# Web-Based CASE Tool for Automated Rendering of UML Models

**Sellappan Palaniappan and Louis Ling,**

Department of Information Technology,
Malaysia University of Science and Technology, Kelana Square, 47301 Petaling Jaya, Malaysia

**Abstract**
Traditional CASE tools are desktop-based, so they do not support online software collaboration. That is, they do not allow members of software project teams who are geographically distributed to collaborate and work together on software projects. This constrains software houses from tapping into global talent pools that can help reduce software costs and acquire needed expertise. Online CASE tools can help solve this problem. This paper presents a prototype web-based CASE tool that supports automated rendering of UML models by enabling team members who are geographically distributed to work together. Software modelers specify their software systems using a set of triplets for each UML diagram and the tool uses the triplets to automatically render high-quality SVG graphics, thus eliminating the need for manual diagramming. The current version supports three UML diagrams: Class, Use Case and Sequence. The tool is implemented using Active Server Page (the Microsoft's server-side scripting engine) and VBScript (the Microsoft's Visual Basic Scripting Language) and can be accessed on the Web.

***Keywords:***
*Web-based CASE tools, Scalable Vector Graphics, collaborative software modeling, UML diagrams.*

## 1. Introduction

Software houses (developers) typically use CASE tools to develop their software systems. CASE tools can help improve the quality of software produced. They can be used to specify, design, construct, test and document software systems. They can also reduce costs and improve delivery times. Today's CASE tools have varying degrees of sophistication. Some support early phases of software processes, e.g., specification; some, later phases, e.g., code generation; and some, all phases (including forward and reverse engineering).

Most existing commercial CASE tools are desktop-based, i.e., they cannot be accessed on the Web. This limits software houses to tap into global talent pools which can help to reduce software costs and acquire needed expertise. Online or Web-based CASE tools can help overcome this problem as they allow developers who are geographically distributed to collaborate and work together on software projects. Thus there is a strong case for online CASE tools.

As Internet technology has matured over the years and has now become the de facto medium for global communication, it provides an ideal platform for collaborative software development. It is standards-based. So it provides a very flexible, inexpensive and effective environment for collaborative software development [17].

Thick-client desktop-based CASE tools provide features such as diagram editing and viewing facilities, sophisticated interaction features, and information management on local workstations [6, 7, 9]. However, they are not accessible via the Web, so they are not very flexible. Thin-client online CASE tools, on the other hand, are more flexible as they are accessible via the Web. They also have a consistent 'feel and look' user interface. Web browsers also eliminate the need to install CASE tools on every workstation whenever they undergo version changes. Also, they are likely to be less expensive compared to desktop-based CASE tools. However, they pose a big challenge: Online CASE tools are restricted in the types of interaction allowed.

This research presents a prototype thin-client Web-based CASE tool that allows members of software teams who are geographically distributed to collaborate and work together on software projects. It allows them to specify software using a set of triplets for each UML diagram and the tool uses the triplets to automatically generate the diagram. It is flexible and cost-effective. The current version supports three UML diagrams: Class, Use Case and Sequence.

## 2. Literature Review

### 2.1 Related Works

Thick-client desktop-based CASE tools provide features such as repository support, versioning control, data modeling and diagram views. However, they lack automated diagramming support and they cannot be accessed via the Web. To overcome these limitations, vendors are now beginning to make CASE tools Web-enabled and provide automated diagram rendering support.

Some popular desktop-based CASE tools are Argo/UML [13], Composers [7], Pounamu [18] and Rational Rose [12]. Because these are based on thick-clients, they must be installed on every workstation. There are also meta-CASE tools such as GraMMI [15], JViews [7] and KOOGE [4] that cater to different user preferences.

Research on online CASE tools has produced several prototypes: BSCW [1], OzWeb [8], Webworlds [2], Web-CASRE [10], Cliki [5], NutCASE [11] and Seek [9]. However, they do not provide automated diagram rendering features; users can only view UML diagrams derived from formal specifications as in TCOZ [16]. Thus there is a need for online CASE tools.

### 2.2 The Unified Modeling Language

The Unified Modeling Language (UML), managed by the Object Management Group (OMG), provides industry standard mechanisms for visualizing, specifying, constructing, testing and documenting artifacts of software systems [14]. UML is used to understand, design, browse, configure, maintain and control information on such systems. UML is intended to be used with all development methods, life cycle stages, application domains and media. However, the manner in which these components are used in lifecycle stages varies. A recent study has shown that there is considerable variation in the use of these components: Class, Sequence and Use Case diagrams are used most frequently while Collaboration diagrams are used least frequently [3].

### 2.3 Scalable Vector Graphics

Scalable Vector Graphics (SVG) is an XML-based graphics standard from the World Wide Web Consortium (W3C). SVG enables Web developers to go beyond the limitations of HTML. It lets them create robust visual contents using a simple declarative programming model. SVG produces a two-dimensional vector graphics to display several types of information, e.g., statistics, graphs, maps, technical diagrams. It is particularly suited to creating graphics from XML data. Interest on tools for creating and viewing SVG files on the Web is still growing.

## 3. Design and Methods

### 3.1 Building Blocks for UML Diagrams

A fundamental building block of a UML diagram is the combination of two identity elements and a binding element that creates bonding relationship between the two identity elements. The identity and binding elements however may differ in different UML diagrams (e.g., in class diagrams, the identity elements are classes and the binding elements are relationships). Some identity and binding elements are unique to a particular diagram (e.g., extend and include relationships in a use case) while others may be shared among diagrams (e.g., the actor element in use case diagram can appear as stereotype in a sequence diagram).

Note that each diagram is made up of a finite number of identity and binding elements (use case consists of actors, use cases, and relationships between actors, between actors and use cases, and between use cases). The tool uses this information to render diagrams automatically. Every building block or UML diagram is expressed as a set of triplets. With this one can deconstruct any diagram and generate the corresponding triplets that provide the visual information.

### 3.2 Automated Rendering of UML Diagrams

Automated rendering of UML diagrams comprises of sixteen processes, categorized into four groups:

Preliminary

1. Create a pool of identity and binding elements.
2. Create a pool of triplets.
3. Create a virtual canvas.
4. Insert triplets into the virtual canvas.
5. Update the virtual canvas version.
6. Generate a unique filename for SVG and HTML documents.

SVG Document Authoring

7. Create a SVG document using the unique filename.
8. Write into the SVG document the SVG starting framework.

9. Write into the SVG document the definitions of objects and elements.
10. Write into the SVG document the positions of objects and elements.
11. Write into the SVG document the SVG closing framework.

HTML Document Authoring

12. Create a HTML document from the unique filename.
13. Write into the HTML document the HTML starting framework.
14. Embed the SVG document into the HTML document.
15. Write into the HTML document the HTML closing framework.

Database Update

16. Update the server database.

### 3.3 Triplet Positioning Algorithm

Although all diagrams are rendered using the steps listed in Section 3.2, not all diagrams share the same elements and triplets or use the same triplet positioning algorithm to position elements on the virtual canvas. The algorithm for rendering class, use case and sequence diagrams are described below.

*Class Diagram*

As the types of relationship between classes are finite, it is easy to determine the positioning of each triplet. The strategy is to group each relationship under one of the following: association, aggregation, composition and generalization, and assign them to different parts of the diagram. Figure 1 shows the general pattern. Each class has a set of four relationships with the centre class being defined as the source class and the classes surrounding it being defined as the groups of destination classes. The automated rendering of class diagrams appear in the area marked “Template”. The absolute positions of class names and multiplicity relationship between classes are determined by using another algorithm.

Each group of destination classes are further aggregated into those that fall under the northern or southern hemisphere. The positions of all destination classes related to a source class are relative to the positions of the starting destination classes for each group and for each hemisphere, which are defined in the template. The positioning of destination classes start at the northern hemisphere and proceeds to the southern hemisphere on an alternate pattern. For each hemisphere, each destination class is assigned a position that branches away from the source class as shown by the arrow for each group.

*Sequence Diagram*

As in the class diagram, the pattern in the sequence diagram is expressed in terms of types of objects and types of messages. Messages in sequence diagrams proceed vertically down and all actor objects are organized from top left. The strategy used in the triplet positioning algorithm is to position all objects in the order of actor objects, boundary objects, entity objects and control objects as shown in Figure 2.

After assigning positions for the interacting objects, their lifelines are drawn. The lifelines depend on the number and type of messages passed. A lifeline can accommodate one sequence triplet if the message is of type self call or two sequence triplets if the message is of any other type.

After setting the layout of the sequence diagram each triplet is assigned a position as follows:

1. Obtain the positions of the source and destination objects of the triplets on the virtual canvas.
2. Determine the direction of the message.
3. Determine the distance between the source and destination objects.
4. Write the message type, message timeline and message contents.
5. Update the starting y-coordinate of the focus of control for the next set of triplet.

*Use Case Diagram*

Use case diagram has only three types of triplets: actor-actor, actor-use case, and use case-use case. In the actor-actor triplet, only one type of relationship exists: inheritance. In the actor-use triplet, only one type of relationship exists: association. In the use case-use case triplet, three types of relationships exist: inheritance, extension and inclusion. For automated rendering of diagrams, only the last two types of triplets are used.

The strategy used in the triplet positioning algorithm is to partition a virtual canvas into three columns: left column for all actors, centre column for all primary use cases; and right column for all subsidiary use cases (Figure 3). The right column is further partitioned into three rows: top row for all subsidiary use cases that have inheritance relationship with primary use case (denoted by the «inherit» link to primary use case);

middle row for all subsidiary use cases that have extension relationship with primary use case (denoted by the «extend» link to primary use case); and bottom row for all subsidiary use cases that have inclusion relationship with the primary use case (denoted by the «include» link to primary use case).

the southern hemisphere on an alternate pattern. For each hemisphere, each subsidiary use case is assigned a position that branches away from the primary use case as shown by the arrow for each group.

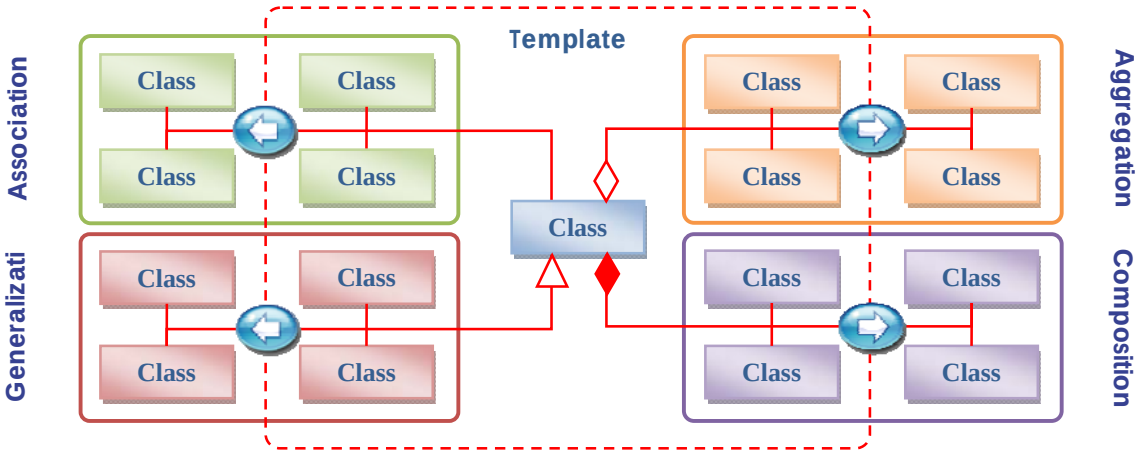


Fig.1 A compendium of four relationships for a class.

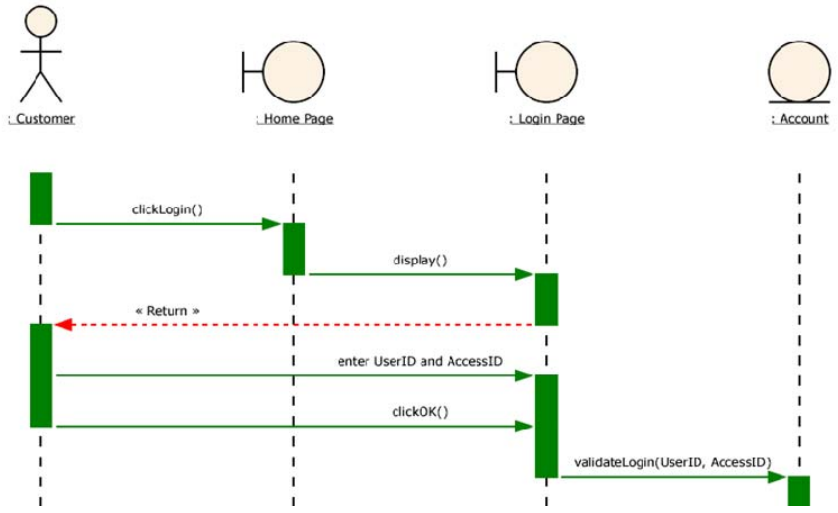


Fig 2 An example of UML sequence diagram created from the automated rendering of UML diagrams.

Figure 3 shows the template for the automated diagram. The positioning of element names is given by another algorithm. The «inherit» and «extend» subsidiary use cases are aggregated further into groups of those that fall in the northern and southern hemispheres. The position of all subsidiary use cases related to the primary use case are relative to the position of the starting subsidiary use case for each group and for each hemisphere. The positioning for each subsidiary use case starts at the northern hemisphere and proceeds to

## 4. Evaluation

### 4.1 File Size Comparison

We can evaluate performance by comparing the size of image files sent over the Internet for the various graphics formats. Table 1 shows the file size for various graphics formats for a sequence diagram. It also shows whether there is loss of color information needed to perform automatic rendering. The smallest

file size without any loss of color information is the SVG format. Next is the GIF format but it suffers from loss of color information. PNG, JPEG and TIFF files retain color information, but they are much larger than the SVG file. The BMP file for all resolutions and colors is the largest and suffers from loss of color information. This tells that BMP format is not suitable for sending images on the Internet.

### 4.2 Triplet Positioning Algorithm Limitations

The triplet positioning algorithms for rendering class, sequence and use case diagrams have their limitations. For class diagram, itemization of each class and grouping of other classes based on relationship for each itemized class can be viewed to give detailed information on each class. However, the algorithm is not designed to position class triplets to create a class diagram where the position of each class is arbitrary. From the viewpoint of each itemized class, the region for each type of class relationship has been predetermined and the position of each class depends on the type of relationship it has with each itemized class.

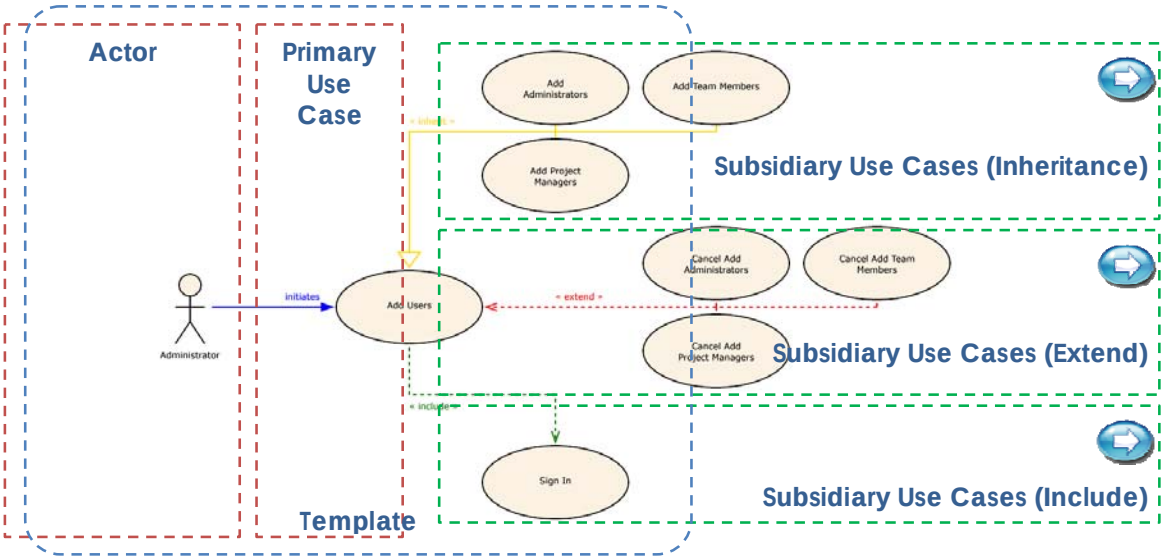


Fig 3 “Virtual” canvas division for UML use case diagram in the automated rendering of UML diagrams.

Table 1. File size comparison for images of different graphics standards.

| Graphics Format | Size (KB) | Loss of Color Information |
|---|---|---|
| SVG | 10.2 | No |
| GIF | 14.9 | Yes |
| PNG | 30.8 | No |
| JPEG | 48.9 | No |
| TIFF | 66.2 | No |
| Monochrome BMP | 109 | Yes |
| 16-Colour BMP | 434 | Yes |
| 256-Colour BMP | 870 | Yes |

The study has shown that for any image, the file size of vector graphics is always smaller than the file size of raster graphics. The vector graphics always retains color information while the raster graphics may or may not retain color information.

For sequence diagram, ordering of participating objects presents a problem. The current practice orders the actor objects to the left of the diagram. While results of the automated rendering correspond to the ordering of participating objects (the first object in the set of interactions appears at the left and the last object at the right), there is no mechanism for the user to determine the order of participating objects manually. It is possible to introduce some flexibility that allows the user to manually order the participating objects.

For use case diagram, itemization of each primary use case and grouping of subsidiary use cases based on relationships with each itemized primary use case can be viewed to have more information on each class. However, the algorithm is not designed to position

triplets where the position of each class is arbitrary. From the viewpoint of each itemized primary use case, the region for each type of primary use case relationship with subsidiary use cases has been predetermined and the position of each subsidiary use case depends on the type of relationship it has with each itemized class.

Current categorization of use case triplets is still incomplete. We could add another category of triplets: subsidiary use case-subsidiary use case. The triplet interaction in use case is more complex and the categorization fails to address this complexity.

## 5. Conclusion and Future Work

Although existing thick client desktop-based CASE tools perform complex tasks covering all or most of the software processes, they have two shortcomings: (1) they lack automatic diagram rendering and (2) they are not Web-enabled. This research has presented a prototype thin-client Web-based CASE tool that allows members of software teams who are geographically distributed to collaborate and work together on software projects. The tool overcomes both these limitations. It generates UML diagrams automatically and it is Web-enabled. The current version supports Class, Use Case and Sequence diagrams. Further work can incorporate other UML diagrams. The triplet positioning algorithms can be further improved to provide dynamic triplet positioning and interactive system-rendered diagrams.

**Acknowledgements**

This paper was supported by E-Science Research Grant (No. 01-02-05-SF0016), Ministry of Science and Technology and Innovation (MOSTI), Government of Malaysia.

## References

[1] Bentley, R., Horstmann, T., Sikkel, K., and Trevor, J. (1995). Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System, In *Proc. of the 4th International World Wide Web Conference*, Boston, Massachusetts, 11–14 December 1995.

[2] Chalk, P. D. (2000). Webworlds – Online Modeling Environment for Learning Software Engineering, *Journal of Computer Science Education*, 10(1), April 2000, pp. 39–56.

[3] Dobing, B and Parsons, J. (2006). How UML Is Used? *Communications of the ACM*, 49(5), May 2006, pp. 109-113.

[4] Ebert, J., Süttenbach, R., Uhe, I. (1997). Meta-CASE in Practice: A Case for KOGGE, In *Proc. of the 9th International Conference on Advanced Information Systems Engineering*, Lecture Notes on Computer Science, 1250, Barcelona, Spain, Springer-Verlag, pp. 203–216.

[5] Gordon, D., Biddle, R., Noble, J. and Tempero, E. (2003). A Technology for Lightweight Online Visual Applications, In *Proc. of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments*, Auckland, New Zealand, 28–31 October 2003, *IEEE Computer Society Press*, pp. 245–247.

[6] Green, T. R. G. (1989). Cognitive Dimensions of Notations, In A. Sutcliffe and L. Macaulay (Eds.) *Proc. of the 5th Conference of the British Computer Society, Human-Computer Specialist Group on People and Computers V*, Cambridge, United Kingdom, *Cambridge University Press*, pp. 443–460.

[7] Grundy, J. C., Mugridge, W. B., and Hosking, J. G. (2000). Constructing Component-Based Software Engineering Environments: Issues and Experiences, *Journal of Information and Software Technology*, 42(2), January 2000, pp. 117–128.

[8] Kaiser, G. E., Dossick, S. E., Jiang, W., Yang, J. J., Ye, S. X. (1998). WWW-Based Collaboration Environments with Distributed Tool Services, *World Wide Web*, 1(1), March 1998, pp. 3–25.

[9] Khaled, R., MacKay, D., Biddle, R., Noble, J. and Tempero, E. (2002). A Lightweight Online CASE Tool for Sequence Diagrams, In *Proc. of SIGCHI-NZ Symposium on Computer-Human Interaction*, Hamilton, New Zealand, 11–12 July 2002, pp. 55–60.

[10] Lyu, M. R. and Schönwälder, J. (1998). Web-CASRE: An Online Tool for Software Reliability Modeling, In *Proc. of the 9th International Symposium on Software Reliability Engineering*, Paderborn, Germany, 4–7 November 1998, *IEEE Computer Society Press*, pp. 151–160.

[11] Mackay, D., Biddle, R. and Noble, J. (2003). A Lightweight Online CASE Tool for UML Class Diagrams, In *Proc. of the 4th Australasian User Interface Conference on User Interfaces*, Adelaide, South Australia, 4–7 February 2003, Australian Computer Society, pp. 95–98.

[12] Quatrani, T. and Booch, G. (2002). *Visual Modeling with Rational Rose 2002 and UML* (3rd ed.), Addison-Wesley Professional, 2003.

[13] Robbins, J., Hilbert, D. M., and Redmiles, D. F. (1998). Extending Design Environments to Software Architecture Design, *Journal of Automated Software Engineering*, 5(3), July 1998, pp. 261–290.

[14] Rumbaugh, J., Jacobson, I., and Booch, G. (2005). *The Unified Modeling Language Reference Manual* (2nd ed.). New York: Addison-Wesley Professional.

[15] Sapia, C., Blaschka, M. and Höfling, G. (2000). GraMMi: Using a Standard Repository Management System to Build a Generic Graphical Modeling Tool, In *Proc. of the 33rd Hawaii International Conference on System Sciences*, Maui, Hawaii, 4–7 January 2000, *IEEE Computer Society Press*, pp. 1–10.

[16] Sun J., Dong, J. S., Liu, J. and Wang, H. (2001). An XML/XSL Approach to Visualize and Animate TCOZ, In *Proc. of the 8th Asia-Pacific on Software Engineering Conference*, Macau SAR, China, 4–7 December 2001, *IEEE Computer Society Press*, pp. 453–460.
[17] Webster, M. (2005). The Requirements for Managing the Geographically Distributed Development Organization and the CollabNet Solution, *IDC*, February 2005. Retrieved 16 May 2007, from http://enterprise-development.open.collab.net/files/documents/86/24/.
[18] Zhu, N., Grundy, J. C., and Hosking, J. G. (2004). Pounamu: A Meta-Tool for Multi-View Visual Language Environment Construction, In *Proc. of International Conference on Visual Languages and Human Centric Computing*, Rome, Italy, 25–29 September 2004, *IEEE Computer Society Press*, pp. 254–256.

**Sellappan Palaniappan** obtained his PhD in Interdisciplinary Information Science from University of Pittsburgh and a MSc in Computer Science from University of London. He is an Associate Professor at the Department of Information Technology, Malaysia University of Science and Technology. His current research interests include information integration, clinical decision support systems, OLAP and data mining, web services and collaborative CASE tools.

**Louis Ling** obtained his MSc in Information Technology from Malaysia University of Science and Technology (MUST). He is a Research Affiliate Officer in the Department of Information Technology at MUST. His current research interests include Collaborative CASE Tool and Scalable Vector Graphics (SVG).