getElementById("artifactsTa
b2")).setAttribute('class', 'active');
   if(selectedObj.objecttype == "node")

(document.getElementById("containeda
rtifactsTable")).setAttribute('style',
'display: block;');
}

function taggedvaluesTab() {
   hideAllPropertiesBoxTable();

(document.getElementById("taggedvalu
esTab2")).setAttribute('class', 'active');
   if(selectedObj.objecttype == "node")

(document.getElementById("taggedvalu
esTable")).setAttribute('style', 'display:
block;');
}

function internalactivitiesTab() {
   hideAllPropertiesBoxTable();

(document.getElementById("internalact
ivitiesTab2")).setAttribute('class',
'active');
   if(selectedObj.objecttype ==
"statebox")

(document.getElementById("internalact
ivitiesTable")).setAttribute('style',
'display: block;');
}

function deleteObj() {
   if(selectedObj == null) return;

   var index =
diagramObjects.indexOf(selectedObj);
   diagramObjects.splice(index, 1);
   noOfDiagramObjects =
diagramObjects.length;
   selectNone();

   generateOp(new
deleteOperation(index));
   generateOp(new noneOperation());
}

function zOrderUp() {
   if(selectedObj == null) return;

   var index =
diagramObjects.indexOf(selectedObj);

   if(index == diagramObjects.length - 1)
return;

   tempObject =
diagramObjects.splice(index, 1);
   diagramObjects.splice(index + 1, 0,
tempObject[0]);

   select(index + 1);
   tempObject = {};

   generateOp(new
moveOperation(index, index + 1));
   generateOp(new noneOperation());
}

function zOrderDown() {
   if(selectedObj == null) return;

   var index =
diagramObjects.indexOf(selectedObj);

   if(index == 0) return;

   tempObject =
diagramObjects.splice(index, 1);
   diagramObjects.splice(index - 1, 0,
tempObject[0]);

   select(index - 1);
   tempObject = {};

   generateOp(new
moveOperation(index, index - 1));
   generateOp(new noneOperation());
}

function change(attribute, value) {
   selectedObj[attribute] = value;
   invalidate();

   generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
      attribute, selectedObj[attribute],
null));
   generateOp(new noneOperation());
}

function changeFloat(attribute, value,
objecttype) {
   if(attribute == "rotate") {
      if(!isNumber(value)) {
         alert("You must enter a valid
number");

document.getElementById(objecttype +
"_rotate").value = selectedObj.rotate;
         return;
      }
   } else if(!isNumber(value) ||
parseFloat(value) <= 0) {
      alert("You must enter a number
greater than 0");
      switch(attribute) {
         case "width" :
         case "height" :
         case "linewidth" :
         case "listboxpinsize" :
         case "topheight" :
         case "fontsize" :
         case "classfontsize" :

document.getElementById(objecttype +
"_" + attribute).value =
selectedObj[attribute];
            break;
         default:
            break;
      }
      return;
   }

   selectedObj[attribute] =
parseFloat(value);
   invalidate();

   generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
      attribute, selectedObj[attribute],
null));
   generateOp(new noneOperation());
}

function changeBoolean(attribute,
value) {
   selectedObj[attribute] = value;
   invalidate();

   generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
      attribute, selectedObj[attribute],
null));
   generateOp(new noneOperation());
}

function changeXY(attribute, value,
objecttype) {
   if(!isNumber(value)) {
      alert("You must enter a number");

      switch(attribute) {
```

```
        case "x" :
        case "y" :
        case "x1" :
        case "y1" :
        case "x2" :
        case "y2" :
        case "ctrl1x" :
        case "ctrl1y" :
        case "ctrl2x" :
        case "ctrl2y" :
        case "listboxpin1x" :
        case "listboxpin1y" :
        case "listboxpin2x" :
        case "listboxpin2y" :

document.getElementById(objecttype +
"_" + attribute).value =
selectedObj[attribute];
          break;
        default:
          break;
    }

    return;
  }

  selectedObj[attribute] =
parseFloat(value);
  invalidate();

  switch(attribute) {
    case "x" :
    case "y" :
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "x", selectedObj.x, null));
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "y", selectedObj.y, null));
      break;
    case "x1" :
    case "y1" :
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "x1", selectedObj.x1, null));
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "y1", selectedObj.y1, null));
      break;
    case "x2" :
    case "y2" :
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "x2", selectedObj.x2, null));
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "y2", selectedObj.y2, null));
      break;
    case "ctrl1x" :
    case "ctrl1y" :
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "ctrl1x", selectedObj.ctrl1x,
null));
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "ctrl1y", selectedObj.ctrl1y,
null));
      break;
    case "ctrl2x" :
    case "ctrl2y" :
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "ctrl2x", selectedObj.ctrl2x,
null));
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "ctrl2y", selectedObj.ctrl2y,
null));
      break;
    case "listboxpin1x" :
    case "listboxpin1y" :
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "listboxpin1x",
selectedObj.listboxpin1x, null));
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "listboxpin1y",
selectedObj.listboxpin1y, null));
      break;
    case "listboxpin2x" :
    case "listboxpin2y" :
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "listboxpin2x",
selectedObj.listboxpin2x, null));
      generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
          "listboxpin2y",
selectedObj.listboxpin2y, null));
      break;
    default:
      break;
  }

  generateOp(new noneOperation());
}

function changeColor(object, style) {
  var rgbTemp =
document.getElementById(object + "_"
+ style).value;
  var r = parseInt(rgbTemp.substr(0, 2),
16);
  var g = parseInt(rgbTemp.substr(2, 2),
16);
  var b = parseInt(rgbTemp.substr(4, 2),
16);
  var a =
document.getElementById(object + "_"
+ style + "_A").value;
  /*var r =
document.getElementById(object + "_"
+ style + "_R").value;
  var g =
document.getElementById(object + "_"
+ style + "_G").value;
  var b =
document.getElementById(object + "_"
+ style + "_B").value;
  var a =
document.getElementById(object + "_"
+ style + "_A").value;*/

  var errMsg = "";
  var hasError = false;
  if(!isNumber(r) || r < 0 || r > 255) {
    //errMsg += "R value must be a
number between 0 and 255.\n";
    hasError = true;
  }
  if(!isNumber(g) || g < 0 || g > 255) {
    //errMsg += "G value must be a
number between 0 and 255.\n";
    hasError = true;
  }
  if(!isNumber(b) || b < 0 || b > 255) {
    //errMsg += "B value must be a
number between 0 and 255.\n";
    hasError = true;
  }
  if(!isNumber(a) || a < 0 || a > 1.0) {
    errMsg += "alpha value must be a
number between 0 and 1.\n";
    hasError = true;
  }
  if(hasError) {
    errMsg += "There is an error in the
rgba value.";
    alert(errMsg);
    var rgba =
selectedObj[style].split(",");

    var rDefault =
parseInt(rgba[0]).toString(16);
    var gDefault =
parseInt(rgba[1]).toString(16);
    var bDefault =
parseInt(rgba[2]).toString(16);
    rDefault = (rDefault.length < 2) ? '0'
+ rDefault : rDefault;
    gDefault = (gDefault.length < 2) ? '0'
+ gDefault : gDefault;
    bDefault = (bDefault.length < 2) ?
'0' + bDefault : bDefault;
    (document.getElementById(object
+ "_" + style)).color.fromString(rDefault
+ gDefault + bDefault);
    document.getElementById(object +
"_" + style + "_A").value = rgba[3];

    return;
```

```
   }

   selectedObj[style] = parseInt(r) + "," +
parseInt(g) + "," + parseInt(b) + "," + a;
   invalidate();

   generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj),
      style, selectedObj[style], null));
   generateOp(new noneOperation());
}

function isNumber(n) {
  return !isNaN(parseFloat(n)) &&
isFinite(n);
}
```

---

properties box 2.js

```
function itemSelect(type) {
   if(!(selectedObj.objecttype == "class"
|| selectedObj.objecttype == "instance"
||
       selectedObj.objecttype == "node"
|| selectedObj.objecttype ==
"statebox"))
      return;

   selectedIndex =
itemlist[type].selectedIndex;

   if(isNaN(selectedIndex) ||
selectedIndex < 0)
      return;

   switch(type) {
      case "attribute" :
         if(selectedIndex >=
selectedObj.attributes.length) return;


(document.getElementById("attributeEd
it")).value =

selectedObj.attributes[selectedIndex].v
alue;

(document.getElementById("attributeSt
atic")).checked =

selectedObj.attributes[selectedIndex].is
static;

(document.getElementById("attributeSt
atic")).disabled = false;
         break;
      case "operation" :
         if(selectedIndex >=
selectedObj.operations.length) return;


(document.getElementById("operationE
dit")).value =

selectedObj.operations[selectedIndex].v
alue;

(document.getElementById("operationS
tatic")).checked =

selectedObj.operations[selectedIndex].i
sstatic;

(document.getElementById("operationS
tatic")).disabled = false;

(document.getElementById("operationA
bstract")).checked =

selectedObj.operations[selectedIndex].i
sabstract;

(document.getElementById("operationA
bstract")).disabled = false;
         break;
      case "template" :
         if(selectedIndex >=
selectedObj.templates.length) return;


(document.getElementById("templateE
dit")).value =

selectedObj.templates[selectedIndex].v
alue;
         break;
      case "containedartifact" :
         if(selectedIndex >=
selectedObj.containedartifacts.length)
return;


(document.getElementById("containeda
rtifactEdit")).value =

selectedObj.containedartifacts[selectedI
ndex];
         break;
      case "taggedvalue" :
         if(selectedIndex >=
selectedObj.taggedvalues.length)
return;


(document.getElementById("taggedvalu
eEdit")).value =

selectedObj.taggedvalues[selectedIndex
];
         break;
      case "internalactivity" :
         if(selectedIndex >=
selectedObj.internalactivitys.length)
return;


(document.getElementById("internalact
ivityEdit")).value =

selectedObj.internalactivitys[selectedIn
dex];
         break;
      default :return;
   }

   (document.getElementById(type +
"Delete")).disabled = false;
   (document.getElementById(type +
"Up")).disabled = false;
   (document.getElementById(type +
"Down")).disabled = false;
   (document.getElementById(type +
"Edit")).disabled = false;
   (document.getElementById(type +
"Edit")).focus();
}

function itemUnselect(type) {
   selectedIndex = -1;
   itemlist[type].blur();
   (document.getElementById(type +
"Delete")).disabled = true;
   (document.getElementById(type +
"Up")).disabled = true;
   (document.getElementById(type +
"Down")).disabled = true;
   (document.getElementById(type +
"Edit")).disabled = true;
   (document.getElementById(type +
"Edit")).value = "";

   if(type == "attribute") {

(document.getElementById("attributeSt
atic")).disabled = true;

(document.getElementById("attributeSt
atic")).checked = false;
   } else if(type == "operation") {

(document.getElementById("operationS
tatic")).disabled = true;

(document.getElementById("operationS
tatic")).checked = false;

(document.getElementById("operationA
bstract")).disabled = true;

(document.getElementById("operationA
bstract")).checked = false;
   }
}

function itemNew(type) {
   var newItem = null;
   switch(type) {
      case "attribute" :
         newItem = {"value" : "", "static" :
false};
         selectedIndex =
selectedObj.attributes.push(newItem) -
1;
         break;
```

```
      case "operation" :
         newItem = {"value" : "", "static" :
false, "abstract" : false};
         selectedIndex =
selectedObj.operations.push(newItem) -
1;
         break;
      case "template" :
         newItem = {"value" : ""};
         selectedIndex =
selectedObj.templates.push(newItem) -
1;
         break;
      case "containedartifact" :
         newItem = "";
         selectedIndex =
selectedObj.containedartifacts.push(ne
wItem) - 1;
         break;
      case "taggedvalue" :
         newItem = "";
         selectedIndex =
selectedObj.taggedvalues.push(newIte
m) - 1;
         break;
      case "internalactivity" :
         newItem = "";
         selectedIndex =
selectedObj.internalactivitys.push(newIt
em) - 1;
         break;
      default:
         return;
   }

   invalidate();

   itemlist[type].options[selectedIndex]
= new Option("", "", false, true);
   itemSelect(type);
   (document.getElementById(type +
"Edit")).focus();

   // Generate and send operations
   var attrOp = new
insertOperation(newItem,
selectedIndex);
   generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), type, null, attrOp));
   generateOp(new noneOperation());
}

function itemEdit(type, attr, value) {
   if(selectedIndex == -1) return;

   var attrOp = null;

   switch(type) {
      case "attribute" :
         if(attr == "value") {

selectedObj.attributes[selectedIndex].v
alue = value;
            attrOp = new
editOperation(selectedIndex,
"attributeValue", value, null);
         } else if(attr == "static") {

selectedObj.attributes[selectedIndex].is
static = value;
            attrOp = new
editOperation(selectedIndex,
"attributeStatic", value, null);
         } else return;

         break;
      case "operation" :
         if(attr == "value") {

selectedObj.operations[selectedIndex].v
alue = value;
            attrOp = new
editOperation(selectedIndex,
"attributeValue", value, null);
         } else if(attr == "static") {

selectedObj.operations[selectedIndex].i
sstatic = value;
            attrOp = new
editOperation(selectedIndex,
"attributeStatic", value, null);
         } else if(attr == "abstract") {

selectedObj.operations[selectedIndex].i
sabstract = value;
            attrOp = new
editOperation(selectedIndex,
"attributeAbstract", value, null);
         } else return;
         break;
      case "template" :
         if(attr == "value") {

selectedObj.templates[selectedIndex].v
alue = value;
            attrOp = new
editOperation(selectedIndex,
"attributeValue", value, null);
         } else return;
         break;
      case "containedartifact" :
         if(attr == "value") {

selectedObj.containedartifacts[selectedI
ndex] = value;
            attrOp = new
editOperation(selectedIndex,
"attributeValue", value, null);
         } else return;
         break;
      case "taggedvalue" :
         if(attr == "value") {

selectedObj.taggedvalues[selectedIndex
] = value;
            attrOp = new
editOperation(selectedIndex,
"attributeValue", value, null);
         } else return;
         break;
      case "internalactivity" :
         if(attr == "value") {

selectedObj.internalactivitys[selectedIn
dex] = value;
            attrOp = new
editOperation(selectedIndex,
"attributeValue", value, null);
         } else return;
         break;
      default:
         return;
   }

   invalidate();

   // Change text of corresponding
select option
   if(attr == "value") {
      var selOption =
itemlist[type].options[selectedIndex];
      selOption.text = value;
      selOption.value = "";
   }

   // Generate and send operations
   generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), type, null, attrOp));
   generateOp(new noneOperation());
}

function itemDelete(type) {
   if(selectedIndex == -1) return;

   switch(type) {
      case "attribute" :

selectedObj.attributes.splice(selectedIn
dex, 1);
         break;
      case "operation" :

selectedObj.operations.splice(selectedIn
dex, 1);
         break;
      case "template" :

selectedObj.templates.splice(selectedIn
dex, 1);
         break;
      case "containedartifact" :

selectedObj.containedartifacts.splice(sel
ectedIndex, 1);
         break;
      case "taggedvalue" :

selectedObj.taggedvalues.splice(selecte
dIndex, 1);
         break;
      case "internalactivity" :

selectedObj.internalactivitys.splice(selec
tedIndex, 1);
```

```
        break;
      default:
         return;
   }

   invalidate();

   itemlist[type].remove(selectedIndex);
   var delIndex = selectedIndex;
   itemUnselect(type);

   // Generate and send operations
   var attrOp = new
deleteOperation(delIndex);
   generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), type, null, attrOp));
   generateOp(new noneOperation());
}

function itemUp(type) {
   if(selectedIndex <= 0)
      return;

   switch(type) {
      case "attribute" :
         var temp =
selectedObj.attributes[selectedIndex -
1];

selectedObj.attributes[selectedIndex -
1] =
selectedObj.attributes[selectedIndex];

selectedObj.attributes[selectedIndex] =
temp;
         break;
      case "operation" :
         var temp =
selectedObj.operations[selectedIndex -
1];

selectedObj.operations[selectedIndex -
1] =
selectedObj.operations[selectedIndex];

selectedObj.operations[selectedIndex] =
temp;
         break;
      case "template" :
         var temp =
selectedObj.templates[selectedIndex -
1];

selectedObj.templates[selectedIndex -
1] =
selectedObj.templates[selectedIndex];

selectedObj.templates[selectedIndex] =
temp;
         break;
      case "containedartifact" :
         var temp =
selectedObj.containedartifacts[selectedI
ndex - 1];

selectedObj.containedartifacts[selectedI
ndex - 1] =
selectedObj.containedartifacts[selectedI
ndex];

selectedObj.containedartifacts[selectedI
ndex] = temp;
         break;
      case "taggedvalue" :
         var temp =
selectedObj.taggedvalues[selectedIndex
- 1];

selectedObj.taggedvalues[selectedIndex
- 1] =
selectedObj.taggedvalues[selectedIndex
];

selectedObj.taggedvalues[selectedIndex
] = temp;
         break;
      case "internalactivity" :
         var temp =
selectedObj.internalactivitys[selectedIn
dex - 1];

selectedObj.internalactivitys[selectedIn
dex - 1] =
selectedObj.internalactivitys[selectedIn
dex];

selectedObj.internalactivitys[selectedIn
dex] = temp;
         break;
      default:
         return;
   }

   invalidate();

   // Switch the positions of the affected
items in the select options
   temp =
itemlist[type].options[selectedIndex -
1].text;
   itemlist[type].options[selectedIndex -
1].text =
itemlist[type].options[selectedIndex].te
xt;

itemlist[type].options[selectedIndex].te
xt = temp;
   selectedIndex--;

itemlist[type].options[selectedIndex].sel
ected = true;

   // Generate and send operations
   var attrOp = new
moveOperation(selectedIndex + 1,
selectedIndex);
   generateOp(new
editOperation(diagramObjects.indexOf(
selectedObj), type, null, attrOp));
   generateOp(new noneOperation());
}

function itemDown(type) {
   switch(type) {
      case "attribute" :
         if(selectedIndex >=
selectedObj.attributes.length - 1)
return;

         var temp =
selectedObj.attributes[selectedIndex +
1];

selectedObj.attributes[selectedIndex +
1] =
selectedObj.attributes[selectedIndex];

selectedObj.attributes[selectedIndex] =
temp;
         break;
      case "operation" :
         if(selectedIndex >=
selectedObj.operations.length - 1)
return;

         var temp =
selectedObj.operations[selectedIndex +
1];

selectedObj.operations[selectedIndex +
1] =
selectedObj.operations[selectedIndex];

selectedObj.operations[selectedIndex] =
temp;
         break;
      case "template" :
         if(selectedIndex >=
selectedObj.templates.length - 1)
return;

         var temp =
selectedObj.templates[selectedIndex +
1];

selectedObj.templates[selectedIndex +
1] =
selectedObj.templates[selectedIndex];

selectedObj.templates[selectedIndex] =
temp;
         break;
      case "containedartifact" :
         if(selectedIndex >=
selectedObj.containedartifacts.length -
1) return;

         var temp =
selectedObj.containedartifacts[selectedI
ndex + 1];

selectedObj.containedartifacts[selectedI
ndex + 1] =
selectedObj.containedartifacts[selectedI
ndex];
```

```
selectedObj.containedartifacts[selectedIndex] = temp;
        break;
      case "taggedvalue" :
        if(selectedIndex >= selectedObj.taggedvalues.length - 1) return;

        var temp = selectedObj.taggedvalues[selectedIndex + 1];

selectedObj.taggedvalues[selectedIndex + 1] = selectedObj.taggedvalues[selectedIndex];

selectedObj.taggedvalues[selectedIndex] = temp;
        break;
      case "internalactivity" :
        if(selectedIndex >= selectedObj.internalactivitys.length - 1) return;

        var temp = selectedObj.internalactivitys[selectedIndex + 1];

selectedObj.internalactivitys[selectedIndex + 1] = selectedObj.internalactivitys[selectedIndex];

selectedObj.internalactivitys[selectedIndex] = temp;
        break;
      default:
        return;
  }

  invalidate();

  // Switch the positions of the affected items in the select options
  temp = itemlist[type].options[selectedIndex + 1].text;
  itemlist[type].options[selectedIndex + 1].text = itemlist[type].options[selectedIndex].text;

itemlist[type].options[selectedIndex].text = temp;
  selectedIndex++;

itemlist[type].options[selectedIndex].selected = true;

  // Generate and send operations
  var attrOp = new moveOperation(selectedIndex - 1, selectedIndex);
  generateOp(new editOperation(diagramObjects.indexOf(selectedObj), type, null, attrOp));
  generateOp(new noneOperation());
}

function addPoint() {
  if(!(selectedObj.objecttype == "polygon" || selectedObj.objecttype == "polyline"))
    return;

  if(pointOperation != "none") return;


document.getElementById(selectedObj.objecttype + "_addpoint_message").setAttribute("style", "display: block");

document.getElementById(selectedObj.objecttype + "_addpoint").disabled = true;

document.getElementById(selectedObj.objecttype + "_removepoint").disabled = true;
  expectingInputClick = true;
  pointOperation = "add";
}

function removePoint() {
  if(!(selectedObj.objecttype == "polygon" || selectedObj.objecttype == "polyline"))
    return;

  if(pointOperation != "none") return;


document.getElementById(selectedObj.objecttype + "_removepoint_message").setAttribute("style", "display: block");

document.getElementById(selectedObj.objecttype + "_addpoint").disabled = true;

document.getElementById(selectedObj.objecttype + "_removepoint").disabled = true;
  expectingInputClick = true;
  pointOperation = "remove";
}
```

---

tools.js

```
var diagramTitle = '';

var tools = {};
var tool;
var activeButton;

var ghostCanvas;
var ghostContext;

var tempCanvas;
var tempContext;

var hasMovedCursor = false;
var isDrag = false;
var isResizeDrag = false;
var expectResize = -1;
var offsetx, offsety;
var offsetx2, offsety2;
var offsetxp = [];
var offsetyp = [];
var selectedObj;

var expectingInputClick = false;
var pointOperation = "none";

var textAreaPopUp;
var textValueInput;

function tools_init() {
  ghostCanvas = document.createElement('canvas');
  if (!ghostCanvas) {
    alert('Error: I cannot create a new canvas element!');
    return;
  }
  ghostCanvas.height = HEIGHT;
  ghostCanvas.width = WIDTH;
  ghostContext = ghostCanvas.getContext('2d');

  // Add the temporary canvas.
  var container = canvas.parentNode;
  tempCanvas = document.createElement('canvas');
  if (!tempCanvas) {
    alert('Error: I cannot create a new canvas element!');
    return;
  }
  tempCanvas.id     = 'drawingAreaTemp';
  tempCanvas.width  = WIDTH;
  tempCanvas.height = HEIGHT;

//container.appendChild(tempCanvas);
  container.insertBefore(tempCanvas, document.getElementById("textAreaPopUp"));

  tempContext = tempCanvas.getContext('2d');

  // Attach the mousedown, mousemove and mouseup event listeners.

tempCanvas.addEventListener('mousedown', ev_canvas, false);

tempCanvas.addEventListener('mousemove', ev_canvas, false);
```

```
tempCanvas.addEventListener('mouseup',   ev_canvas, false);

   // Fixes problem where double clicking causes text to get selected
   tempCanvas.onselectstart = function () {return false;}

   // Display title

document.getElementById('diagramTitle').appendChild(document.createTextNode('Title: ' + diagramTitle + ''));

   // Textarea pop up for text tool
   textAreaPopUp = document.getElementById("textAreaPopUp");
   textValueInput = document.getElementById("textValueInput");

   activateDefaultTool();
   default_init();
   propertiesBoxTab();
}

// The general-purpose event handler. This function just determines the mouse
// position relative to the canvas element.
function ev_canvas (ev) {
   if (ev.layerX || ev.layerX == 0) { // Firefox
      ev._x = ev.layerX;
      ev._y = ev.layerY;
   } else if (ev.offsetX || ev.offsetX == 0) { // Opera
      ev._x = ev.offsetX;
      ev._y = ev.offsetY;
   }
   // Call the event handler of the tool.
   var func = tool[ev.type];
   if (func) func(ev);
}

function selectNone() {
   expectingInputClick = false;
   pointOperation = "none";
   selectedObj = null;
   updatePropertiesBox();
   invalidate();
}

function select(index) {
   selectedObj = diagramObjects[index];
   updatePropertiesBox();
   invalidate();
}

function activateDefaultTool() {
   if (tools[DEFAULT_TOOL]) {
      tool = new tools[DEFAULT_TOOL]();
      if(activeButton) activeButton.removeAttribute("class");
      activeButton = document.getElementById(DEFAULT_TOOL);
      activeButton.setAttribute("class", "active");
   }
}

function activateTool(toolName) {

activeButton.removeAttribute("class");
   activeButton = document.getElementById(toolName);
   activeButton.setAttribute("class", "active");
   tool = new tools[toolName]();
}

tools.select = function() {
   tool = this;
   this.started = false;

   resetTextAreaPopUp();

   this.mousedown = function (ev) {
      tool.started = true;

      if(expectingInputClick) return;

      tool.x0 = ev._x;
      tool.y0 = ev._y;

      if(expectResize !== -1) {
         isResizeDrag = true;
         return;
      }

      ghostContext.clear();
      for(var i = noOfDiagramObjects-1; i >= 0; i--) {

ghostContext.draw(diagramObjects[i]);

         var imageData = ghostContext.getImageData(tool.x0, tool.y0, 1, 1);

         // if the mouse pixel exists, select and break
         if (imageData.data[3] > 0) {
            selectedObj = diagramObjects[i];

            if(selectedObj.objecttype == "line") {
               offsetx = tool.x0 - selectedObj.x1;
               offsety = tool.y0 - selectedObj.y1;
               offsetx2 = tool.x0 - selectedObj.x2;
               offsety2 = tool.y0 - selectedObj.y2;
            } else if(selectedObj.objecttype == "path" || selectedObj.objecttype == "polygon" || selectedObj.objecttype == "polyline") {
               for(var c = 0; c < selectedObj.points.length; c++) {
                  offsetxp[c] = tool.x0 - selectedObj.points[c].x;
                  offsetyp[c] = tool.y0 - selectedObj.points[c].y;
               }
            } else if(selectedObj.objecttype == "bezier") {
               offsetxp[0] = tool.x0 - selectedObj.x1;
               offsetyp[0] = tool.y0 - selectedObj.y1;
               offsetxp[1] = tool.x0 - selectedObj.x2;
               offsetyp[1] = tool.y0 - selectedObj.y2;
               offsetxp[2] = tool.x0 - selectedObj.ctrl1x;
               offsetyp[2] = tool.y0 - selectedObj.ctrl1y;
               offsetxp[3] = tool.x0 - selectedObj.ctrl2x;
               offsetyp[3] = tool.y0 - selectedObj.ctrl2y;
            } else {
               offsetx = tool.x0 - selectedObj.x;
               offsety = tool.y0 - selectedObj.y;
            }

            isDrag = true;

            updatePropertiesBox();
            invalidate();
            ghostContext.clear();

tempCanvas.style.cursor='move';
            return;
         }
      }

      ghostContext.clear();
      selectNone();
   };

   this.mousemove = function (ev) {
      tool.x0 = ev._x;
      tool.y0 = ev._y;

      if(expectingInputClick) return;

      if(isDrag) {
         if(selectedObj.objecttype == "line") {
            selectedObj.x1 = tool.x0 - offsetx;
            selectedObj.y1 = tool.y0 - offsety;
            selectedObj.x2 = tool.x0 - offsetx2;
```

```
            selectedObj.y2 = tool.y0 -
offsety2;
        } else if(selectedObj.objecttype
== "path" ||
                  selectedObj.objecttype ==
"polygon" ||
                  selectedObj.objecttype ==
"polyline") {
            for(var i = 0; i <
selectedObj.points.length; i++) {
               selectedObj.points[i].x =
tool.x0 - offsetxp[i];
               selectedObj.points[i].y =
tool.y0 - offsetyp[i];
            }
        } else if(selectedObj.objecttype
== "bezier") {
            selectedObj.x1 = tool.x0 -
offsetxp[0];
            selectedObj.y1 = tool.y0 -
offsetyp[0];
            selectedObj.x2 = tool.x0 -
offsetxp[1];
            selectedObj.y2 = tool.y0 -
offsetyp[1];
            selectedObj.ctrl1x = tool.x0 -
offsetxp[2];
            selectedObj.ctrl1y = tool.y0 -
offsetyp[2];
            selectedObj.ctrl2x = tool.x0 -
offsetxp[3];
            selectedObj.ctrl2y = tool.y0 -
offsetyp[3];
        } else {
            selectedObj.x = tool.x0 -
offsetx;
            selectedObj.y = tool.y0 -
offsety;
        }

        //updatePropertiesBox();
        hasMovedCursor = true;
        invalidate();
        tempCanvas.style.cursor='move';
     } else if(isResizeDrag) {
        resizeSelectedObj(tool.x0,
tool.y0);
        hasMovedCursor = true;
        //updatePropertiesBox();
        invalidate();
     }

     if(selectedObj !== null &&
!isResizeDrag && !isDrag)
changeHandleCursor();
   };

   this.mouseup = function (ev) {
     if(tool.started == false) return;

     if(expectingInputClick) {
        if(pointOperation == "add") {
           generateAddPointOp(ev._x,
ev._y);
        } else if(pointOperation ==
"remove") {
           generateRemovePointOp(ev._x, ev._y);
        }
        expectingInputClick = false;
     }

     updatePropertiesBox2();

     // Generate the operation that was
just done
     if(isDrag && hasMovedCursor)
        generateDragOp();
     else if(isResizeDrag &&
hasMovedCursor)
        generateResizeOp();

     hasMovedCursor = false;
     isDrag = false;
     isResizeDrag = false;
     expectResize = -1;
     tool.started = false;
   };
};

tools.rectangle = function() {
   tool = this;
   this.started = false;

   selectNone();
   resetTextAreaPopUp();

   var x, y, w, h;

   this.mousedown = function (ev) {
      tool.started = true;
      tool.x0 = ev._x;
      tool.y0 = ev._y;

      tempContext.save();

tempContext.setFillAndStroke(DEFAULT
_FILLCOLOR, DEFAULT_STROKESTYLE);

tempContext.setLineSettings(DEFAULT_
LINEWIDTH, DEFAULT_LINECAP,
DEFAULT_LINEJOIN);
   };

   this.mousemove = function (ev) {
      if (!tool.started) return;

      x = Math.min(ev._x,  tool.x0);
      y = Math.min(ev._y,  tool.y0);
      w = Math.abs(ev._x - tool.x0);
      h = Math.abs(ev._y - tool.y0);

      tempContext.clear();

      if (!w || !h) return;

      tempContext.drawRectangle(x, y,
w, h, 0);
   };

   this.mouseup = function (ev) {
      if (tool.started) {
         tool.mousemove(ev);
         tool.started = false;
         tempContext.restore();
         addNewRectangle(x, y, w, h);
         diagramUpdate();
         activateDefaultTool();
      }
   };
};

tools.ellipse = function() {
   tool = this;
   this.started = false;

   selectNone();
   resetTextAreaPopUp();

   var x, y, w, h;

   this.mousedown = function (ev) {
      tool.started = true;
      tool.x0 = ev._x;
      tool.y0 = ev._y;

      tempContext.save();

tempContext.setFillAndStroke(DEFAULT
_FILLCOLOR, DEFAULT_STROKESTYLE);

tempContext.setLineSettings(DEFAULT_
LINEWIDTH, DEFAULT_LINECAP,
DEFAULT_LINEJOIN);
   };

   this.mousemove = function (ev) {
      if (!tool.started) return;

      x = Math.min(ev._x,  tool.x0);
      y = Math.min(ev._y,  tool.y0);
      w = Math.abs(ev._x - tool.x0);
      h = Math.abs(ev._y - tool.y0);

      tempContext.clear();

      if (!w || !h) return;

      tempContext.drawEllipse(x, y, w, h,
0);
   };

   this.mouseup = function (ev) {
      if (tool.started) {
         tool.mousemove(ev);
         tool.started = false;
         tempContext.restore();
         addNewEllipse(x, y, w, h);
         diagramUpdate();
         activateDefaultTool();
      }
   };
};

tools.line = function() {
   tool = this;
   this.started = false;
```

```
   selectNone();
   resetTextAreaPopUp();

   this.mousedown = function (ev) {
      tool.started = true;
      tool.x0 = ev._x;
      tool.y0 = ev._y;

      tempContext.save();

tempContext.setFillAndStroke(DEFAULT
_STROKESTYLE,
DEFAULT_STROKESTYLE);

tempContext.setLineSettings(DEFAULT_
LINEWIDTH, DEFAULT_LINECAP,
DEFAULT_LINEJOIN);
   };

   this.mousemove = function (ev) {
      if (!tool.started) return;

      tempContext.clear();

      tempContext.drawLine(tool.x0,
tool.y0, ev._x, ev._y,
DEFAULT_ARROWSTYLE1,
         DEFAULT_ARROWSTYLE2,
DEFAULT_LINESTYLE);
   };

   this.mouseup = function (ev) {
      if (tool.started) {
         tool.mousemove(ev);
         tool.started = false;
         tempContext.restore();
         addNewLine(tool.x0, tool.y0,
ev._x, ev._y);
         diagramUpdate();
         activateDefaultTool();
      }
   };
};

tools.path = function() {
   tool = this;
   this.started = false;
   var pathPoints = [];

   selectNone();
   resetTextAreaPopUp();

   this.mousedown = function (ev) {
      tempContext.save();

tempContext.setFillAndStroke(DEFAULT
_FILLCOLOR, DEFAULT_STROKESTYLE);

tempContext.setLineSettings(DEFAULT_
LINEWIDTH, DEFAULT_LINECAP,
DEFAULT_LINEJOIN);

      tempContext.beginPath();
      tempContext.moveTo(ev._x, ev._y);
      tool.started = true;
   };

   this.mousemove = function (ev) {
      if (tool.started) {
         tempContext.lineTo(ev._x, ev._y);
         tempContext.stroke();

         var temp = {"x" : ev._x, "y" :
ev._y};
         pathPoints.push(temp);
      }
   };

   this.mouseup = function (ev) {
      if (tool.started) {
         tool.mousemove(ev);
         tool.started = false;
         tempContext.restore();
         addNewPath(pathPoints);
         diagramUpdate();
         pathPoints = [];
         activateDefaultTool();
      }
   };
};

tools.text = function() {
   tool = this;
   this.started = false;

   selectNone();

   this.mousedown = function (ev) {
      tool.started = true;


//////////////////////////////////////////
//////////////////

      var mouseX = ev.pageX -
document.getElementById("content").o
ffsetLeft
         +
document.getElementById("canvasArea
").scrollLeft;
      var mouseY = ev.pageY -
document.getElementById("content").o
ffsetTop
         +
document.getElementById("canvasArea
").scrollTop;

      var textAreaPopUp =
document.getElementById("textAreaPo
pUp");

      if((mouseX +
textAreaPopUp.offsetWidth) >

document.getElementById("canvasArea
").offsetWidth) {

         mouseX -=
textAreaPopUp.offsetWidth;
      }

      if((mouseY +
textAreaPopUp.offsetHeight) >

document.getElementById("canvasArea
").offsetHeight) {

         mouseY -=
textAreaPopUp.offsetHeight;
      }

      textAreaPopUp.setAttribute("style",
"visibility:visible;top:" + mouseY
         + "px;left:" + mouseX + "px;");

      var textValueInput =
document.getElementById("textValueIn
put");

setTimeout(function(){textValueInput.fo
cus();}, 100);

//////////////////////////////////////////
//////////////////
   };

   this.mousemove = function (ev) {
      if (tool.started) {
      }
   };

   this.mouseup = function (ev) {
      /*if (tool.started) {
         tool.mousemove(ev);
         tool.started = false;
         addNewText(ev._x, ev._y);
         diagramUpdate();
         activateDefaultTool();
      }*/
   };
};

function saveText() {

addNewText(textAreaPopUp.offsetLeft,
textAreaPopUp.offsetTop,
      textAreaPopUp.offsetWidth,
textAreaPopUp.offsetHeight,
textValueInput.value);
   resetTextAreaPopUp();
   diagramUpdate();
   activateDefaultTool();
}

function resetTextAreaPopUp() {
   textAreaPopUp.setAttribute("style",
"visibility:hidden;");
   textValueInput.value = "";
}

tools.classnotation = function() {
   tool = this;
   this.started = false;

   selectNone();
   resetTextAreaPopUp();
```

```
  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewClass(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.note = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewNote(ev._x, ev._y, "");
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.bezier = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
    tool.x0 = ev._x;
    tool.y0 = ev._y;

    tempContext.save();

tempContext.setFillAndStroke(DEFAULT
_STROKESTYLE,
DEFAULT_STROKESTYLE);

tempContext.setLineSettings(DEFAULT_
LINEWIDTH, DEFAULT_LINECAP,
DEFAULT_LINEJOIN);
  };

  this.mousemove = function (ev) {
    if (!tool.started) return;

    tempContext.clear();
    tempContext.drawBezier(tool.x0,
tool.y0, ev._x, ev._y, (ev._x + 2 *
tool.x0) / 3, tool.y0,
      (tool.x0 + 2 * ev._x) / 3, ev._y,
DEFAULT_ARROWSTYLE1,
DEFAULT_ARROWSTYLE2,
DEFAULT_LINESTYLE);
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;
      tempContext.restore();
      addNewBezier(tool.x0, tool.y0,
ev._x, ev._y, (ev._x + 2 * tool.x0) / 3,
        tool.y0, (tool.x0 + 2 * ev._x) / 3,
ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.polygon = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewPolygon(ev._x, ev._y, "");
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.polyline = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
    tool.x0 = ev._x;
    tool.y0 = ev._y;

    tempContext.save();

tempContext.setFillAndStroke(DEFAULT
_STROKESTYLE,
DEFAULT_STROKESTYLE);

tempContext.setLineSettings(DEFAULT_
LINEWIDTH, DEFAULT_LINECAP,
DEFAULT_LINEJOIN);
  };

  this.mousemove = function (ev) {
    if (!tool.started) return;

    tempContext.clear();

    tempContext.drawPolyline([{"x" :
tool.x0, "y" : tool.y0}, {"x" : ev._x, "y" :
ev._y}], DEFAULT_ARROWSTYLE1,
      DEFAULT_ARROWSTYLE2,
DEFAULT_LINESTYLE);
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;
      tempContext.restore();
      addNewPolyline(tool.x0, tool.y0,
ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.instance = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewInstance(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.node = function() {
  tool = this;
```

```
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewNode(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.statebox = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewStatebox(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.expansionregion = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewExpansionRegion(ev._x,
ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.arrowhead = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewArrowhead(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.activationbar = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewActivationBar(ev._x,
ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.lifeline = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewLifeline(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.deletion = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewDeletion(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.initialps = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
```

```
        tool.mousemove(ev);
        tool.started = false;

        addNewInitialPS(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.finalstate = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewFinalState(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.shallowhistps = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewShallowHistPS(ev._x,
ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.deephistps = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewDeepHistPS(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.initialnode = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewInitialNode(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.finalnode = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewFinalNode(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.blackbar = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewBlackBar(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.diamond = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewDiamond(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.pin = function() {
    tool = this;
    this.started = false;

    selectNone();
```

```
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewPin(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.flowfinal = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewFlowFinal(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.port = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewPort(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.frame = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewFrame(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.package1 = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewPackage1(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.package2 = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewPackage2(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.component = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewComponent(ev._x,
ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.artifact = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewArtifact(ev._x, ev._y);
```

```
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.usecase = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewUseCase(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.actor = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewActor(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.superstate = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewSuperstate(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.action = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewAction(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.subactivity = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewSubactivity(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.timesignal = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewTimeSignal(ev._x, ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.acceptsignal = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
    };

    this.mousemove = function (ev) {
    };

    this.mouseup = function (ev) {
      if (tool.started) {
        tool.mousemove(ev);
        tool.started = false;

        addNewAcceptSignal(ev._x,
ev._y);
        diagramUpdate();
        activateDefaultTool();
      }
    };
};

tools.sendsignal = function() {
    tool = this;
    this.started = false;

    selectNone();
    resetTextAreaPopUp();

    this.mousedown = function (ev) {
      tool.started = true;
```

```
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewSendSignal(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.connector = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewConnector(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.transformation = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewTransformation(ev._x,
ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

tools.part = function() {
  tool = this;
  this.started = false;

  selectNone();
  resetTextAreaPopUp();

  this.mousedown = function (ev) {
    tool.started = true;
  };

  this.mousemove = function (ev) {
  };

  this.mouseup = function (ev) {
    if (tool.started) {
      tool.mousemove(ev);
      tool.started = false;

      addNewPart(ev._x, ev._y);
      diagramUpdate();
      activateDefaultTool();
    }
  };
};

// Draw the #drawingAreaTemp canvas
on top of #drawingArea and clear
// #drawingAreaTemp afterward.
Function is called each time the user
// completes a drawing operation.
function diagramUpdate () {
  tempContext.clear();
  invalidate();
}
```

---

tools 2.js

```
function propertiesBoxTab() {

document.getElementById("propertiesT
abButton").setAttribute("class",
"active");

document.getElementById("collaborato
rsTabButton").removeAttribute("class");

document.getElementById("recentChan
gesTabButton").removeAttribute("class"
);


document.getElementById("appliedOps
Table").setAttribute("style", "display:
none;");

document.getElementById("collaborato
rsAndCurrentUsers").setAttribute("style
", "display: none;");

document.getElementById("propertiesB
ox").setAttribute("style", "display:
block;");
  styleTab();
}

function collaboratorsTab() {

document.getElementById("collaborato
rsTabButton").setAttribute("class",
"active");

document.getElementById("propertiesT
abButton").removeAttribute("class");

document.getElementById("recentChan
gesTabButton").removeAttribute("class"
);


document.getElementById("propertiesB
ox").setAttribute("style", "display:
none;");

document.getElementById("appliedOps
Table").setAttribute("style", "display:
none;");

document.getElementById("collaborato
rsAndCurrentUsers").setAttribute("style
", "display: block;");
  //currentusersSubTab();
  collaboratorsSubTab();
}

function recentChangesTab() {

document.getElementById("recentChan
gesTabButton").setAttribute("class",
"active");

document.getElementById("propertiesT
abButton").removeAttribute("class");

document.getElementById("collaborato
rsTabButton").removeAttribute("class");


document.getElementById("propertiesB
ox").setAttribute("style", "display:
none;");

document.getElementById("collaborato
rsAndCurrentUsers").setAttribute("style
", "display: none;");

document.getElementById("appliedOps
Table").setAttribute("style", "display:
block;");
}

function default_init() {
  // arrow settings

(document.getElementById("defaultArr
owhead1")).value =
DEFAULT_ARROWSTYLE1;
```

```
(document.getElementById("defaultArrowhead2")).value = DEFAULT_ARROWSTYLE2;

(document.getElementById("defaultLinestyle")).value = DEFAULT_LINESTYLE;

   // font settings

(document.getElementById("selectedFontsize")).value = DEFAULT_FONTSIZE;

(document.getElementById("selectedFontfamily")).value = DEFAULT_FONTFAMILY;

   // fill settings
   var rgbaFill = DEFAULT_FILLCOLOR.split(",");
   var rFill = parseInt(rgbaFill[0]).toString(16);
   var gFill = parseInt(rgbaFill[1]).toString(16);
   var bFill = parseInt(rgbaFill[2]).toString(16);
   rFill = (rFill.length < 2) ? '0' + rFill : rFill;
   gFill = (gFill.length < 2) ? '0' + gFill : gFill;
   bFill = (bFill.length < 2) ? '0' + bFill : bFill;

(document.getElementById("defaultFill")).color.fromString(rFill + gFill + bFill);

(document.getElementById("defaultFillA")).value = rgbaFill[3];

   // stroke settings
   var rgbaStroke = DEFAULT_STROKESTYLE.split(",");
   var rStroke = parseInt(rgbaStroke[0]).toString(16);
   var gStroke = parseInt(rgbaStroke[1]).toString(16);
   var bStroke = parseInt(rgbaStroke[2]).toString(16);
   rStroke = (rStroke.length < 2) ? '0' + rStroke : rStroke;
   gStroke = (gStroke.length < 2) ? '0' + gStroke : gStroke;
   bStroke = (bStroke.length < 2) ? '0' + bStroke : bStroke;

(document.getElementById("defaultStroke")).color.fromString(rStroke + gStroke
      + bStroke);

(document.getElementById("defaultStrokeA")).value = rgbaStroke[3];

   // text color settings
   var rgbaText = DEFAULT_TEXTCOLOR.split(",");
   var rText = parseInt(rgbaText[0]).toString(16);
   var gText = parseInt(rgbaText[1]).toString(16);
   var bText = parseInt(rgbaText[2]).toString(16);
   rText = (rText.length < 2) ? '0' + rText : rText;
   gText = (gText.length < 2) ? '0' + gText : gText;
   bText = (bText.length < 2) ? '0' + bText : bText;

(document.getElementById("defaultText")).color.fromString(rText + gText + bText);

(document.getElementById("defaultTextA")).value = rgbaText[3];
}

function changeSetting(setting, newValue) {
   switch(setting) {
      case "arrowhead1" :
         DEFAULT_ARROWSTYLE1 = newValue;
         break;
      case "linestyle" :
         DEFAULT_LINESTYLE = newValue;
         break;
      case "arrowhead2" :
         DEFAULT_ARROWSTYLE2 = newValue;
         break;
      case "fontsize" :
         if(!isNumber(newValue) || parseFloat(newValue) <= 0) {
            alert("You must enter a number greater than 0");

document.getElementById("selectedFontsize").value = DEFAULT_FONTSIZE;
            return;
         }
         DEFAULT_FONTSIZE = parseFloat(newValue);
         break;
      case "fontfamily" :
         DEFAULT_FONTFAMILY = newValue;
         break;
      default:
         break;
   }
}

function changeColorSetting(setting) {
   var rgbTemp = document.getElementById("default" + setting).value;
   var r = parseInt(rgbTemp.substr(0, 2), 16);
   var g = parseInt(rgbTemp.substr(2, 2), 16);
   var b = parseInt(rgbTemp.substr(4, 2), 16);
   var a = document.getElementById("default" + setting + "A").value;

   var errMsg = "";
   var hasError = false;
   if(!isNumber(r) || r < 0 || r > 255) {
      //errMsg += "R value must be a number between 0 and 255.\n";
      hasError = true;
   }
   if(!isNumber(g) || g < 0 || g > 255) {
      //errMsg += "G value must be a number between 0 and 255.\n";
      hasError = true;
   }
   if(!isNumber(b) || b < 0 || b > 255) {
      //errMsg += "B value must be a number between 0 and 255.\n";
      hasError = true;
   }
   if(!isNumber(a) || a < 0 || a > 1.0) {
      errMsg += "alpha value must be a number between 0 and 1.\n";
      hasError = true;
   }
   if(hasError) {
      errMsg += "There is an error in the rgba value.";
      alert(errMsg);
      var rgbaDefault;
      switch(setting) {
         case "Fill" :
            rgbaDefault = DEFAULT_FILLCOLOR.split(",");
            break;
         case "Stroke" :
            rgbaDefault = DEFAULT_STROKESTYLE.split(",");
            break;
         case "Text" :
            rgbaDefault = DEFAULT_TEXTCOLOR.split(",");
            break;
         default :
            break;
      }

      var rDefault = parseInt(rgbaDefault[0]).toString(16);
      var gDefault = parseInt(rgbaDefault[1]).toString(16);
      var bDefault = parseInt(rgbaDefault[2]).toString(16);
      rDefault = (rDefault.length < 2) ? '0' + rDefault : rDefault;
      gDefault = (gDefault.length < 2) ? '0' + gDefault : gDefault;
      bDefault = (bDefault.length < 2) ? '0' + bDefault : bDefault;

(document.getElementById("default" + setting)).color.fromString(rDefault + gDefault + bDefault);
```

```
document.getElementById("default" +
setting + "A").value = rgbaDefault[3];
      return;
   }

   var rgba = r + "," + g + "," + b + "," + a;

   switch(setting) {
      case "Fill" :
         DEFAULT_FILLCOLOR = rgba;
         break;
      case "Stroke" :
         DEFAULT_STROKESTYLE = rgba;
         break;
      case "Text" :
         DEFAULT_TEXTCOLOR = rgba;
         break;
      default :
         break;
   }
}
```

AuthenticateFilter.java

```
package cordie;

import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class AuthenticateFilter
implements Filter {
   public void init(FilterConfig
filterConfig) {
   }

   public void doFilter(ServletRequest
req, ServletResponse res, FilterChain
chain)
         throws IOException,
ServletException {
      HttpServletRequest request =
(HttpServletRequest) req;
      HttpServletResponse response =
(HttpServletResponse) res;
      HttpSession session =
request.getSession(false);

      if (session == null ||
session.getAttribute("USERNAME") ==
null) {

response.sendRedirect("../denyAccess.js
p"); // No logged-in user found, so
redirect to login page.
      } else {
         chain.doFilter(req, res); //
Logged-in user found, so just continue
request.
      }
   }

   public void destroy() {
   }
}
```

CordieDiagramsMap.java

```
package cordie;

import cordie.diagram.CordieDiagram;
import java.util.HashMap;
import java.util.Collections;
import java.util.Map;
import java.util.Timer;
import java.util.TimerTask;
import java.util.ArrayList;

public class CordieDiagramsMap {
   private Map<String, CordieDiagram>
map;
   private Timer timer;

   private CordieDiagramsMap() {
      map =
Collections.synchronizedMap(new
HashMap<String, CordieDiagram>());

      //int delay = 5 * 1000;   // delay for
5 sec.
      //int period = 10 * 1000;  // repeat
every sec.
      //timer = new Timer();
      //timer.scheduleAtFixedRate(new
MonitorDiagrams(), delay, period);
   }

   private static CordieDiagramsMap
cdm = new CordieDiagramsMap();

   public static CordieDiagramsMap
getInstance() {
      return cdm;
   }

   /*private class MonitorDiagrams
extends TimerTask {
      @Override
      public void run() {
         ArrayList<String>
diagramsToRemove = new
ArrayList<String>();
         for(String diagramID :
map.keySet()) {
            //System.out.println("current
key" + diagramID);

if(map.get(diagramID).isInactive()) {

diagramsToRemove.add(diagramID);
            } else {
//System.out.println("diagram " +
diagramID + " is still active.");
            }
         }
         for(String diagramToRemove :
diagramsToRemove) {
            System.out.println("diagram "
+ diagramToRemove + " is no longer
active.");

map.remove(diagramToRemove);
         }
      }
   }*/

   public Map getMap() {
      return map;
   }
}
```

CordieEditor.java

```
package cordie;

import cordie.diagram.*;
import cordie.diagram.operation.*;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletConfig;
import java.util.Map;
import java.io.File;
import java.io.FileOutputStream;
import java.net.URLEncoder;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import
javax.xml.bind.DatatypeConverter;


public class CordieEditor extends
HttpServlet {
   private Map<String, CordieDiagram>
diagrams;

   @Override
   public void init(ServletConfig config)
throws ServletException {
      super.init(config);
      //contextPath =
config.getServletContext().getContextPa
th();

      CordieDiagramsMap cdm =
CordieDiagramsMap.getInstance();
      diagrams = cdm.getMap();
   }
```

```
    @Override
    protected void
doGet(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException
{
        response.sendRedirect("index.jsp");
    }

    @Override
    protected void
doPost(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException
{
        String action =
request.getParameter("action");
        String id =
request.getParameter("id");
        String username = (String)
request.getSession().getAttribute("USER
NAME");
        String editorID =
request.getParameter("editorID");

        PrintWriter writer =
response.getWriter();

        if(action.equals("newUser")) {
//<editor-fold>
            synchronized(diagrams) { // avoid
errors when two or more users open a
diagram all at the same time
                if(!diagrams.containsKey(id)) {
                    Connection connection =
null;
                    ResultSet rs;

                    try {

Class.forName("com.mysql.jdbc.Driver")
;
                        connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

                        String sql = "SELECT *
FROM collaborator WHERE
c_diagram_id = ? AND c_username = ?";
                        PreparedStatement stmt =
connection.prepareStatement(sql);
                        stmt.setInt(1,
Integer.parseInt(id));
                        stmt.setString(2,
username);
                        rs = stmt.executeQuery();

                        if(!rs.next()) {
                            writer.println("{\"error\"
: \"This diagram does not exist, no
longer available, or you do not have
permission to edit.\", \"errorType\" :
1}");
                            writer.close();
                            return;
                        }
                    } catch (Exception e) {

System.err.println("Exception: " +
e.getMessage());
                        writer.println("{\"error\" :
\"An error occured.\", \"errorType\" :
1}");
                        writer.close();
                        return;
                    }

                    diagrams.put(id, new
CordieDiagram(id));
                } else
if(!diagrams.get(id).allowsEditFrom(user
name)) {
                    writer.println("{\"error\" :
\"You do not have permission to edit
this diagram.\", \"errorType\" : 1}");
                    writer.close();
                    return;
                }


//diagrams.get(id).addEditor(username)
;

//writer.println(diagrams.get(id));


writer.println(diagrams.get(id).addEdito
r(username));
            }
            return;
        } //</editor-fold>

        if(action.equals("addcollaborator"))
{ //<editor-fold>
            String newCollaborator =
request.getParameter("collaborator");

            Connection connection = null;
            ResultSet rs;

            // If diagram is not open
            if(!diagrams.containsKey(id)) { //
<editor-fold>
                // Check if user is the creator
                try {

Class.forName("com.mysql.jdbc.Driver")
;
                    connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

                    String sql = "SELECT
COUNT(*) FROM collaborator WHERE
c_diagram_id = ? AND c_username = ?
AND is_creator = ?";
                    PreparedStatement stmt =
connection.prepareStatement(sql);
                    stmt =
connection.prepareStatement(sql);
                    stmt.setInt(1,
Integer.parseInt(id));
                    stmt.setString(2, username);
                    stmt.setString(3, "YES");
                    rs = stmt.executeQuery();
                    rs.next();
                    if(rs.getInt(1) != 1) {
                        writer.println("{\"error\" :
\"You are not allowed to add
collaborators.\", \"errorType\" : 2}");
                        return;
                    } else { // <editor-fold>
                        try {

Class.forName("com.mysql.jdbc.Driver")
;
                            connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

                            String ncFirstname = "";
                            String ncLastname = "";
                            String ncEmail = "";
                            String ncDisplaypic = "";

                            // Get user info
                            sql = "SELECT firstname,
lastname, email, displaypic "
                                    + "FROM user
WHERE username = ?";
                            stmt =
connection.prepareStatement(sql);
                            stmt.setString(1,
newCollaborator);
                            rs =
stmt.executeQuery();
                            if(rs.next()) {
                                ncFirstname =
rs.getString("firstname");
                                ncLastname =
rs.getString("lastname");
                                ncEmail =
rs.getString("email");
                                ncDisplaypic =
rs.getString("displaypic");
                            } else {

writer.println("{\"error\" : \"User with
the username "
                                    +
newCollaborator + " does not exist.\",
\"errorType\" : 2}");
                                writer.close();
                                return;
                            }

                            // Check if the user is
already a collaborator
                            sql = "SELECT COUNT(*)
FROM collaborator WHERE
c_diagram_id = ? AND c_username = ?";
                            stmt =
connection.prepareStatement(sql);
```

```
                    stmt.setInt(1,
Integer.parseInt(id));
                    stmt.setString(2,
newCollaborator);
                    rs =
stmt.executeQuery();
                    rs.next();
                    if(rs.getInt(1) == 1) {

writer.println("{\"error\" : \"" +
newCollaborator +
                            " is already a
collaborator.\", \"errorType\" : 2}");
                        writer.close();
                        return;
                    }

                    // Update database
                    sql = "INSERT INTO
Cordie.collaborator (c_diagram_id,
c_username, is_creator) "
                            + "VALUES (?, ?,
'NO')";
                    stmt =
connection.prepareStatement(sql);
                    stmt.setInt(1,
Integer.parseInt(id));
                    stmt.setString(2,
newCollaborator);
                    stmt.executeUpdate();


writer.println("{\"response\" : \" " +
newCollaborator + " is being added as
collaborator.\"}");
                    writer.close();
                    return;
                } catch (Exception e) {

System.err.println("Exception: " +
e.getMessage());
                    writer.println("{\"error\"
: \"An error occured.\", \"errorType\" :
1}");
                    writer.close();
                    return;
                }
                // </editor-fold>
            }
        } catch(Exception e) {

System.err.println("Exception: " +
e.getMessage());
            writer.println("{\"error\" :
\"An error occured.\", \"errorType\" :
1}");
            writer.close();
            return;
        }
        // </editor-fold>
    }

    // Only creator of the diagram
can add collaborators
if(!diagrams.get(id).getCreator().equals(
username)) {
        writer.println("{\"error\" :
\"You are not allowed to add
collaborators.\", \"errorType\" : 2}");
        return;
    }

    try { // <editor-fold>

Class.forName("com.mysql.jdbc.Driver")
;
        connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

        String ncFirstname = "";
        String ncLastname = "";
        String ncEmail = "";
        String ncDisplaypic = "";

        // Get user info
        String sql = "SELECT firstname,
lastname, email, displaypic "
                + "FROM user WHERE
username = ?";
        PreparedStatement stmt =
connection.prepareStatement(sql);
        stmt.setString(1,
newCollaborator);
        rs = stmt.executeQuery();
        if(rs.next()) {
            ncFirstname =
rs.getString("firstname");
            ncLastname =
rs.getString("lastname");
            ncEmail =
rs.getString("email");
            ncDisplaypic =
rs.getString("displaypic");
        } else {
            writer.println("{\"error\" :
\"User with the username "
                    + newCollaborator + "
does not exist.\", \"errorType\" : 2}");
            writer.close();
            return;
        }

        // Check if the user is already a
collaborator
        sql = "SELECT COUNT(*) FROM
collaborator WHERE c_diagram_id = ?
AND c_username = ?";
        stmt =
connection.prepareStatement(sql);
        stmt.setInt(1,
Integer.parseInt(id));
        stmt.setString(2,
newCollaborator);
        rs = stmt.executeQuery();
        rs.next();
        if(rs.getInt(1) == 1) {
            writer.println("{\"error\" :
\"" + newCollaborator +
                    " is already a
collaborator.\", \"errorType\" : 2}");
            writer.close();
            return;
        }

        // Update database
        sql = "INSERT INTO
Cordie.collaborator (c_diagram_id,
c_username, is_creator) "
                + "VALUES (?, ?, 'NO')";
        stmt =
connection.prepareStatement(sql);
        stmt.setInt(1,
Integer.parseInt(id));
        stmt.setString(2,
newCollaborator);
        stmt.executeUpdate();

        // Send out an AddCollaborator
operation to the diagram
        CordieOperation co = new
CordieAddCollaboratorOperation(newC
ollaborator,
                ncFirstname, ncLastname,
ncEmail, ncDisplaypic);

diagrams.get(id).queueMessage(new
CordieMessage(null, co, 0, 0));

        writer.println("{\"response\" :
\" " + newCollaborator + " is being
added as collaborator.\"}");
        writer.close();
        return;
        // </editor-fold>
    } catch (Exception e) {
        System.err.println("Exception:
" + e.getMessage());
        writer.println("{\"error\" : \"An
error occured.\", \"errorType\" : 1}");
        writer.close();
        return;
    }
} //</editor-fold>


if(action.equals("removecollaborator"))
{ // <editor-fold>
    String toRemove =
request.getParameter("collaborator");

    Connection connection = null;
    ResultSet rs;

    // If diagram is not open
    if(!diagrams.containsKey(id)) {
        // Check if user is the creator
        try {

Class.forName("com.mysql.jdbc.Driver")
;
            connection =
DriverManager.getConnection("jdbc:my
```

```
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

            String sql = "SELECT
COUNT(*) FROM collaborator WHERE
c_diagram_id = ? AND c_username = ?
AND is_creator = ?";
            PreparedStatement stmt =
connection.prepareStatement(sql);
            stmt =
connection.prepareStatement(sql);
            stmt.setInt(1,
Integer.parseInt(id));
            stmt.setString(2, username);
            stmt.setString(3, "YES");
            rs = stmt.executeQuery();
            rs.next();
            if(rs.getInt(1) != 1) {
               writer.println("{\"error\" :
\"You are not allowed to remove
collaborators.\", \"errorType\" : 2}");
               return;
            } else { // <editor-fold>
               try { // <editor-fold>

Class.forName("com.mysql.jdbc.Driver")
;
                  connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

                  // Check if the user is a
collaborator
                  sql = "SELECT * FROM
collaborator WHERE c_diagram_id = ?
AND c_username = ?";
                  stmt =
connection.prepareStatement(sql);
                  stmt.setInt(1,
Integer.parseInt(id));
                  stmt.setString(2,
toRemove);
                  rs =
stmt.executeQuery();

                  if(rs.next()) {

if(rs.getString("is_creator").equals("YES"
)) {

writer.println("{\"error\" : \"The creator
of this diagram "
                           + "cannot be
removed from the list of
collaborators.\", \"errorType\" : 2}");
                     writer.close();
                     return;
                  }
                  } else {

writer.println("{\"error\" : \"" +
toRemove +
                        " is not a
collaborator of this diagram.\",
\"errorType\" : 2}");
                     writer.close();
                     return;
                  }

                  // Update database
                  sql = "DELETE FROM
collaborator WHERE c_diagram_id = ? "
                        + "AND c_username
= ? AND is_creator = 'NO'";
                  stmt =
connection.prepareStatement(sql);
                  stmt.setInt(1,
Integer.parseInt(id));
                  stmt.setString(2,
toRemove);
                  stmt.executeUpdate();


writer.println("{\"response\" : \"" +
toRemove + "is being removed as
collaborator.\"}");
                  writer.close();
                  return;
                  // </editor-fold>
               } catch (Exception e) {

System.err.println("Exception: " +
e.getMessage());
                  writer.println("{\"error\"
: \"An error occured.\", \"errorType\" :
1}");
                  writer.close();
                  return;
               }
               // </editor-fold>
            }
         } catch(Exception e) {

System.err.println("Exception: " +
e.getMessage());
            writer.println("{\"error\" :
\"An error occured.\", \"errorType\" :
1}");
            writer.close();
            return;
         }
         // </editor-fold>
      }

      // Only creator of the diagram
can remove collaborators

if(!diagrams.get(id).getCreator().equals(
username)) {
         writer.println("{\"error\" :
\"You are not allowed to remove other
collaborators.\", \"errorType\" : 2}");
         return;
      }

      try { // <editor-fold>

Class.forName("com.mysql.jdbc.Driver")
;
         connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

         // Check if the user is a
collaborator
         String sql = "SELECT * FROM
collaborator WHERE c_diagram_id = ?
AND c_username = ?";
         PreparedStatement stmt =
connection.prepareStatement(sql);
         stmt.setInt(1,
Integer.parseInt(id));
         stmt.setString(2, toRemove);
         rs = stmt.executeQuery();

         if(rs.next()) {

if(rs.getString("is_creator").equals("YES"
)) {
            writer.println("{\"error\" :
\"The creator of this diagram "
                  + "cannot be removed
from the list of collaborators.\",
\"errorType\" : 2}");
            writer.close();
            return;
         }
         } else {
            writer.println("{\"error\" :
\"" + toRemove +
               " is not a collaborator of
this diagram.\", \"errorType\" : 2}");
            writer.close();
            return;
         }

         // Update database
         sql = "DELETE FROM
collaborator WHERE c_diagram_id = ? "
            + "AND c_username = ?
AND is_creator = 'NO'";
         stmt =
connection.prepareStatement(sql);
         stmt.setInt(1,
Integer.parseInt(id));
         stmt.setString(2, toRemove);
         stmt.executeUpdate();

         // Send out a
RemoveCollaborator operation to the
diagram
         CordieOperation co = new
CordieRemoveCollaboratorOperation(to
Remove);

diagrams.get(id).queueMessage(new
CordieMessage(null, co, 0, 0));

         writer.println("{\"response\" :
\"" + toRemove + "is being removed as
collaborator.\"}");
         writer.close();
         return;
         // </editor-fold>
      } catch (Exception e) {
```

```
            System.err.println("Exception:
" + e.getMessage());
            writer.println("{\"error\" : \"An
error occured.\", \"errorType\" : 1}");
            writer.close();
            return;
         }
      } //</editor-fold>


      if(!diagrams.containsKey(id)) {
         //writer.println("{\"error\" :
\"Sending message failed. Please reload
the page.\", \"errorType\" : 1}");
         writer.println("{\"error\" : \"This
diagram no longer exists.\",
\"errorType\" : 1}");
         return;
      }


if(!diagrams.get(id).hasCurrentEditor(ed
itorID)) {
         writer.println("{\"error\" : \"You
no longer have permission to edit this
diagram.\", \"errorType\" : 1}");
         return;
      }

      if(action.equals("send")) {
//<editor-fold>
         int myMsgs =
Integer.parseInt(request.getParameter(
"myMsgs"));
         int otherMsgs =
Integer.parseInt(request.getParameter(
"otherMsgs"));

         CordieOperation co;
         String optype =
request.getParameter("optype");

         if(optype.equals("insert")) {
            co = new
CordieInsertOperation(request.getPara
meterMap());
         } else if(optype.equals("delete"))
{
            co = new
CordieDeleteOperation(

Integer.parseInt(request.getParameter(
"position")));
         } else if(optype.equals("edit")) {

if(request.getParameter("value") != null)
{
               co = new
CordieEditOperation(

Integer.parseInt(request.getParameter(
"position")),

request.getParameter("attribute"),

request.getParameter("value"));
            } else {
               co = new
CordieEditOperation(request.getParame
terMap());
            }
         } else if(optype.equals("move")) {
            co = new
CordieMoveOperation(

Integer.parseInt(request.getParameter(
"position")),

Integer.parseInt(request.getParameter(
"destination")));
         } else {
            co = new
CordieNoneOperation();
         }


diagrams.get(id).queueMessage(new
CordieMessage(editorID, co, myMsgs,
otherMsgs));

         return;
      } //</editor-fold>

      if(action.equals("createImage"))
{//<editor-fold>
         String value =
request.getParameter("data");

         byte[] btDataFile =
DatatypeConverter.parseBase64Binary(
value);
         File of = new File("files/" +
URLEncoder.encode(username + "_" +
editorID + "_" + id + ".png"));
         FileOutputStream osf = new
FileOutputStream(of);
         osf.write(btDataFile);
         osf.flush();
         if (osf != null) {
            try {
               osf.close();
            } catch (IOException e) {
               e.printStackTrace();
            }
         }
         return;
      } //</editor-fold>

      if(action.equals("askUpdate")) {
//<editor-fold>

if(diagrams.get(id).messageQueueEmpt
y() &&

diagrams.get(id).getClient(editorID).has
Updates()) {

writer.println(diagrams.get(id).getClient
(editorID).getUpdates());
         } else {
         }
      } //</editor-fold>
   }
}
```

DiagramList.java

```
package cordie;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Map;
import cordie.diagram.*;
import cordie.diagram.operation.*;
import java.io.File;
import java.io.FileWriter;
import java.sql.PreparedStatement;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.output.XMLOutputter;
import org.jdom.output.Format;
import
org.apache.commons.io.FileUtils;
import
org.apache.commons.lang3.StringEscap
eUtils;

public class DiagramList extends
HttpServlet {

   @Override
   protected void
doPost(HttpServletRequest request,
HttpServletResponse response)
   throws ServletException, IOException
{

response.setContentType("text/html;ch
arset=UTF-8");
      PrintWriter out =
response.getWriter();

      Connection connection = null;
      ResultSet rs;

      try {

Class.forName("com.mysql.jdbc.Driver")
;
         connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

         String username = (String)
request.getSession().getAttribute("USER
NAME");
```

```
        String action =
request.getParameter("action");
        if(action == null) {
           // do nothing
        } else if(action.equals("add")) {
           // <editor-fold>
           String sql = "INSERT INTO
Cordie.diagram (creator, title, "
                 + "date_last_edited,
date_created, description) "
                 + "VALUES (?, ?, DEFAULT,
CURRENT_TIMESTAMP, ?)";
           PreparedStatement stmt =
connection.prepareStatement(sql);
           stmt.setString(1, username);
           stmt.setString(2,
request.getParameter("title"));
           stmt.setString(3,
request.getParameter("description"));
           stmt.executeUpdate();

           // get id of newly created
diagram
           sql = "SELECT diagram_id
FROM diagram WHERE creator = ? AND
title = ? "
                 + "ORDER BY date_created
DESC";
           stmt =
connection.prepareStatement(sql);
           stmt.setString(1, username);
           stmt.setString(2,
request.getParameter("title"));
           rs = stmt.executeQuery();

           String id = "";
           if(rs.next()) {
              id =
rs.getString("diagram_id");
           }

           // create an XML file for the
newly created diagram
           Element root = new
Element("diagram");
           Document dom = new
Document(root);
           XMLOutputter outputter = new
XMLOutputter(Format.getPrettyFormat(
));
           FileWriter writer = new
FileWriter("cordie_diagram_files/" + id
+ ".xml");
           outputter.output(dom, writer);
           writer.close();
           // </editor-fold>
        } else if(action.equals("delete")) {
           // <editor-fold>
           String id =
request.getParameter("id");

           // check if diagram has active
users
           CordieDiagramsMap cdm =
CordieDiagramsMap.getInstance();
           Map<String, CordieDiagram>
diagrams = cdm.getMap();
           synchronized(diagrams) {
              if(diagrams.containsKey(id)) {

//diagrams.get(id).cancelTimer();
                 diagrams.remove(id);
              }
           }

           // delete from database
           String sql = "DELETE FROM
diagram WHERE diagram_id = ? AND
creator = ?";
           PreparedStatement stmt =
connection.prepareStatement(sql);
           stmt.setInt(1,
Integer.parseInt(id));
           stmt.setString(2, username);
           stmt.executeUpdate();

           // delete XML file
           File xmlFile = new
File("cordie_diagram_files/" + id +
".xml");
           xmlFile.delete();
           // </editor-fold>
        } else if(action.equals("leave")) {
           // <editor-fold>
           String id =
request.getParameter("id");
           String sql = "DELETE FROM
collaborator WHERE c_diagram_id = ? "
                 + "AND c_username = ?
AND is_creator = 'NO'";
           PreparedStatement stmt =
connection.prepareStatement(sql);
           stmt.setInt(1,
Integer.parseInt(id));
           stmt.setString(2, username);
           stmt.executeUpdate();

           // create message for users
currently editing the diagram
           CordieDiagramsMap cdm =
CordieDiagramsMap.getInstance();
           Map<String, CordieDiagram>
diagrams = cdm.getMap();
           if(diagrams.containsKey(id)) {
              CordieOperation co = new
CordieRemoveCollaboratorOperation(us
ername);

diagrams.get(id).queueMessage(new
CordieMessage(null, co, 0, 0));
           }
           // </editor-fold>
        } else if(action.equals("copy")) {
           // <editor-fold>
           String sql = "INSERT INTO
Cordie.diagram (creator, title, "
                 + "date_last_edited,
date_created, description) "
                 + "VALUES (?, ?, DEFAULT,
CURRENT_TIMESTAMP, ?)";
           PreparedStatement stmt =
connection.prepareStatement(sql);
           stmt.setString(1, username);
           stmt.setString(2,
request.getParameter("title"));
           stmt.setString(3,
request.getParameter("description"));
           stmt.executeUpdate();

           // get id of the copy diagram
           sql = "SELECT diagram_id
FROM diagram WHERE creator = ? AND
"
                 + "title = ? ORDER BY
date_created DESC";
           stmt =
connection.prepareStatement(sql);
           stmt.setString(1, username);
           stmt.setString(2,
request.getParameter("title"));
           rs = stmt.executeQuery();

           String id = "";
           if(rs.next()) {
              id =
rs.getString("diagram_id");
           }

           // copy tne XML file of the
original diagram into the XML file of this
copy
           File srcFile = new
File("cordie_diagram_files/" +

request.getParameter("from") + ".xml");
           File destFile = new
File("cordie_diagram_files/" + id +
".xml");
           FileUtils.copyFile(srcFile,
destFile);

           // </editor-fold>
        }

        String sql = "SELECT diagram.*
FROM diagram INNER JOIN
collaborator"
              + " ON diagram.diagram_id =
collaborator.c_diagram_id"
              + " WHERE
collaborator.c_username = ?";
        PreparedStatement stmt =
connection.prepareStatement(sql);
        stmt.setString(1, username);
        rs = stmt.executeQuery();

        out.print("[");
        while(rs.next()) {
           //Connection connection2 =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");
           String sql2 = "SELECT
c_username FROM collaborator WHERE
c_diagram_id = ?";
```

```
            PreparedStatement stmt2 =
connection.prepareStatement(sql2);
            stmt2.setInt(1,
Integer.parseInt(rs.getString("diagram_i
d")));
            ResultSet rs2 =
stmt2.executeQuery();

            String collaborators = "[";
            while(rs2.next()) {
               collaborators += "\"" +
rs2.getString("c_username")+ "\"";
               if(!rs2.isLast()) collaborators
+= ", ";
            }
            collaborators += "]";

            String id =
rs.getString("diagram_id");
            String title =
rs.getString("title");
            String description =
rs.getString("description");
            String creator =
rs.getString("creator");
            String dateCreated =
rs.getString("date_created");
            //String dateLastEdited =
rs.getString("date_last_edited");

            out.print("[\"" + id + "\", \"" +
StringEscapeUtils.escapeJava(title)
                  + "\", \"" +
StringEscapeUtils.escapeJava(descriptio
n)
                  + "\", \"" +
StringEscapeUtils.escapeJava(creator) +
"\", \""
                  + dateCreated + "\", " +
collaborators + "]");
                  //"\", \"" + dateLastEdited
+ "\"]");

            if(!rs.isLast()) out.print(", ");
         }
         out.print("]");
      } catch (ClassNotFoundException e)
{
         System.err.println("Driver
Error");
      } catch (SQLException e) {

System.err.println("SQLException: " +
e.getMessage());
      } finally {
         out.close();
      }
   }

   @Override
   protected void
doGet(HttpServletRequest request,
HttpServletResponse response)
   throws ServletException, IOException
{
response.sendRedirect("diagrams.jsp");
   }

   @Override
   public String getServletInfo() {
      return "Handles request for the list
of diagrams "
            + "accessible to a user (in JSON
format).";
   }
}
```

DisplayPicture.java

```
package cordie;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;

import java.io.*;

public class DisplayPicture extends
HttpServlet {

   protected void
processRequest(HttpServletRequest
request, HttpServletResponse response)
   throws ServletException, IOException
{
      File image = new
File("uploaded_images/" +
request.getParameter("imgTitle"));

      //get the image file
      if (image.exists()) {//check whether
it exist and if it is
         String contentType =
getServletContext().getMimeType(imag
e.getName());

         int BUFFER_SIZE = 10240;
         // Set servlet response.
         response.reset();

response.setBufferSize(BUFFER_SIZE );

response.setContentType(contentType);
         response.setHeader("Content-
Length",
String.valueOf(image.length()));

         // Prepare streams.
         BufferedInputStream input = null;
         BufferedOutputStream output =
null;

         try {
            // Open streams.
            input = new
BufferedInputStream(new
FileInputStream(image), BUFFER_SIZE);
            output = new
BufferedOutputStream(response.getOut
putStream(), BUFFER_SIZE);

            // Write image to response.
            byte[] buffer = new
byte[BUFFER_SIZE];
            int length;
            while ((length =
input.read(buffer)) > 0) {
               output.write(buffer, 0,
length);
            }
         } finally {
            // Close streams.
            output.close();
            input.close();
         }
      } else {
         //System.out.println("Could not
Find the image");
      }
   }

   // <editor-fold
defaultstate="collapsed"
desc="HttpServlet methods. Click on the
+ sign on the left to edit the code.">
   /**
    * Handles the HTTP
<code>GET</code> method.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a
servlet-specific error occurs
    * @throws IOException if an I/O
error occurs
    */
   @Override
   protected void
doGet(HttpServletRequest request,
HttpServletResponse response)
   throws ServletException, IOException
{
      processRequest(request, response);
   }

   /**
    * Handles the HTTP
<code>POST</code> method.
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a
servlet-specific error occurs
    * @throws IOException if an I/O
error occurs
    */
   @Override
   protected void
doPost(HttpServletRequest request,
HttpServletResponse response)
   throws ServletException, IOException
{
      processRequest(request, response);
   }
```

```
    /**
     * Returns a short description of the
servlet.
     * @return a String containing servlet
description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

FileServlet.java

```
package cordie;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.Closeable;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URLEncoder;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;

/**
 * The File servlet for serving from
absolute path.
 * @author BalusC
 * @link
http://balusc.blogspot.com/2007/07/fil
eservlet.html
 */
public class FileServlet extends
HttpServlet {
    // Constants ----------------------------------
------------------------------------------------
    private static final int
DEFAULT_BUFFER_SIZE = 10240; //
10KB.

    // Properties ---------------------------------
------------------------------------------------
    private String filePath;

    // Actions ------------------------------------
------------------------------------------------
    public void init() throws
ServletException {
        // Define base path somehow. You
can define it as init-param of the servlet.
        this.filePath = "files";

        // In a Windows environment with
the Applicationserver running on the
        // c: volume, the above path is
exactly the same as "c:\files".
        // In UNIX, it is just straightforward
"/files".
    }

    protected void
doGet(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException,
IOException
    {
        File file = new File(filePath,
URLEncoder.encode((String)
request.getSession().getAttribute("USER
NAME")
                + "_" +
request.getParameter("editorID") + "_"
+ request.getParameter("id") + ".png"));

        // Check if file actually exists in
filesystem.
        if (!file.exists()) {
            // Do your thing if the file
appears to be non-existing.
            // Throw an exception, or send
404, or show default/warning page, or
just ignore it.

response.sendError(HttpServletRespons
e.SC_NOT_FOUND); // 404.
            return;
        }

        // Get content type by filename.
        String contentType =
getServletContext().getMimeType(file.g
etName());

        // If content type is unknown, then
set the default value.
        // For all content types, see:
http://www.w3schools.com/media/me
dia_mimeref.asp
        // To add new content types, add
new mime-mapping entry in web.xml.
        if (contentType == null) {
            contentType =
"application/octet-stream";
        }

        // Init servlet response.
        response.reset();

response.setBufferSize(DEFAULT_BUFFE
R_SIZE);

response.setContentType(contentType);
        response.setHeader("Content-
Length", String.valueOf(file.length()));
        response.setHeader("Content-
Disposition", "attachment; filename=\""
+ file.getName() + "\"");

        // Prepare streams.
        BufferedInputStream input = null;
        BufferedOutputStream output =
null;

        try {
            // Open streams.
            input = new
BufferedInputStream(new
FileInputStream(file),
DEFAULT_BUFFER_SIZE);
            output = new
BufferedOutputStream(response.getOut
putStream(), DEFAULT_BUFFER_SIZE);

            // Write file contents to
response.
            byte[] buffer = new
byte[DEFAULT_BUFFER_SIZE];
            int length;
            while ((length =
input.read(buffer)) > 0) {
                output.write(buffer, 0, length);
            }
        } finally {
            // Gently close streams.
            close(output);
            close(input);
        }
    }

    // Helpers (can be refactored to
public utility class) ----------------------------
------------

    private static void close(Closeable
resource) {
        if (resource != null) {
            try {
                resource.close();
            } catch (IOException e) {
                // Do your thing with the
exception. Print it, log it or mail it.
                e.printStackTrace();
            }
        }
    }

}
```

LogInExec.java

```
package cordie;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class LogInExec extends
HttpServlet {
```

```
    public void
processRequest(HttpServletRequest
request, HttpServletResponse response)
        throws ServletException,
IOException {
        Connection connection = null;
        ResultSet rs;

        try {

Class.forName("com.mysql.jdbc.Driver")
;
            connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

            String username =
request.getParameter("username");
            String password =
request.getParameter("password");

            String sql = "SELECT * FROM user
WHERE username = ? AND password =
md5(?)";
            PreparedStatement stmt =
connection.prepareStatement(sql);
            stmt.setString(1, username);
            stmt.setString(2, password);
            rs = stmt.executeQuery();

            if(rs.next()) {

request.getSession().setAttribute("USER
NAME", rs.getString("username"));

request.getSession().setAttribute("FIRST
NAME", rs.getString("firstname"));

request.getSession().setAttribute("LAST
NAME", rs.getString("lastname"));

request.getSession().setAttribute("EMAI
L", rs.getString("email"));

request.getSession().setAttribute("DISPL
AYPIC", rs.getString("displaypic"));


response.sendRedirect("index.jsp");
            } else {

response.sendRedirect("loginfailed.jsp")
;
            }
        } catch (ClassNotFoundException e)
{
            System.err.println("Driver
Error");
        } catch (SQLException e) {

System.err.println("SQLException: " +
e.getMessage());
        }
    }

    @Override
    protected void
doGet(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException,
IOException {
        processRequest(request, response);
    }

    @Override
    protected void
doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException,
IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }
}
```

CordieDiagram.java

```
package cordie.diagram;

import cordie.diagram.operation.*;
import java.util.Map;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.ListIterator;
import
java.util.concurrent.ArrayBlockingQueu
e;
import
java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;
import org.jdom.output.Format;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashSet;
import java.util.Set;
import
org.apache.commons.lang3.StringEscap
eUtils;

public class CordieDiagram {
    // <editor-fold>
    private String diagramID;
    private String diagramTitle;
    private String creator;
    private Date dateCreated;
    private List<String> collaboratorList;
    private Map<String, String[]>
collaboratorInfo;

    private Map<String, CordieClient>
currentEditors;
    private ArrayList<String>
currentUsers;
    private int editorsCount;
    private
ArrayBlockingQueue<CordieMessage>
messageQueue;
    private XMLOutputter outputter;

    private Document dom;
    private Element root;

    //private final long
CHECKING_PERIOD = 5 * 60 * 1000; //
check user activity every 5 mins
    //private final long
MAX_IDLE_ALLOWED = 10 * 60 * 1000;
// allow users to be idle for less than 10
minutes
    private final long CHECKING_PERIOD =
10 * 1000; // check user activity every
10 sec
    private final long
MAX_IDLE_ALLOWED = 25 * 1000; //
allow users to be idle for less than 1
minute
    private Timer timer;
    // </editor-fold>

    public CordieDiagram(String
iDiagramID) {
        // Parse XML file
        SAXBuilder builder = new
SAXBuilder();
        try {
            dom = builder.build(new
File("cordie_diagram_files/" +
iDiagramID + ".xml"));
        } catch(JDOMException e) {
            e.printStackTrace();
        } catch(IOException e) {
            e.printStackTrace();
        }

        this.root = dom.getRootElement();
        this.diagramID = iDiagramID;


setCollaboratorsAndOtherAttributes();

        this.currentEditors = new
HashMap<String, CordieClient>();
        this.currentUsers = new
ArrayList<String>();
        this.editorsCount = 0;
        this.messageQueue = new
ArrayBlockingQueue<CordieMessage>(2
00);
```

```
      this.outputter = new
XMLOutputter(Format.getPrettyFormat(
));

      // Run this diagram's message
processor
      ExecutorService executor =
Executors.newFixedThreadPool(1);
      try {
         executor.execute(new
CordieMessageDirector());
      } catch(Exception exception){
         exception.printStackTrace();
      }
      executor.shutdown();

      // Set up a TimerTask object to
monitor user activity
      timer = new Timer();
      timer.scheduleAtFixedRate(new
MonitorEditors(), 5 * 1000,
CHECKING_PERIOD);
   }

   private void
setCollaboratorsAndOtherAttributes() {
      Connection connection = null;
      String sql = "";
      PreparedStatement stmt;
      ResultSet rs;

      this.collaboratorList = new
ArrayList<String>();
      this.collaboratorInfo = new
HashMap<String, String[]>();

      try {

Class.forName("com.mysql.jdbc.Driver")
;
         connection =
DriverManager.getConnection("jdbc:my
sql://localhost:3306/Cordie", "Cordie",
"pSJcwyTNSeLHAAV2");

         sql = "SELECT user.username,
user.firstname, user.lastname,
user.email,"
               + " user.displaypic FROM
user INNER JOIN collaborator "
               + "ON user.username =
collaborator.c_username "
               + "WHERE c_diagram_id = ?";
         stmt =
connection.prepareStatement(sql);
         stmt.setInt(1,
Integer.parseInt(diagramID));
         rs = stmt.executeQuery();

         while(rs.next()) {

collaboratorList.add(rs.getString("usern
ame"));
            String[] temp = {
rs.getString("firstname"),
rs.getString("lastname"),
                  rs.getString("email"),
rs.getString("displaypic")};

this.collaboratorInfo.put(rs.getString("u
sername"), temp);
         }

         sql = "SELECT title, creator,
date_created FROM diagram WHERE
diagram_id = ?";
         stmt =
connection.prepareStatement(sql);
         stmt.setInt(1,
Integer.parseInt(diagramID));
         rs = stmt.executeQuery();

         if(rs.next()) {
            this.diagramTitle =
rs.getString("title");
            this.creator =
rs.getString("creator");
            this.dateCreated =
rs.getDate("date_created");
         }
      } catch (ClassNotFoundException e)
{
         System.err.println("Driver
Error");
      } catch (SQLException e) {

System.err.println("SQLException: " +
e.getMessage());
      }
   }

   public boolean isInactive() {
      return currentEditors.isEmpty();
   }

   public boolean allowsEditFrom(String
username) {
      return
collaboratorList.contains(username);
   }

   public String getCreator() {
      return creator;
   }

   public String addEditor(String
username) {
      String editorID =
Integer.toString(editorsCount);
      synchronized(currentEditors) {
         //System.out.println(editorID + "
is now an editor");
         currentEditors.put(editorID, new
CordieClient(username));
         editorsCount++;

if(!currentUsers.contains(username)) {
            currentUsers.add(username);
         }

         CordieOperation co = new
CordieAddUserOperation(username);
         this.queueMessage(new
CordieMessage(editorID, co, 0, 0));
      }

      return "{\"diagram\" : " +
this.toString() + ", \"editorID\" : \"" +
            editorID + "\"}";
      //return editorID;
   }

   public void
queueMessage(CordieMessage msg) {
      try {
         messageQueue.put(msg);
      } catch(Exception exception) {
         exception.printStackTrace();
      }
   }

   /*public
ArrayBlockingQueue<CordieMessage>
getMessageQueue() {
      return messageQueue;
   }*/

   public boolean
messageQueueEmpty() {
      return messageQueue.isEmpty();
   }

   public CordieClient getClient(String
editorID) {
      return currentEditors.get(editorID);
   }

   public boolean
hasCurrentEditor(String editorID) {
      return
currentEditors.containsKey(editorID);
   }

   public void apply(CordieOperation op,
String sender) {
      List doChildren = root.getChildren();

      // Apply to server copy
      if(op instanceof
CordieInsertOperation) { // <editor-
fold>
         CordieInsertOperation cio =
(CordieInsertOperation) op;
         doChildren.add(cio.getPosition(),
cio.getObject());
         // </editor-fold>
      } else if(op instanceof
CordieDeleteOperation) { // <editor-
fold>
         CordieDeleteOperation cdo =
(CordieDeleteOperation) op;

doChildren.remove(cdo.getPosition());
         // </editor-fold>
      } else if(op instanceof
CordieEditOperation) { // <editor-fold>
         CordieEditOperation ceo =
(CordieEditOperation) op;
```

```
        Element target = (Element)
doChildren.get(ceo.getPosition());

        if(ceo.getValue() != null) {

target.setAttribute(ceo.getAttribute(),
ceo.getValue());
        } else if
(ceo.getAttribute().equals("attribute")
||
ceo.getAttribute().equals("operation")
||

ceo.getAttribute().equals("template") ||
ceo.getAttribute().equals("taggedvalue"
) ||

ceo.getAttribute().equals("containedarti
fact") ||
ceo.getAttribute().equals("internalactivi
ty")) {
            CordieOperation attrOp =
ceo.getAttributeOp();
            List attrItems =
target.getChild(ceo.getAttribute() +
"s").getChildren();

            if(attrOp instanceof
CordieInsertOperation) {
                CordieInsertOperation
attrInsertOp = (CordieInsertOperation)
attrOp;

attrItems.add(attrInsertOp.getPosition()
, attrInsertOp.getObject());
            } else if(attrOp instanceof
CordieDeleteOperation) {

attrItems.remove(((CordieDeleteOperati
on) attrOp).getPosition());
            } else if(attrOp instanceof
CordieEditOperation) {
                CordieEditOperation
attrEditOp = (CordieEditOperation)
attrOp;

if(attrEditOp.getAttribute().equals("attri
buteValue")) {
                    ((Element)
attrItems.get(attrEditOp.getPosition())).
setText(attrEditOp.getValue());
                } else
if(attrEditOp.getAttribute().equals("attri
buteStatic")) {
                    ((Element)
attrItems.get(attrEditOp.getPosition())).
setAttribute("static",
attrEditOp.getValue());
                } else
if(attrEditOp.getAttribute().equals("attri
buteAbstract")) {
                    ((Element)
attrItems.get(attrEditOp.getPosition())).
setAttribute("abstract",
attrEditOp.getValue());
                }
            } else if(attrOp instanceof
CordieMoveOperation) {
                CordieMoveOperation
attrMoveOp = (CordieMoveOperation)
attrOp;
                Element attrTarget =
(Element)
attrItems.remove(attrMoveOp.getPositi
on());

attrItems.add(attrMoveOp.getDestinati
on(), attrTarget);
            }
        } else
if(ceo.getAttribute().equals("point")) {
            CordieOperation attrOp =
ceo.getAttributeOp();
            List points =
target.getChildren("point");

            if(attrOp instanceof
CordieInsertOperation) {
                CordieInsertOperation
attrInsertOp = (CordieInsertOperation)
attrOp;

points.add(attrInsertOp.getPosition(),
attrInsertOp.getObject());
            } else if(attrOp instanceof
CordieDeleteOperation) {

points.remove(((CordieDeleteOperation
) attrOp).getPosition());
            } else if(attrOp instanceof
CordieEditOperation) {
                CordieEditOperation
attrEditOp = (CordieEditOperation)
attrOp;

if(attrEditOp.getAttribute().equals("attri
buteX")) {
                    ((Element)
points.get(attrEditOp.getPosition())).set
Attribute("x", attrEditOp.getValue());
                } else
if(attrEditOp.getAttribute().equals("attri
buteY")) {
                    ((Element)
points.get(attrEditOp.getPosition())).set
Attribute("y", attrEditOp.getValue());
                }
            }
        }
        // </editor-fold>
    } else if(op instanceof
CordieMoveOperation) { // <editor-
fold>
        CordieMoveOperation cmo =
(CordieMoveOperation) op;
        Element target = (Element)
doChildren.remove(cmo.getPosition());

doChildren.add(cmo.getDestination(),
target);
        // </editor-fold>
    } else if(op instanceof
CordieRemoveCollaboratorOperation) {
// <editor-fold>

CordieRemoveCollaboratorOperation
crco =
(CordieRemoveCollaboratorOperation)
op;
        String userToRemove =
crco.getUsername();

        synchronized(currentEditors) {
            Set<String> editorsToRemove =
new HashSet<String>();
            for(String key :
currentEditors.keySet()) {

if(currentEditors.get(key).getUsername(
).equals(userToRemove)) {

//currentEditors.remove(key);

editorsToRemove.add(key);
                }
            }
            for(String key :
editorsToRemove) {

//System.out.println("removing user: " +
key + " for " + userToRemove);
                currentEditors.remove(key);
            }


if(currentUsers.contains(userToRemove)
) {

currentUsers.remove(currentUsers.inde
xOf(userToRemove));
            }


//if(!collaboratorList.contains(userToRe
move)) return;


collaboratorList.remove(userToRemove)
;

collaboratorInfo.remove(userToRemove
);

//currentEditors.remove(userToRemove
);
        }
        // </editor-fold>
    } else if(op instanceof
CordieRemoveUserOperation) { //
<editor-fold>
        CordieRemoveUserOperation
cruo = (CordieRemoveUserOperation)
op;
        String userToRemove =
cruo.getUsername();

        synchronized(currentEditors) {
```

```
            for(String key :
currentEditors.keySet()) {

if(currentEditors.get(key).getUsername(
).equals(userToRemove)) {

currentEditors.remove(key);
                }
            }

if(currentUsers.contains(userToRemove)
) {

currentUsers.remove(currentUsers.inde
xOf(userToRemove));
            }


//System.out.println(cruo.getUsername(
) + " has not sent messages in a long
time. "
            //      + "He/She will now be
removed from current users.");

//currentEditors.remove(userToRemove
);
        }
        // </editor-fold>
    } else if(op instanceof
CordieAddCollaboratorOperation) { //
<editor-fold>
        CordieAddCollaboratorOperation
caco =
(CordieAddCollaboratorOperation) op;

collaboratorList.add(caco.getUsername(
));
        String[] temp = {
caco.getFirstname(),
caco.getLastname(), caco.getEmail(),
                    caco.getDisplaypic() };

collaboratorInfo.put(caco.getUsername(
), temp);
        // </editor-fold>
    }

    // Broadcast the operation just
applied to all other clients
    for(String editorID :
currentEditors.keySet()) {
        if(!editorID.equals(sender))

currentEditors.get(editorID).send(op,
sender);
    }
  }

  @Override
  public String toString() {
    String temp = "{";
    temp = temp + "\"diagramid\" : \""
+ diagramID + "\", \"title\" : \""
            + diagramTitle + "\",
\"creator\" : \"" +
StringEscapeUtils.escapeJava(creator) +
            "\", \"datecreated\" : \"" +
dateCreated + "\", \"collaborators\" : [";

    ListIterator itr;
    itr = collaboratorList.listIterator();
    while(itr.hasNext()) {
      if(itr.hasPrevious()) temp += ", ";

      String currCol = (String) itr.next();
      String[] currColInfo =
collaboratorInfo.get(currCol);
      temp += "{\"username\" : \"" +
StringEscapeUtils.escapeJava(currCol) +
"\", \"firstname\" : \""
              +
StringEscapeUtils.escapeJava(currColInf
o[0]) + "\", \"lastname\" : \"" +
StringEscapeUtils.escapeJava(currColInf
o[1])
              + "\", \"email\" : \"" +
StringEscapeUtils.escapeJava(currColInf
o[2]) + "\", \"displaypic\" : \""
              +
StringEscapeUtils.escapeJava(currColInf
o[3]) + "\"}";
    }

    temp += "], \"currentusers\" : [";
    boolean first = true;
    for(String currentuser :
currentUsers) {
      if(!first) temp = temp + ", ";
      temp = temp + "\"" +
StringEscapeUtils.escapeJava(currentus
er) + "\"";
      first = false;
    }

    temp += "], \"objects\" : [";
    itr = root.getChildren().listIterator();
    while(itr.hasNext()) {
      if(itr.hasPrevious()) temp += ", ";
      temp +=
CordieObjectConverter.convert((Elemen
t) itr.next());
    }

    temp += "] }";
    return temp;
  }

  private void printToFile(){
    try {
      //output to a file
      FileWriter writer = new
FileWriter("cordie_diagram_files/" +
diagramID + ".xml");
      outputter.output(dom, writer);
      writer.close();

    } catch(java.io.IOException e) {
      e.printStackTrace();
    }
  }

  public class CordieClient {
    private String username;
    private int myMsgs;
    private int otherMsgs;
    private Date lastUpdate;
    private ArrayList<CordieMessage>
outgoing;
    private ArrayList<String> updates;
    //private CometContext
cometContext;
    private Timer lastEditTimer;
    private final long EDIT_TIMEOUT =
30 * 1000;
    private EditTimeoutReached
timeoutTask;

    public CordieClient(String
iUsername) {
      this.username = iUsername;
      this.myMsgs = 0;
      this.otherMsgs = 0;
      this.lastUpdate = new Date();
      this.outgoing = new
ArrayList<CordieMessage>();
      this.updates = new
ArrayList<String>();

      // Set up a TimerTask object to
monitor the latest edit received from
the server
      this.lastEditTimer = new Timer();
      this.timeoutTask = new
EditTimeoutReached();

this.lastEditTimer.schedule(timeoutTask
, EDIT_TIMEOUT);
    }

    public void
processMessage(CordieMessage msg) {
      // Discard acknowledged
messages
      for(int i = 0; i < outgoing.size();
i++) {
        if(outgoing.get(i).getMyMsgs()
< msg.getOtherMsgs()) {
          outgoing.remove(i);
          i--;
        }
      }

      for(int i = 0; i < outgoing.size();
i++) {
        CordieOperation[] origOps =
{msg.getOp(), outgoing.get(i).getOp()};
        CordieOperation[]
transformedOps =
Transform.doTransform(origOps);

msg.setOp(transformedOps[0]);

outgoing.get(i).setOp(transformedOps[1
]);
      }

      apply(msg.getOp(),
msg.getSender());
```

```
        this.lastUpdate = new Date();

        otherMsgs++;
    }

    public void send(CordieOperation
op, String sender) {
        if(op instanceof
CordieInsertOperation) {
            outgoing.add(new
CordieMessage("server",
                new
CordieInsertOperation((CordieInsertOpe
ration)op),
                myMsgs, otherMsgs));
        } else if(op instanceof
CordieDeleteOperation) {
            outgoing.add(new
CordieMessage("server",
                new
CordieDeleteOperation((CordieDeleteO
peration)op),
                myMsgs, otherMsgs));
        } else if(op instanceof
CordieEditOperation) {
            outgoing.add(new
CordieMessage("server",
                new
CordieEditOperation((CordieEditOperati
on)op),
                myMsgs, otherMsgs));
        } else if(op instanceof
CordieMoveOperation) {
            outgoing.add(new
CordieMessage("server",
                new
CordieMoveOperation((CordieMoveOpe
ration)op),
                myMsgs, otherMsgs));
        } else if(op instanceof
CordieRemoveCollaboratorOperation) {
            outgoing.add(new
CordieMessage("server",
                new
CordieRemoveCollaboratorOperation((C
ordieRemoveCollaboratorOperation)op)
,
                myMsgs, otherMsgs));
        } else {
            outgoing.add(new
CordieMessage("server",
                new
CordieNoneOperation(), myMsgs,
otherMsgs));
        }

        updates.add("{ \"op\" : " +
op.toString() + ", \"otherMsgs\" : "
            + otherMsgs + ", \"from\" :
\"" +
            //(sender != null ?
StringEscapeUtils.escapeJava(sender) :
"")
            (sender != null ?
StringEscapeUtils.escapeJava(currentEdi
tors.get(sender).getUsername()) : "")
            + "\" }");
        myMsgs++;

        timeoutTask.cancel();
        this.timeoutTask = new
EditTimeoutReached();

this.lastEditTimer.schedule(timeoutTask
, EDIT_TIMEOUT);
    }

    public boolean hasUpdates() {
        return !updates.isEmpty();
    }

    public String getUpdates() {
        String temp = "[";

        boolean first = true;
        for(String update : updates) {
            if(!first)
                temp += ", ";
            else
                first = false;

            temp += update;
        }

        temp += "]";

        updates.clear();

        return temp;
    }

    public Date getLastUpdate() {
        return lastUpdate;
    }

    public String getUsername() {
        return username;
    }

    public String toString() {
        return "username: " + username
+ " outgoing: " + outgoing +
            " updates: " + updates;
    }

    // this is only run when no
messages are sent by the server
    // for EDIT_TIMEOUT milliseconds
    private class EditTimeoutReached
extends TimerTask {
        @Override
        public void run() {
            if(hasUpdates()) {
                timeoutTask = new
EditTimeoutReached();

lastEditTimer.schedule(timeoutTask,
EDIT_TIMEOUT);
            } else {
                send(new
CordieNoneOperation(), null);
            }
        }
    }
}

// Thread that processes the
messages in the message queue
public class CordieMessageDirector
implements Runnable {
    public CordieMessageDirector() {
    }

    @Override
    public void run() {
        while(true) {
            try {
                CordieMessage msg =
messageQueue.take();
                if(msg.getSender() != null) {

currentEditors.get(msg.getSender()).pro
cessMessage(msg);
                } else {
                    apply(msg.getOp(),
msg.getSender());
                }

                if(messageQueueEmpty() &&
(msg.getOp() instanceof
CordieNoneOperation)) {
                    printToFile();
                }
            } catch(Exception exception) {
                exception.printStackTrace();
            }
        }
    }
}

// Periodically checks which editors
are no longer active
private class MonitorEditors extends
TimerTask {
    public void run() {
        long timeNow = (new
Date()).getTime();
        /*synchronized(currentEditors) {
            for(String editorID :
currentEditors.keySet()) {
                long idleDuration = timeNow
-
currentEditors.get(editorID).getLastUpd
ate().getTime();


System.out.println(currentEditors.get(e
ditorID).getUsername() + " idle for " +
idleDuration);

                if(idleDuration >
MAX_IDLE_ALLOWED) {
                    CordieOperation co = new
CordieRemoveUserOperation(currentEd
itors.get(editorID).getUsername());
```

```
                queueMessage(new
CordieMessage(null, co, 0, 0));
              }
            }
          }*/
          synchronized(currentEditors) {
            HashMap<String, Boolean>
removeList = new HashMap<String,
Boolean>();
            for(String collaborator :
currentUsers) {
              removeList.put(collaborator,
true);
            }

            for(String editorID :
currentEditors.keySet()) {
              long idleDuration = timeNow
-
currentEditors.get(editorID).getLastUpd
ate().getTime();

              if(idleDuration <=
MAX_IDLE_ALLOWED) {

removeList.put(currentEditors.get(edito
rID).getUsername(), false);
                //CordieOperation co =
new
CordieRemoveUserOperation(currentEd
itors.get(editorID).getUsername());
                //queueMessage(new
CordieMessage(null, co, 0, 0));
              }
            }

            for(String collaborator :
currentUsers) {

if(removeList.get(collaborator)) {
                CordieOperation co = new
CordieRemoveUserOperation(collaborat
or);
                queueMessage(new
CordieMessage(null, co, 0, 0));

System.out.println("Removing " +
collaborator);
              }
            }
          }
        }
      }

      public void cancelTimer() {
        timer.cancel();
      }
    }
```

CordieMessage.java

```
package cordie.diagram;

import
cordie.diagram.operation.CordieOperati
on;

public class CordieMessage {
  private String sender;
  private CordieOperation op;
  private int myMsgs;
  private int otherMsgs;

  public CordieMessage(String iSender,
CordieOperation iOp, int iMyMsgs, int
iOtherMsgs) {
    this.sender = iSender;
    this.op = iOp;
    this.myMsgs = iMyMsgs;
    this.otherMsgs = iOtherMsgs;
  }

  public void setOp(CordieOperation
iOp) {
    this.op = iOp;
  }

  public String getSender() {
    return sender;
  }

  public CordieOperation getOp() {
    return op;
  }

  public int getMyMsgs() {
    return myMsgs;
  }

  public int getOtherMsgs() {
    return otherMsgs;
  }

  public String toString() {
    return "sender: " + sender + " op: "
+ op + " myMsgs: " + myMsgs
        + " otherMsgs: " + otherMsgs;
  }
}
```

CordieObjectConverter.java

```
package cordie.diagram;

import java.util.ListIterator;
import org.jdom.Element;
import
org.apache.commons.lang3.StringEscap
eUtils;

public class CordieObjectConverter {
  public static String convert(Element
elt) {
    String eltName = elt.getName();
    if(eltName.equals("rectangle") ||
eltName.equals("ellipse") ||
eltName.equals("lifeline")
        ||
eltName.equals("arrowhead") ||
eltName.equals("activationbar")
        || eltName.equals("deletion")
|| eltName.equals("initialps")
        || eltName.equals("finalstate")
|| eltName.equals("shallowhistps")
        ||
eltName.equals("deephistps") ||
eltName.equals("initialnode")
        || eltName.equals("blackbar")
|| eltName.equals("diamond")
        || eltName.equals("finalnode")
|| eltName.equals("pin")
        || eltName.equals("flowfinal")
|| eltName.equals("port")) { // <editor-
fold>
      return "{\"objecttype\" : \"" +
eltName + "\", \"linewidth\" : " +
elt.getAttributeValue("linewidth")
        + ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") +
        "\", \"linejoin\" : \"" +
elt.getAttributeValue("linejoin") +
        "\", \"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle")
        + "\", \"x\" : " +
elt.getAttributeValue("x") + ", \"y\" : " +
        elt.getAttributeValue("y") + ",
\"width\" : " +
elt.getAttributeValue("width")
        + ", \"height\" : " +
elt.getAttributeValue("height") + ",
\"fillcolor\" : \""
        +
elt.getAttributeValue("fillcolor") + "\",
\"rotate\" : " +
        elt.getAttributeValue("rotate")
+ " }";
      // </editor-fold>
    } else if(eltName.equals("line")) { //
<editor-fold>
      return "{\"objecttype\" : \"line\",
\"linewidth\" : " +
elt.getAttributeValue("linewidth") +
        ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") + "\",
\"linejoin\" : \"" +

elt.getAttributeValue("linejoin") + "\",
\"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle") +
        "\", \"x1\" : " +
elt.getAttributeValue("x1") +
        ", \"y1\" : " +
elt.getAttributeValue("y1") +
        ", \"x2\" : " +
elt.getAttributeValue("x2") +
        ", \"y2\" : " +
elt.getAttributeValue("y2") +
        ", \"arrowstyle1\" : \"" +
elt.getAttributeValue("arrowstyle1") +
        "\", \"arrowstyle2\" : \"" +
elt.getAttributeValue("arrowstyle2") +
        "\", \"linestyle\" : \"" +
elt.getAttributeValue("linestyle") + "\"
}";
      // </editor-fold>
    } else if(eltName.equals("path")) {
// <editor-fold>
```

```
        String temp = "{\"objecttype\" :
\"path\", \"linewidth\" : " +
elt.getAttributeValue("linewidth")
                + ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") +
                "\", \"linejoin\" : \"" +
elt.getAttributeValue("linejoin")
                + "\", \"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle")
                + "\", \"points\" : [";

        String[] temp2 =
elt.getAttributeValue("points").split("[
]+");
        for(int i = 0; i < temp2.length; i++)
{
            String[] temp3 =
temp2[i].split("[,]+");
            if(i != 0) temp += ",";
            temp = temp + " {\"x\" : " +
temp3[0] + ", \"y\" : " +
                    temp3[1] + "}";
        }
        temp += " ] }";

        return temp;
        // </editor-fold>
    } else if(eltName.equals("polygon"))
{ // <editor-fold>
        String temp = "{\"objecttype\" :
\"polygon\", \"linewidth\" : " +
elt.getAttributeValue("linewidth")
                + ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") +
                "\", \"linejoin\" : \"" +
elt.getAttributeValue("linejoin")
                + "\", \"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle")
                + "\", \"fillcolor\" : \"" +
elt.getAttributeValue("fillcolor")
                + "\", \"points\" : [";

        ListIterator itr =
elt.getChildren("point").listIterator();
        while(itr.hasNext()) {
            if(itr.hasPrevious()) temp += ",
";
            Element point = (Element)
itr.next();
            temp += "{\"x\" : " +
point.getAttributeValue("x") + ", \"y\" :
"
                    +
point.getAttributeValue("y") + "}";
        }

        temp += " ] }";

        return temp;
        // </editor-fold>
    } else if(eltName.equals("text")) { //
<editor-fold>
        return "{\"objecttype\" : \"text\",
\"fontstyle\" : \"" +
elt.getAttributeValue("fontstyle")
                + "\", \"fontweight\" : \"" +
elt.getAttributeValue("fontweight")
                + "\", \"fontsize\" : " +
elt.getAttributeValue("fontsize")
                + ", \"fontfamily\" : \"" +
elt.getAttributeValue("fontfamily")
                + "\", \"x\" : " +
elt.getAttributeValue("x")
                + ", \"y\" : " +
elt.getAttributeValue("y")
                + ", \"width\" : " +
elt.getAttributeValue("width")
                + ", \"height\" : " +
elt.getAttributeValue("height")
                + ", \"label\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("label"))
                + "\", \"textcolor\" : \"" +
elt.getAttributeValue("textcolor")
                + "\", \"rotate\" : " +
elt.getAttributeValue("rotate") + " }";
        // </editor-fold>
    } else if(eltName.equals("class")) {
// <editor-fold>
        String temp = "{\"objecttype\" :
\"class\", \"linewidth\" : " +
elt.getAttributeValue("linewidth") +
                ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") + "\",
\"linejoin\" : \"" +

elt.getAttributeValue("linejoin") + "\",
\"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle") +
                "\", \"x\" : " +
elt.getAttributeValue("x") + ", \"y\" : " +
elt.getAttributeValue("y") +
                ", \"width\" : " +
elt.getAttributeValue("width") + ",
\"height\" : " +
                elt.getAttributeValue("height")
+ ", \"fillcolor\" : \"" +
elt.getAttributeValue("fillcolor") +
                "\", \"textcolor\" : \"" +
elt.getAttributeValue("textcolor") +
                "\", \"fontsize\" : " +
elt.getAttributeValue("fontsize") +
                ", \"classfontsize\" : " +
elt.getAttributeValue("classfontsize") +
                ", \"classname\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("classname")) +
                "\", \"abstractclass\" : " +
elt.getAttributeValue("abstractclass") +
                ", \"qualifiedassociation\" : " +
elt.getAttributeValue("qualifiedassociati
on") +
                ", \"qualifier\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("qualifier")) +
                "\", \"stereotype\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("stereotype")) +
                "\", \"templateclass\" : " +
elt.getAttributeValue("templateclass") +
                ", \"activeclass\" : " +
elt.getAttributeValue("activeclass") + ",
\"attributes\" : [";

        ListIterator itr;

        Element attributes =
elt.getChild("attributes");
        if(attributes != null) {
            itr =
attributes.getChildren("attribute").listIt
erator();
            while(itr.hasNext()) {
                if(itr.hasPrevious()) temp +=
", ";
                temp += convert((Element)
itr.next());
            }
        }
        temp = temp + "],
\"showattributes\" : " +
elt.getAttributeValue("showattributes")
                + ", \"operations\" : [";

        Element operations =
elt.getChild("operations");
        if(operations != null) {
            itr =
operations.getChildren("operation").listI
terator();
            while(itr.hasNext()) {
                if(itr.hasPrevious()) temp +=
", ";
                temp += convert((Element)
itr.next());
            }
        }
        temp = temp + "],
\"showoperations\" : " +
elt.getAttributeValue("showoperations"
)
                + ", \"templates\" : [";

        Element templates =
elt.getChild("templates");
        if(templates != null) {
            itr =
templates.getChildren("template").listIt
erator();
            while(itr.hasNext()) {
                if(itr.hasPrevious()) temp +=
", ";
                temp += convert((Element)
itr.next());
            }
        }
        temp += "], \"rotate\" : " +
elt.getAttributeValue("rotate") + " }";

        return temp;
        // </editor-fold>
    } else if(eltName.equals("note") ||
eltName.equals("frame")
            || eltName.equals("package1")
|| eltName.equals("component")
```

```
            || eltName.equals("artifact")
|| eltName.equals("usecase")
            || eltName.equals("actor") ||
eltName.equals("superstate")
            || eltName.equals("action") ||
eltName.equals("subactivity")
            ||
eltName.equals("timesignal") ||
eltName.equals("acceptsignal")
            ||
eltName.equals("sendsignal") ||
eltName.equals("connector")
            ||
eltName.equals("transformation") ||
eltName.equals("part")
            ||
eltName.equals("package2")) { //
<editor-fold>
        return "{\"objecttype\" : \"" +
eltName + "\", \"fillcolor\" : \"" +
elt.getAttributeValue("fillcolor")
                + "\", \"fontsize\" : " +
elt.getAttributeValue("fontsize")
                + ", \"fontfamily\" : \"" +
elt.getAttributeValue("fontfamily")
                + "\", \"fontstyle\" : \"" +
elt.getAttributeValue("fontstyle")
                + "\", \"fontweight\" : \"" +
elt.getAttributeValue("fontweight")
                + "\", \"height\" : " +
elt.getAttributeValue("height")
                + ", \"label\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("label"))
                + "\", \"linecap\" : \"" +
elt.getAttributeValue("linecap")
                + "\", \"linejoin\" : \"" +
elt.getAttributeValue("linejoin")
                + "\", \"linewidth\" : " +
elt.getAttributeValue("linewidth")
                + ", \"rotate\" : " +
elt.getAttributeValue("rotate")
                + ", \"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle")
                + "\", \"textcolor\" : \"" +
elt.getAttributeValue("textcolor")
                + "\", \"width\" : " +
elt.getAttributeValue("width")
                + ", \"x\" : " +
elt.getAttributeValue("x")
                + ", \"y\" : " +
elt.getAttributeValue("y")  + "}";
        // </editor-fold>
    } else
if(eltName.equals("attribute")) { //
<editor-fold>
        return "{\"value\" : \"" +
StringEscapeUtils.escapeJava(elt.getText
()) + "\", \"isstatic\" : " +

elt.getAttributeValue("static") + "}";
        // </editor-fold>
    } else
if(eltName.equals("operation")) { //
<editor-fold>
        return "{\"value\" : \"" +
StringEscapeUtils.escapeJava(elt.getText
()) + "\", \"isstatic\" : " +

elt.getAttributeValue("static") + ",
\"isabstract\" : " +

elt.getAttributeValue("abstract") + "}";
        // </editor-fold>
    } else
if(eltName.equals("template")) { //
<editor-fold>
        return "{\"value\" : \"" +
StringEscapeUtils.escapeJava(elt.getText
()) + "\"}";
        // </editor-fold>
    } else if(eltName.equals("bezier")) {
// <editor-fold>
        return "{\"objecttype\" :
\"bezier\", \"linewidth\" : " +
elt.getAttributeValue("linewidth") +
            ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") + "\",
\"linejoin\" : \"" +

elt.getAttributeValue("linejoin") + "\",
\"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle") +
            "\", \"x1\" : " +
elt.getAttributeValue("x1") +
            ", \"y1\" : " +
elt.getAttributeValue("y1") +
            ", \"x2\" : " +
elt.getAttributeValue("x2") +
            ", \"y2\" : " +
elt.getAttributeValue("y2") +
            ", \"arrowstyle1\" : \"" +
elt.getAttributeValue("arrowstyle1") +
            "\", \"arrowstyle2\" : \"" +
elt.getAttributeValue("arrowstyle2") +
            "\", \"linestyle\" : \"" +
elt.getAttributeValue("linestyle") +
            "\", \"ctrl1x\" : " +
elt.getAttributeValue("ctrl1x")+ ",
\"ctrl1y\" : "
            +
elt.getAttributeValue("ctrl1y")+ ",
\"ctrl2x\" : " +
elt.getAttributeValue("ctrl2x")
            + ", \"ctrl2y\" : " +
elt.getAttributeValue("ctrl2y")+ " }";
        // </editor-fold>
    } else if(eltName.equals("point")) {
// <editor-fold>
        return "{\"x\" : " +
elt.getAttributeValue("x") + ",\"y\" : " +
            elt.getAttributeValue("y") +
"}";
        // </editor-fold>
    } else
if(eltName.equals("instance")) { //
<editor-fold>
        String temp = "{\"objecttype\" :
\"instance\", \"linewidth\" : " +
elt.getAttributeValue("linewidth") +
            ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") +
            "\", \"linejoin\" : \"" +
elt.getAttributeValue("linejoin") +
            "\", \"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle") +
            "\", \"x\" : " +
elt.getAttributeValue("x") +
            ", \"y\" : " +
elt.getAttributeValue("y") +
            ", \"width\" : " +
elt.getAttributeValue("width") +
            ", \"height\" : " +
elt.getAttributeValue("height") +
            ", \"fillcolor\" : \"" +
elt.getAttributeValue("fillcolor") +
            "\", \"textcolor\" : \"" +
elt.getAttributeValue("textcolor") +
            "\", \"fontsize\" : " +
elt.getAttributeValue("fontsize") +
            ", \"classname\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("classname")) +
            "\", \"instancename\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("instancename")) +
            "\", \"rotate\" : " +
elt.getAttributeValue("rotate") +
            ", \"showattributes\" : " +
elt.getAttributeValue("showattributes")
+
            ", \"attributes\" : [";

        Element attributes =
elt.getChild("attributes");
        if(attributes != null) {
            ListIterator itr =
attributes.getChildren("attribute").listIt
erator();
            while(itr.hasNext()) {
                if(itr.hasPrevious()) temp +=
", ";
                temp += convert((Element)
itr.next());
            }
        }
        temp += "] }";

        return temp;
        // </editor-fold>
    } else if(eltName.equals("node")) {
// <editor-fold>
        String temp = "{\"objecttype\" :
\"node\", \"linewidth\" : " +
elt.getAttributeValue("linewidth") +
            ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") +
            "\", \"linejoin\" : \"" +
elt.getAttributeValue("linejoin") +
            "\", \"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle") +
            "\", \"x\" : " +
elt.getAttributeValue("x") +
            ", \"y\" : " +
elt.getAttributeValue("y") +
```

```
            ", \"width\" : " +
elt.getAttributeValue("width") +
            ", \"height\" : " +
elt.getAttributeValue("height") +
            ", \"topheight\" : " +
elt.getAttributeValue("topheight") +
            ", \"fillcolor\" : \"" +
elt.getAttributeValue("fillcolor") +
            "\", \"textcolor\" : \"" +
elt.getAttributeValue("textcolor") +
            "\", \"fontsize\" : " +
elt.getAttributeValue("fontsize") +
            ", \"nodename\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("nodename")) +
            "\", \"stereotype\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("stereotype")) +
            "\", \"rotate\" : " +
elt.getAttributeValue("rotate") +
            ", \"showartifacts\" : " +
elt.getAttributeValue("showartifacts") +
            ", \"containedartifacts\" : [";

        Element artifacts =
elt.getChild("containedartifacts");
        if(artifacts != null) {
           ListIterator itr =
artifacts.getChildren("containedartifact"
).listIterator();
           while(itr.hasNext()) {
              if(itr.hasPrevious()) temp +=
", ";
              temp += convert((Element)
itr.next());
           }
        }
        temp += "], \"taggedvalues\" : [";

        Element taggedvalues =
elt.getChild("taggedvalues");
        if(taggedvalues != null) {
           ListIterator itr =
taggedvalues.getChildren("taggedvalue"
).listIterator();
           while(itr.hasNext()) {
              if(itr.hasPrevious()) temp +=
", ";
              temp += convert((Element)
itr.next());
           }
        }
        temp += "] }";

        return temp;
        // </editor-fold>
     } else
if(eltName.equals("taggedvalue")) { //
<editor-fold>
        return "\"" +
StringEscapeUtils.escapeJava(elt.getText
()) + "\"";
        // </editor-fold>
     } else
if(eltName.equals("containedartifact")) {
// <editor-fold>
        return "\"" +
StringEscapeUtils.escapeJava(elt.getText
()) + "\"";
        // </editor-fold>
     } else
if(eltName.equals("statebox")) { //
<editor-fold>
        String temp = "{\"objecttype\" :
\"statebox\", \"linewidth\" : " +
elt.getAttributeValue("linewidth") +
            ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") +
            "\", \"linejoin\" : \"" +
elt.getAttributeValue("linejoin") +
            "\", \"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle") +
            "\", \"x\" : " +
elt.getAttributeValue("x") +
            ", \"y\" : " +
elt.getAttributeValue("y") +
            ", \"width\" : " +
elt.getAttributeValue("width") +
            ", \"height\" : " +
elt.getAttributeValue("height") +
            ", \"fillcolor\" : \"" +
elt.getAttributeValue("fillcolor") +
            "\", \"textcolor\" : \"" +
elt.getAttributeValue("textcolor") +
            "\", \"fontsize\" : " +
elt.getAttributeValue("fontsize") +
            ", \"statename\" : \"" +
StringEscapeUtils.escapeJava(elt.getAttr
ibuteValue("statename")) +
            "\", \"rotate\" : " +
elt.getAttributeValue("rotate") +
            ", \"showinternalactivities\" : "
+
elt.getAttributeValue("showinternalacti
vities") +
            ", \"internalactivitys\" : [";

        Element internalactivitys =
elt.getChild("internalactivitys");
        if(internalactivitys != null) {
           ListIterator itr =
internalactivitys.getChildren("internalac
tivity").listIterator();
           while(itr.hasNext()) {
              if(itr.hasPrevious()) temp +=
", ";
              temp += convert((Element)
itr.next());
           }
        }
        temp += "] }";

        return temp;
        // </editor-fold>
     } else
if(eltName.equals("internalactivity")) {
// <editor-fold>
        return "\"" +
StringEscapeUtils.escapeJava(elt.getText
()) + "\"";
        // </editor-fold>
     } else
if(eltName.equals("expansionregion")) {
// <editor-fold>
        return "{\"objecttype\" :
\"expansionregion\", \"linewidth\" : " +

elt.getAttributeValue("linewidth") + ",
\"linecap\" : \"" +

elt.getAttributeValue("linecap") + "\",
\"linejoin\" : \"" +

elt.getAttributeValue("linejoin") + "\",
\"strokestyle\" : \"" +

elt.getAttributeValue("strokestyle") +
"\", \"x\" : " +
           elt.getAttributeValue("x") + ",
\"y\" : " + elt.getAttributeValue("y")
           + ", \"width\" : " +
elt.getAttributeValue("width") + ",
\"height\" : "
           +
elt.getAttributeValue("height") + ",
\"fillcolor\" : \"" +

elt.getAttributeValue("fillcolor") + "\",
\"rotate\" : " +
           elt.getAttributeValue("rotate")
+ ", \"listboxpin1x\" : " +

elt.getAttributeValue("listboxpin1x") +
", \"listboxpin1y\" : "
           +
elt.getAttributeValue("listboxpin1y")  +
", \"listboxpin2x\" : "
           +
elt.getAttributeValue("listboxpin2x") +
", \"listboxpin2y\" : "
           +
elt.getAttributeValue("listboxpin2y")+ ",
\"listboxpinsize\" : "
           +
elt.getAttributeValue("listboxpinsize") +
" }";
        // </editor-fold>
     } else if(eltName.equals("polyline"))
{ // <editor-fold>
        String temp = "{\"objecttype\" :
\"polyline\", \"linewidth\" : " +
elt.getAttributeValue("linewidth")
            + ", \"linecap\" : \"" +
elt.getAttributeValue("linecap") +
            "\", \"linejoin\" : \"" +
elt.getAttributeValue("linejoin")
            + "\", \"strokestyle\" : \"" +
elt.getAttributeValue("strokestyle")
            + "\", \"arrowstyle1\" : \"" +
elt.getAttributeValue("arrowstyle1")
            + "\", \"arrowstyle2\" : \"" +
elt.getAttributeValue("arrowstyle2")
            + "\", \"linestyle\" : \"" +
elt.getAttributeValue("linestyle")
            + "\", \"points\" : [";
```

```
        ListIterator itr =
elt.getChildren("point").listIterator();
        while(itr.hasNext()) {
          if(itr.hasPrevious()) temp += ",
";
          Element point = (Element)
itr.next();
          temp += "{\"x\" : " +
point.getAttributeValue("x") + ", \"y\" :
"
                +
point.getAttributeValue("y") + "}";
        }

        temp += " ] }";

        return temp;
        // </editor-fold>
      } else {
        return "{}";
      }
    }
}
```

Transform.java

```
package cordie.diagram;

import cordie.diagram.operation.*;

public class Transform {
   public static CordieOperation[]
doTransform(CordieOperation[] tOps) {
      if(tOps[0] instanceof
CordieInsertOperation) { // <editor-
fold>
        CordieInsertOperation a =
(CordieInsertOperation) tOps[0];
        if(tOps[1] instanceof
CordieInsertOperation) {
          CordieInsertOperation b =
(CordieInsertOperation) tOps[1];
          if(a.getPosition() >
b.getPosition())
            a.incrementPosition();
          else
            b.incrementPosition();
        } else if(tOps[1] instanceof
CordieDeleteOperation) {
          CordieDeleteOperation b =
(CordieDeleteOperation) tOps[1];
          if(a.getPosition() >
b.getPosition())
            a.decrementPosition();
          else
            b.incrementPosition();
        } else if(tOps[1] instanceof
CordieEditOperation) {
          CordieEditOperation b =
(CordieEditOperation) tOps[1];
          if(a.getPosition() <=
b.getPosition())
            b.incrementPosition();
        } else if(tOps[1] instanceof
CordieMoveOperation) {
          CordieMoveOperation b =
(CordieMoveOperation) tOps[1];
          if(a.getPosition() >
b.getPosition()) {
            if(a.getPosition() >
b.getDestination()) {
              //do nothing
            }
            else {
              a.decrementPosition();
              b.incrementDestination();
            }
          } else {
            if(a.getPosition() >
b.getDestination()) {
              a.incrementPosition();
              b.incrementPosition();
            } else {
              b.incrementPosition();
              b.incrementDestination();
            }
          }
        } // </editor-fold>
      } else if(tOps[0] instanceof
CordieDeleteOperation) { // <editor-
fold>
        CordieDeleteOperation a =
(CordieDeleteOperation) tOps[0];
        if(tOps[1] instanceof
CordieInsertOperation) {
          CordieInsertOperation b =
(CordieInsertOperation) tOps[1];
          if(a.getPosition() <
b.getPosition())
            b.decrementPosition();
          else
            a.incrementPosition();
        } else if(tOps[1] instanceof
CordieDeleteOperation) {
          CordieDeleteOperation b =
(CordieDeleteOperation) tOps[1];
          if(a.getPosition() <
b.getPosition())
            b.decrementPosition();
          else if(a.getPosition() >
b.getPosition())
            a.decrementPosition();
          else {
            tOps[0] = new
CordieNoneOperation();
            tOps[1] = new
CordieNoneOperation();
          }
        } else if(tOps[1] instanceof
CordieEditOperation) {
          CordieEditOperation b =
(CordieEditOperation) tOps[1];
          if(a.getPosition() <
b.getPosition())
            b.decrementPosition();
          else if(a.getPosition() ==
b.getPosition())
            tOps[1] = new
CordieNoneOperation();
        } else if(tOps[1] instanceof
CordieMoveOperation) {
          CordieMoveOperation b =
(CordieMoveOperation) tOps[1];
          if(a.getPosition() ==
b.getPosition()) {

a.setPosition(b.getDestination());
            tOps[1] = new
CordieNoneOperation();
          } else if(a.getPosition() <
b.getPosition()) {
            if(a.getPosition() <
b.getDestination()) {
              b.decrementPosition();
              b.decrementDestination();
            } else {
              a.incrementPosition();
              b.decrementPosition();
            }
          } else if(a.getPosition() <=
b.getDestination()) {
            a.decrementPosition();
            b.decrementDestination();
          }
        } // </editor-fold>
      } else if(tOps[0] instanceof
CordieEditOperation) { // <editor-fold>
        CordieEditOperation a =
(CordieEditOperation) tOps[0];
        if(tOps[1] instanceof
CordieInsertOperation) {
          CordieInsertOperation b =
(CordieInsertOperation) tOps[1];
          if(a.getPosition() >=
b.getPosition())
            a.incrementPosition();
        } else if(tOps[1] instanceof
CordieDeleteOperation) {
          CordieDeleteOperation b =
(CordieDeleteOperation) tOps[1];
          if(a.getPosition() >
b.getPosition())
            a.decrementPosition();
          else if(a.getPosition() ==
b.getPosition())
            tOps[0] = new
CordieNoneOperation();
        } else if(tOps[1] instanceof
CordieEditOperation) {
          CordieEditOperation b =
(CordieEditOperation) tOps[1];
          if((a.getPosition() ==
b.getPosition()) &&

a.getAttribute().equals(b.getAttribute())
) {
            if(a.getValue() == null &&
b.getValue() == null) {
              CordieOperation[]
origAttOps = {a.getAttributeOp(),
b.getAttributeOp()};
              CordieOperation[]
transAttOps =
Transform.doTransform(origAttOps);

a.setAttributeOp(transAttOps[0]);
```

```
b.setAttributeOp(transAttOps[1]);
            } else if(a.getValue() != null
&& b.getValue() != null) {
                tOps[1] = new
CordieNoneOperation();
            }
          }
        } else if(tOps[1] instanceof
CordieMoveOperation) {
            CordieMoveOperation b =
(CordieMoveOperation) tOps[1];
            if(a.getPosition() ==
b.getPosition()) {

a.setPosition(b.getDestination());
            } else if(a.getPosition() >=
b.getDestination() &&
                  a.getPosition() <
b.getPosition()) {
              a.incrementPosition();
            } else if(a.getPosition() <=
b.getDestination() &&
                  a.getPosition() >
b.getPosition()) {
              a.decrementPosition();
            } else {
              //do nothing
            }
        } // </editor-fold>
      } else if(tOps[0] instanceof
CordieMoveOperation) { // <editor-
fold>
        CordieMoveOperation a =
(CordieMoveOperation) tOps[0];
        if(tOps[1] instanceof
CordieInsertOperation) {
            CordieInsertOperation b =
(CordieInsertOperation) tOps[1];
            if(b.getPosition() >
a.getPosition()) {
              if(b.getPosition() >
a.getDestination()) {
                //do nothing
              } else {
                a.incrementDestination();
                b.decrementPosition();
              }
            } else if(b.getPosition() >
a.getDestination()) {
              a.incrementPosition();
              b.incrementPosition();
            } else {
              a.incrementPosition();
              a.incrementDestination();
            }
        } else if(tOps[1] instanceof
CordieDeleteOperation) {
            CordieDeleteOperation b =
(CordieDeleteOperation) tOps[1];
            if(b.getPosition() ==
a.getPosition()) {
              tOps[0] = new
CordieNoneOperation();

b.setPosition(a.getDestination());
            } else if(b.getPosition() <
a.getPosition()) {
              if(b.getPosition() <
a.getDestination()) {
                a.decrementPosition();
                a.decrementDestination();
              } else {
                a.decrementPosition();
                b.incrementPosition();
              }
            } else if(b.getPosition() <=
a.getDestination()) {
              a.decrementDestination();
              b.decrementPosition();
            } else {
              //do nothing
            }
        } else if(tOps[1] instanceof
CordieEditOperation) {
            CordieEditOperation b =
(CordieEditOperation) tOps[1];
            if(b.getPosition() ==
a.getPosition()) {

b.setPosition(a.getDestination());
            } else if(b.getPosition() >=
a.getDestination() &&
                  b.getPosition() <
a.getPosition()) {
              b.incrementPosition();
            } else if(b.getPosition() <=
a.getDestination() &&
                  b.getPosition() >
a.getPosition()) {
              b.decrementPosition();
            } else {
              //do nothing
            }
        } else if(tOps[1] instanceof
CordieMoveOperation) {
            CordieMoveOperation b =
(CordieMoveOperation) tOps[1];
            int p = a.getPosition();
            int q = a.getDestination();
            int r = b.getPosition();
            int s = b.getDestination();

            // 1
                        if(p == r) {
              if(q == s) {
                tOps[0] = new
CordieNoneOperation();
                tOps[1] = new
CordieNoneOperation();
                return tOps;
              } else if(r == s) {
                b.setPosition(q);
                b.setDestination(q);
                return tOps;
              } else if(p == q) {
                a.setPosition(s);
                a.setDestination(s);
                return tOps;
              } else {
                a.setPosition(s);
                tOps[1] = new
CordieNoneOperation();
                return tOps;
              }
                        }

            // 2
                        if(p > r) {
              if(p <= s)
a.decrementPosition();
              if(r >= q)
b.incrementPosition();
                        } else {
              if(p >= s)
a.incrementPosition();
              if(r <= q)
b.decrementPosition();
                        }

            // 3
                        if(q > r && q < s)
a.decrementDestination();
                        else if(q < r && q
> s) a.incrementDestination();
                        else {
              if(r == q) {
                if(p < r && r < s)
a.decrementDestination();
                else if(p > r && r > s)
a.incrementDestination();
              } else if(s == q) {
                if(r < s && s <= p)
a.decrementDestination();
                else if(r > s && s >= p)
a.incrementDestination();
              }
                        }

            //4
                        if(s > p) {
              if(s <= q)
b.decrementDestination();
                        } else if(s < p) {
              if(s >= q)
b.incrementDestination();
                        } else { // p == s
              if(p < q && r < s)
b.decrementDestination();
              else if(p > q && r > s)
b.incrementDestination();
                        }
        } // </editor-fold>
      }

      return tOps;
    }
}
```

CordieAddCollaboratorOperation.java

```
package cordie.diagram.operation;

import
org.apache.commons.lang3.StringEscap
eUtils;
```

```
public class
CordieAddCollaboratorOperation
implements CordieOperation {
   private String username;
   private String firstname;
   private String lastname;
   private String email;
   private String displaypic;

   public
CordieAddCollaboratorOperation(String
iUsername, String iFirstname,
         String iLastname, String iEmail,
String iDisplaypic) {
      this.username = iUsername;
      this.firstname = iFirstname;
      this.lastname = iLastname;
      this.email = iEmail;
      this.displaypic = iDisplaypic;
   }

   public
CordieAddCollaboratorOperation(Cordie
AddCollaboratorOperation co) {
      this.username = co.getUsername();
      this.firstname = co.getFirstname();
      this.lastname = co.getLastname();
      this.email = co.getEmail();
      this.displaypic = co.getDisplaypic();
   }

   public String getUsername() {
      return username;
   }

   public String getFirstname() {
      return firstname;
   }

   public String getLastname() {
      return lastname;
   }

   public String getEmail() {
      return email;
   }

   public String getDisplaypic() {
      return displaypic;
   }

   @Override
   public String toString() {
      return "{ \"optype\" :
\"addcollaborator\", \"collaborator\" : {
\"username\" : \"" +

StringEscapeUtils.escapeJava(username)
+ "\", \"firstname\" : \"" +
StringEscapeUtils.escapeJava(firstname)
+
            "\", \"lastname\" : \"" +
StringEscapeUtils.escapeJava(lastname)
+ "\", \"email\" : \"" +

StringEscapeUtils.escapeJava(email) +
"\", \"displaypic\" : \"" +
StringEscapeUtils.escapeJava(displaypic)
+ "\"} }";
   }
}
```

CordieAddUserOperation.java

```
package cordie.diagram.operation;

import
org.apache.commons.lang3.StringEscap
eUtils;

public class CordieAddUserOperation
implements CordieOperation {
   private String user;

   public
CordieAddUserOperation(String u) {
      this.user = u;
   }

   public
CordieAddUserOperation(CordieAddUse
rOperation co) {
      this.user = co.getUser();
   }

   public String getUser() {
      return user;
   }

   @Override
   public String toString() {
      return "{ \"optype\" : \"adduser\",
\"collaborator\" : \""
            +
StringEscapeUtils.escapeJava(user) + "\"
}";
   }
}
```

CordieDeleteOperation.java

```
package cordie.diagram.operation;

public class CordieDeleteOperation
implements CordieOperation {
   private int position;

   public
CordieDeleteOperation(CordieDeleteOp
eration cdo) {
      position = cdo.getPosition();
   }

   public CordieDeleteOperation(int
iPos) {
      position = iPos;
   }

   public void decrementPosition() {
      position--;
   }

   public void setPosition(int iPos) {
      position = iPos;
   }

   public void incrementPosition() {
      position++;
   }

   public int getPosition() {
      return position;
   }

   @Override
   public String toString() {
      return "{ \"optype\" : \"delete\",
\"position\" : " + position + " }";
   }
}
```

CordieEditOperation.java

```
package cordie.diagram.operation;

import java.util.Map;
import
org.apache.commons.lang3.StringEscap
eUtils;

public class CordieEditOperation
implements CordieOperation {
   private int position;
   private String attribute;
   private String value;
   private CordieOperation attributeOp;

   public
CordieEditOperation(CordieEditOperati
on ceo) {
      position = ceo.getPosition();
      attribute = ceo.getAttribute();
      value = ceo.getValue();
      attributeOp = ceo.getAttributeOp();
   }

   public CordieEditOperation(int iPos,
String iAttr, String iVal) {
      position = iPos;
      attribute = iAttr;
      value = iVal;
      attributeOp = new
CordieNoneOperation();
   }

   public
CordieEditOperation(Map<String,
String[]> parameterMap) {
      position =
Integer.parseInt(parameterMap.get("po
sition")[0]);
      attribute =
parameterMap.get("attribute")[0];
      value = null;

      String optype =
parameterMap.get("attrOp")[0];
```

```
      if(optype.equals("insert")) {
         if(attribute.equals("attribute") ||
attribute.equals("operation") ||
            attribute.equals("template") ||
attribute.equals("taggedvalue") ||

attribute.equals("containedartifact") ||
attribute.equals("internalactivity")) {
            attributeOp = new
CordieInsertOperation(attribute,
parameterMap.get("attrVal")[0],

Integer.parseInt(parameterMap.get("att
rPos")[0]));
         } else if(attribute.equals("point"))
{
            attributeOp = new
CordieInsertOperation(attribute,
parameterMap.get("attrX")[0],

parameterMap.get("attrY")[0],
Integer.parseInt(parameterMap.get("att
rPos")[0]));
         }
      } else if(optype.equals("delete")) {
         attributeOp = new
CordieDeleteOperation(Integer.parseInt
(parameterMap.get("attrPos")[0]));
      } else if(optype.equals("edit")) {
         attributeOp = new
CordieEditOperation(Integer.parseInt(p
arameterMap.get("attrPos")[0]),

parameterMap.get("attrAttr")[0],
parameterMap.get("attrVal")[0]);
      } else if(optype.equals("move")) {
         attributeOp = new
CordieMoveOperation(Integer.parseInt(
parameterMap.get("attrPos")[0]),

Integer.parseInt(parameterMap.get("att
rDest")[0]));
      } else {
         attributeOp = new
CordieNoneOperation();
      }
   }

   public void setPosition(int iPos) {
      position = iPos;
   }

   public void
setAttributeOp(CordieOperation op) {
      attributeOp = op;
   }

   public void decrementPosition() {
      position--;
   }

   public void incrementPosition() {
      position++;
   }

   public int getPosition() {
      return position;
   }

   public String getAttribute() {
      return attribute;
   }

   public String getValue() {
      return value;
   }

   public CordieOperation
getAttributeOp() {
      return attributeOp;
   }

   @Override
   public String toString() {
      if(value != null) {
         return "{ \"optype\" : \"edit\",
\"attribute\" : \"" + attribute
            + "\", \"value\" : \"" +
StringEscapeUtils.escapeJava(value) +
"\", \"position\" : "
            + position + " }";
      } else {
         return "{ \"optype\" : \"edit\",
\"attribute\" : \"" + attribute
            + "\", \"attrOp\" : " +
attributeOp + ", \"position\" : "
            + position + " }";
      }
   }
}
```

CordieInsertOperation.java

```
package cordie.diagram.operation;

import java.util.Map;
import org.jdom.Element;
import
cordie.diagram.CordieObjectConverter;

public class CordieInsertOperation
implements CordieOperation {
   private Element obj;
   private int position;

   public
CordieInsertOperation(CordieInsertOper
ation co) {
      position = co.getPosition();
      obj =
(Element)co.getObject().clone();
   }

   public
CordieInsertOperation(Map<String,
String[]> parameterMap) {
      position =
Integer.parseInt(parameterMap.get("po
sition")[0]);
      String  objType =
parameterMap.get("objecttype")[0];
      if(objType.equals("rectangle") ||
objType.equals("ellipse") ||
objType.equals("arrowhead")
            ||
objType.equals("activationbar") ||
objType.equals("lifeline")
            || objType.equals("deletion")
|| objType.equals("initialps")
            || objType.equals("finalstate")
|| objType.equals("shallowhistps")
            ||
objType.equals("deephistps") ||
objType.equals("initialnode")
            || objType.equals("finalnode")
|| objType.equals("blackbar")
            || objType.equals("diamond")
|| objType.equals("pin") ||
objType.equals("flowfinal")
            || objType.equals("port")) {
//<editor-fold>
         obj = new Element(objType);
         obj.setAttribute("x",
parameterMap.get("x")[0]);
         obj.setAttribute("y",
parameterMap.get("y")[0]);
         obj.setAttribute("width",
parameterMap.get("width")[0]);
         obj.setAttribute("height",
parameterMap.get("height")[0]);
         obj.setAttribute("fillcolor",
parameterMap.get("fillcolor")[0]);
         obj.setAttribute("rotate",
parameterMap.get("rotate")[0]);
         obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
         obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
         obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
         obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);
         // </editor-fold>
      } else if(objType.equals("text")) {
//<editor-fold>
         obj = new Element("text");
         obj.setAttribute("fontstyle",
parameterMap.get("fontstyle")[0]);
         obj.setAttribute("fontweight",
parameterMap.get("fontweight")[0]);
         obj.setAttribute("fontsize",
parameterMap.get("fontsize")[0]);
         obj.setAttribute("fontfamily",
parameterMap.get("fontfamily")[0]);
         obj.setAttribute("x",
parameterMap.get("x")[0]);
         obj.setAttribute("y",
parameterMap.get("y")[0]);
         obj.setAttribute("width",
parameterMap.get("width")[0]);
         obj.setAttribute("height",
parameterMap.get("height")[0]);
         obj.setAttribute("label",
parameterMap.get("label")[0]);
         obj.setAttribute("textcolor",
parameterMap.get("textcolor")[0]);
```

```
        obj.setAttribute("rotate",
parameterMap.get("rotate")[0]);
        // </editor-fold>
      } else if(objType.equals("line")) {
//<editor-fold>
        obj = new Element("line");
        obj.setAttribute("x1",
parameterMap.get("x1")[0]);
        obj.setAttribute("y1",
parameterMap.get("y1")[0]);
        obj.setAttribute("x2",
parameterMap.get("x2")[0]);
        obj.setAttribute("y2",
parameterMap.get("y2")[0]);
        obj.setAttribute("arrowstyle1",
parameterMap.get("arrowstyle1")[0]);
        obj.setAttribute("arrowstyle2",
parameterMap.get("arrowstyle2")[0]);
        obj.setAttribute("linestyle",
parameterMap.get("linestyle")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);
        // </editor-fold>
      } else if(objType.equals("bezier")) {
//<editor-fold>
        obj = new Element("bezier");
        obj.setAttribute("x1",
parameterMap.get("x1")[0]);
        obj.setAttribute("y1",
parameterMap.get("y1")[0]);
        obj.setAttribute("x2",
parameterMap.get("x2")[0]);
        obj.setAttribute("y2",
parameterMap.get("y2")[0]);
        obj.setAttribute("ctrl1x",
parameterMap.get("ctrl1x")[0]);
        obj.setAttribute("ctrl1y",
parameterMap.get("ctrl1y")[0]);
        obj.setAttribute("ctrl2x",
parameterMap.get("ctrl2x")[0]);
        obj.setAttribute("ctrl2y",
parameterMap.get("ctrl2y")[0]);
        obj.setAttribute("arrowstyle1",
parameterMap.get("arrowstyle1")[0]);
        obj.setAttribute("arrowstyle2",
parameterMap.get("arrowstyle2")[0]);
        obj.setAttribute("linestyle",
parameterMap.get("linestyle")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);
        // </editor-fold>
      } else if(objType.equals("path")) {
//<editor-fold>
        obj = new Element("path");
        String xcoordinate[] =
parameterMap.get("x");
        String ycoordinate[] =
parameterMap.get("y");

        String temp = "";
        for(int i = 0; i <
xcoordinate.length; i++) {
            if(i != 0)
              temp = temp + " ";

            temp = temp + xcoordinate[i] +
"," + ycoordinate[i];
        }

        obj.setAttribute("points", temp);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);
        // </editor-fold>
      } else if(objType.equals("class")) { //
<editor-fold>
        obj = new Element("class");
        obj.setAttribute("x",
parameterMap.get("x")[0]);
        obj.setAttribute("y",
parameterMap.get("y")[0]);
        obj.setAttribute("width",
parameterMap.get("width")[0]);
        obj.setAttribute("height",
parameterMap.get("height")[0]);
        obj.setAttribute("fillcolor",
parameterMap.get("fillcolor")[0]);
        obj.setAttribute("textcolor",
parameterMap.get("textcolor")[0]);
        obj.setAttribute("fontsize",
parameterMap.get("fontsize")[0]);
        obj.setAttribute("classfontsize",
parameterMap.get("classfontsize")[0]);
        obj.setAttribute("classname",
parameterMap.get("classname")[0]);
        obj.setAttribute("abstractclass",
parameterMap.get("abstractclass")[0]);

obj.setAttribute("qualifiedassociation",
parameterMap.get("qualifiedassociatio
n")[0]);
        obj.setAttribute("qualifier",
parameterMap.get("qualifier")[0]);
        obj.setAttribute("stereotype",
parameterMap.get("stereotype")[0]);
        obj.setAttribute("templateclass",
parameterMap.get("templateclass")[0]);
        obj.setAttribute("activeclass",
parameterMap.get("activeclass")[0]);

obj.setAttribute("showattributes",
parameterMap.get("showattributes")[0]
);

obj.setAttribute("showoperations",
parameterMap.get("showoperations")[0
]);
        obj.setAttribute("rotate",
parameterMap.get("rotate")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);

        Element attributesNode = new
Element("attributes");
        String attributevalues[] =
parameterMap.get("attributevalue");
        if(attributevalues != null) {
          String attributestatics[] =
parameterMap.get("attributestatic");
            for(int i = 0; i <
attributevalues.length; i++) {
              Element attribute = new
Element("attribute");

attribute.setAttribute("static",
attributestatics[i]);

attribute.setText(attributevalues[i]);

attributesNode.addContent(attribute);
            }
        }
        obj.addContent(attributesNode);

        Element operationsNode = new
Element("operations");
        String operationvalues[] =
parameterMap.get("operationvalue");
        if(operationvalues != null) {
          String operationstatics[] =
parameterMap.get("operationstatic");
            String operationabstracts[] =
parameterMap.get("operationstatic");
            for(int i = 0; i <
operationvalues.length; i++) {
              Element operation = new
Element("operation");

operation.setAttribute("static",
operationstatics[i]);

operation.setAttribute("abstract",
operationabstracts[i]);

operation.setText(operationvalues[i]);

operationsNode.addContent(operation)
;
            }
        }

obj.addContent(operationsNode);

        Element templatesNode = new
Element("templates");
```

```
        String templatevalues[] =
parameterMap.get("templatevalue");
        if(templatevalues != null) {
           for(int i = 0; i <
templatevalues.length; i++) {
              Element template = new
Element("template");

template.setText(templatevalues[i]);

templatesNode.addContent(template);
           }
        }
        obj.addContent(templatesNode);
        // </editor-fold>
     } else if(objType.equals("note") ||
objType.equals("frame")
           || objType.equals("package1")
|| objType.equals("package2")
           ||
objType.equals("component") ||
objType.equals("artifact")
           || objType.equals("usecase")
|| objType.equals("actor")
           ||
objType.equals("superstate") ||
objType.equals("action")
           ||
objType.equals("subactivity") ||
objType.equals("timesignal")
           ||
objType.equals("acceptsignal") ||
objType.equals("sendsignal")
           || objType.equals("connector")
|| objType.equals("transformation")
           || objType.equals("part")) {
//<editor-fold>
        obj = new Element(objType);
        obj.setAttribute("fillcolor",
parameterMap.get("fillcolor")[0]);
        obj.setAttribute("fontstyle",
parameterMap.get("fontstyle")[0]);
        obj.setAttribute("fontweight",
parameterMap.get("fontweight")[0]);
        obj.setAttribute("fontsize",
parameterMap.get("fontsize")[0]);
        obj.setAttribute("fontfamily",
parameterMap.get("fontfamily")[0]);
        obj.setAttribute("height",
parameterMap.get("height")[0]);
        obj.setAttribute("label",
parameterMap.get("label")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("rotate",
parameterMap.get("rotate")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);
        obj.setAttribute("textcolor",
parameterMap.get("textcolor")[0]);
        obj.setAttribute("width",
parameterMap.get("width")[0]);
        obj.setAttribute("x",
parameterMap.get("x")[0]);
        obj.setAttribute("y",
parameterMap.get("y")[0]);
        // </editor-fold>
     } else if(objType.equals("polygon"))
{ //<editor-fold>
        obj = new Element("polygon");
        String xcoordinate[] =
parameterMap.get("x");
        String ycoordinate[] =
parameterMap.get("y");

        for(int i = 0; i <
xcoordinate.length; i++) {
           Element point = new
Element("point");
           point.setAttribute("x",
xcoordinate[i]);
           point.setAttribute("y",
ycoordinate[i]);
           obj.addContent(point);
        }

        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);
        obj.setAttribute("fillcolor",
parameterMap.get("fillcolor")[0]);
        // </editor-fold>
     } else if(objType.equals("polyline"))
{ //<editor-fold>
        obj = new Element("polyline");
        String xcoordinate[] =
parameterMap.get("x");
        String ycoordinate[] =
parameterMap.get("y");

        for(int i = 0; i <
xcoordinate.length; i++) {
           Element point = new
Element("point");
           point.setAttribute("x",
xcoordinate[i]);
           point.setAttribute("y",
ycoordinate[i]);
           obj.addContent(point);
        }

        obj.setAttribute("arrowstyle1",
parameterMap.get("arrowstyle1")[0]);
        obj.setAttribute("arrowstyle2",
parameterMap.get("arrowstyle2")[0]);
        obj.setAttribute("linestyle",
parameterMap.get("linestyle")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);
        // </editor-fold>
     } else if(objType.equals("instance"))
{ // <editor-fold>
        obj = new Element("instance");
        obj.setAttribute("x",
parameterMap.get("x")[0]);
        obj.setAttribute("y",
parameterMap.get("y")[0]);
        obj.setAttribute("width",
parameterMap.get("width")[0]);
        obj.setAttribute("height",
parameterMap.get("height")[0]);
        obj.setAttribute("fillcolor",
parameterMap.get("fillcolor")[0]);
        obj.setAttribute("textcolor",
parameterMap.get("textcolor")[0]);
        obj.setAttribute("fontsize",
parameterMap.get("fontsize")[0]);
        obj.setAttribute("instancename",
parameterMap.get("instancename")[0])
;
        obj.setAttribute("classname",
parameterMap.get("classname")[0]);

obj.setAttribute("showattributes",
parameterMap.get("showattributes")[0]
);
        obj.setAttribute("rotate",
parameterMap.get("rotate")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);

        Element attributesNode = new
Element("attributes");
        String attributevalues[] =
parameterMap.get("attributevalue");
        if(attributevalues != null) {
           String attributestatics[] =
parameterMap.get("attributestatic");
           for(int i = 0; i <
attributevalues.length; i++) {
              Element attribute = new
Element("attribute");

attribute.setAttribute("static",
attributestatics[i]);

attribute.setText(attributevalues[i]);

attributesNode.addContent(attribute);
           }
        }
        obj.addContent(attributesNode);
        // </editor-fold>
     } else if(objType.equals("node")) {
// <editor-fold>
        obj = new Element("node");
```

```
        obj.setAttribute("x",
parameterMap.get("x")[0]);
        obj.setAttribute("y",
parameterMap.get("y")[0]);
        obj.setAttribute("width",
parameterMap.get("width")[0]);
        obj.setAttribute("height",
parameterMap.get("height")[0]);
        obj.setAttribute("topheight",
parameterMap.get("topheight")[0]);
        obj.setAttribute("fillcolor",
parameterMap.get("fillcolor")[0]);
        obj.setAttribute("textcolor",
parameterMap.get("textcolor")[0]);
        obj.setAttribute("fontsize",
parameterMap.get("fontsize")[0]);
        obj.setAttribute("nodename",
parameterMap.get("nodename")[0]);
        obj.setAttribute("stereotype",
parameterMap.get("stereotype")[0]);
        obj.setAttribute("showartifacts",
parameterMap.get("showartifacts")[0]);
        obj.setAttribute("rotate",
parameterMap.get("rotate")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);

        Element containedartifactsNode
= new Element("containedartifacts");
        String containedartifactvalues[] =
parameterMap.get("containedartifactva
lue");
        if(containedartifactvalues != null)
{
            for(int i = 0; i <
containedartifactvalues.length; i++) {
                Element containedartifact =
new Element("containedartifact");

containedartifact.setText(containedartif
actvalues[i]);

containedartifactsNode.addContent(con
tainedartifact);
            }
        }

obj.addContent(containedartifactsNode
);

        Element taggedvaluesNode =
new Element("taggedvalues");
        String taggedvaluevalues[] =
parameterMap.get("taggedvaluevalue")
;
        if(taggedvaluevalues != null) {
            for(int i = 0; i <
taggedvaluevalues.length; i++) {
                Element taggedvalue = new
Element("taggedvalue");

taggedvalue.setText(taggedvaluevalues[
i]);

taggedvaluesNode.addContent(taggedv
alue);
            }
        }

obj.addContent(taggedvaluesNode);
        // </editor-fold>
      } else if(objType.equals("statebox"))
{ // <editor-fold>
        obj = new Element("statebox");
        obj.setAttribute("x",
parameterMap.get("x")[0]);
        obj.setAttribute("y",
parameterMap.get("y")[0]);
        obj.setAttribute("width",
parameterMap.get("width")[0]);
        obj.setAttribute("height",
parameterMap.get("height")[0]);
        obj.setAttribute("fillcolor",
parameterMap.get("fillcolor")[0]);
        obj.setAttribute("textcolor",
parameterMap.get("textcolor")[0]);
        obj.setAttribute("fontsize",
parameterMap.get("fontsize")[0]);
        obj.setAttribute("statename",
parameterMap.get("statename")[0]);

obj.setAttribute("showinternalactivities
",
parameterMap.get("showinternalactiviti
es")[0]);
        obj.setAttribute("rotate",
parameterMap.get("rotate")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);

        Element internalactivitysNode =
new Element("internalactivitys");
        String internalactivityvalues[] =
parameterMap.get("internalactivityvalu
e");
        if(internalactivityvalues != null) {
            for(int i = 0; i <
internalactivityvalues.length; i++) {
                Element internalactivity =
new Element("internalactivity");

internalactivity.setText(internalactivityv
alues[i]);

internalactivitysNode.addContent(inter
nalactivity);
            }
        }

obj.addContent(internalactivitysNode);
        // </editor-fold>
      } else
if(objType.equals("expansionregion")) {
//<editor-fold>
        obj = new
Element("expansionregion");
        obj.setAttribute("fillcolor",
parameterMap.get("fillcolor")[0]);
        obj.setAttribute("height",
parameterMap.get("height")[0]);
        obj.setAttribute("linewidth",
parameterMap.get("linewidth")[0]);
        obj.setAttribute("linejoin",
parameterMap.get("linejoin")[0]);
        obj.setAttribute("linecap",
parameterMap.get("linecap")[0]);
        obj.setAttribute("rotate",
parameterMap.get("rotate")[0]);
        obj.setAttribute("strokestyle",
parameterMap.get("strokestyle")[0]);
        obj.setAttribute("width",
parameterMap.get("width")[0]);
        obj.setAttribute("listboxpin1x",
parameterMap.get("listboxpin1x")[0]);
        obj.setAttribute("listboxpin1y",
parameterMap.get("listboxpin1y")[0]);
        obj.setAttribute("listboxpin2x",
parameterMap.get("listboxpin2x")[0]);
        obj.setAttribute("listboxpin2y",
parameterMap.get("listboxpin2y")[0]);
        obj.setAttribute("listboxpinsize",
parameterMap.get("listboxpinsize")[0]);
        // </editor-fold>
      }
    }

    // for attributes, operations,
templates, artifacts, tagged values and
internal activities
    public CordieInsertOperation(String
eltName, String txt, int pos) {
      obj = new Element(eltName);
      obj.setText(txt);
      if(eltName.equals("attribute")) {
        obj.setAttribute("static", "false");
      } else
if(eltName.equals("operation")) {
        obj.setAttribute("static", "false");
        obj.setAttribute("abstract",
"false");
      }
      position = pos;
    }

    // for point
    public CordieInsertOperation(String
eltName, String iX, String iY, int pos) {
      obj = new Element(eltName);
      obj.setAttribute("x", iX);
      obj.setAttribute("y", iY);
      position = pos;
    }

    public void setPosition(int iPos) {
      position = iPos;
    }
```

```
    public void decrementPosition() {
        position--;
    }

    public void incrementPosition() {
        position++;
    }

    public Element getObject() {
        return obj;
    }

    public int getPosition() {
        return position;
    }

    @Override
    public String toString() {
        return "{ \"optype\" : \"insert\",
\"object\" : " +
CordieObjectConverter.convert(obj)
                + ", \"position\" : " + position +
" }";
    }
}
```

CordieMoveOperation.java

```
package cordie.diagram.operation;

public class CordieMoveOperation
implements CordieOperation {
    private int position;
    private int destination;

    public
CordieMoveOperation(CordieMoveOper
ation cmo) {
        position = cmo.getPosition();
        destination = cmo.getDestination();
    }

    public CordieMoveOperation(int iPos,
int iDest) {
        position = iPos;
        destination = iDest;
    }

    public void decrementPosition() {
        position--;
    }

    public void incrementPosition() {
        position++;
    }

    public int getPosition() {
        return position;
    }

    public void setPosition(int iPos) {
        position = iPos;
    }

    public int getDestination() {
        return destination;
    }

    public void setDestination(int iDes) {
        destination = iDes;
    }

    public void decrementDestination() {
        destination--;
    }

    public void incrementDestination() {
        destination++;
    }

    @Override
    public String toString() {
        return "{ \"optype\" : \"move\",
\"position\" : " + position + ", "
                + "\"destination\" : " +
destination + " }";
    }
}
```

CordieNoneOperation.java

```
package cordie.diagram.operation;

public class CordieNoneOperation
implements CordieOperation {

    public CordieNoneOperation() {
    }

    @Override
    public String toString() {
        return "{ \"optype\" : \"none\" }";
    }
}
```

CordieOperation.java

```
package cordie.diagram.operation;

public interface CordieOperation {
}
```

CordieRemoveCollaboratorOperation.ja
va

```
package cordie.diagram.operation;

import
org.apache.commons.lang3.StringEscap
eUtils;

public class
CordieRemoveCollaboratorOperation
implements CordieOperation {
    private String toRemove;

    public
CordieRemoveCollaboratorOperation(C
ordieRemoveCollaboratorOperation co)
{
        this.toRemove = co.getUsername();
    }

    public
CordieRemoveCollaboratorOperation(St
ring str) {
        this.toRemove = str;
    }

    public String getUsername() {
        return toRemove;
    }

    @Override
    public String toString() {
        return "{ \"optype\" :
\"removecollaborator\",
\"collaborator\" : \""
                +
StringEscapeUtils.escapeJava(toRemove
) + "\" }";
    }
}
```

CordieRemoveUserOperation.java

```
package cordie.diagram.operation;

import
org.apache.commons.lang3.StringEscap
eUtils;

public class
CordieRemoveUserOperation
implements CordieOperation {
    private String toRemove;

    public
CordieRemoveUserOperation(CordieRe
moveUserOperation co) {
        this.toRemove = co.getUsername();
    }

    public
CordieRemoveUserOperation(String str)
{
        this.toRemove = str;
    }

    public String getUsername() {
        return toRemove;
    }

    @Override
    public String toString() {
        return "{ \"optype\" :
\"removeuser\", \"collaborator\" : \""
                +
StringEscapeUtils.escapeJava(toRemove
) + "\" }";
    }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  <filter>
    <filter-name>LoginFilter</filter-name>
    <filter-class>cordie.AuthenticateFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>LoginFilter</filter-name>
    <url-pattern>/members/*</url-pattern>
  </filter-mapping>
  <servlet>
    <servlet-name>LogInExec</servlet-name>
    <servlet-class>cordie.LogInExec</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>DisplayPicture</servlet-name>
    <servlet-class>cordie.DisplayPicture</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>DiagramList</servlet-name>
    <servlet-class>cordie.DiagramList</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>FileServlet</servlet-name>
    <servlet-class>cordie.FileServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LogInExec</servlet-name>
    <url-pattern>/LogInExec</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>DisplayPicture</servlet-name>
    <url-pattern>/members/DisplayPicture</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>DiagramList</servlet-name>
    <url-pattern>/members/DiagramList</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>CordieEditor</servlet-name>
    <servlet-class>cordie.CordieEditor</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CordieEditor</servlet-name>
    <url-pattern>/members/CordieEditor</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>FileServlet</servlet-name>
    <url-pattern>/members/file</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <resource-ref>
    <description>Database for CORDIE application</description>
    <res-ref-name>jdbc/CORDIE</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
  </resource-ref>
</web-app>
```

---

sun-web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD GlassFish Application Server 3.0 Servlet 3.0//EN" "http://www.sun.com/software/appserver/dtds/sun-web-app_3_0-0.dtd">
<sun-web-app error-url="">
 <context-root>/Cordie</context-root>
 <class-loader delegate="true"/>
 <jsp-config>
  <property name="keepgenerated" value="true">
   <description>Keep a copy of the generated servlet class' java code.</description>
  </property>
 </jsp-config>
</sun-web-app>
```

ACKNOWLEDGEMENT

First, I would like to thank Doc Vincent Peter Magboo for the guidance he provided me in writing my special problem and for sharing his valuable insights.

This would not have been possible also without the help of my adviser, Mr. Aldrich Colin Co. He gave me the early ideas that eventually led to this special problem. And without his encouragement and advice, I would not have completed this study.

I would also like to express my gratitude to Mr. Geoffrey Solano and Ms. Avegail Carpio who seemed like my guardian angels during my defense, and for being two of the kindest professors I've met throughout my college life.

It's also a pleasure to thank some of my friends. Franz Zerleen Barba, for being my best friend in high school and up to my first two years in college, and without whom I would not have been able to pass the UPCAT in the first place.

Kathleen Benavidez, who gave me the right kind of pressure to finish this.

Ruth Ann Cabria, without whom I wouldn't have found a job. And for always helping me whenever I needed it.

Sevenworks, for the trust they gave me and letting me be a part of their team.

Ahmad Arceo, for being always ready to help and for encouraging me.

Of course, to my mom Marisa Villarico. Thank you for everything.

Last but not the least, to God. Thank You for never failing to answer my prayers.