Supporting the Comprehension of Complex Software Systems

Claudia Susie C. Rodrigues
susie@cos.ufrj.br

Cláudia M. L. Werner
werner@cos.ufrj.br

Programa de Engenharia de Sistemas e Computação COPPE/UFRJ

Abstract

Graphical representations improve the understandability of a complex system. They perform a crucial role in teaching Software Modeling. The rapid changes and increased complexity of software systems present new challenges and put new demands on software engineering education. New proposals have been introduced in the classroom, especially those that make teaching more attractive to students. This paper presents the VisAr3D approach that was designed to provide a 3D visualization of UML models, where the user should intuitively understand the modeling elements in this 3D environment. It includes exploration, interaction and simulation resources to establish a practical and pleasant learning activity, focusing in large scale systems.

Keywords: Software Engineering Education, Software Modeling, Virtual Reality, Augmented Reality, Large Scale Systems.

Introduction

Software engineering teaching experienced pressures over the years as systems become larger and more complex with a great demand for quality. The academy
teaches theory and concepts, and in practical terms, students write relatively small programs in a well-defined domain, sometimes in a restricted time. According to Conn [1], many companies view people straight out of school as raw material to be shaped to the corporate culture. They are forced to supplement their education with training and preparations that provide them with knowledge that supplies this deficiency. Generally, students leave college without dealing with complex problems, working with a team of people under the control of a process, testing and evaluating products, communicating concepts effectively, selling and marketing ideas to others, and measuring product and process quality analytically.

In recent years, the education community has invested much effort in developing new ways of teaching Software Engineering. New education proposals have been introduced in the classroom to meet these new demands and new challenges, especially those that make teaching more attractive to students.

This research meets these challenges by combining features of emerging technologies of 3D visualization, such as virtual reality and augmented reality, recognizing that visual communication can be a key factor in the process of teaching and learning of future software architects due to their high expressiveness and power to be adapted to students' preferences.

According to Feijs and Jong [2], 3D designs can be attractive and fun to walk through. Such visual appeal can be used for presentation purposes, explaining software design to potential customers. As Ware and Hui [3] state, there is an increasing evidence that it is possible to perceive and understand complex information systems if they are displayed as graphical objects in a three dimensional space.

Visualization in 3D space can significantly contribute in education by providing a process of exploration, discovery, observation and construction of a new vision of knowledge, helping the learner to better understand the object of study. Therefore, this technology has the potential to collaborate in the student's cognitive process, providing not only the theory but also practical experience of the content in question.
Although the question of what are the benefits that a 3D representation offers over 2D still remains to be answered, some experiments have shown some optimistic results. Feijs and Jong [2] affirm that 3D makes it possible to create a layout of the design objects that is more consistent with the object’s intended role than it can be done in a single plane. According to Gil and Kent [4], the third dimension is not a mere embellishment. The semantics it carries considerably enhances the expressive power of graphical notations, without resorting to the clutter of textual annotations. Ware and Hui [3] indicate that representing diagrams in 3D can allow more complex information to be comprehended.

The remainder of this paper is structured as follows: Section 2 briefly describes some issues about the study of complex systems. Section 3 presents some related work. Section 4 describes the VisAr3D approach. Finally, Section 5 addresses some final considerations.

Comprehension of Complex Software Systems

As society increasingly relies on software for critical operations, software failures become intolerable. Such complex systems failure can cost lives or, in a commercial software product, can result in a very costly recall of thousands of units.

Booch [5] states that the distinguishing characteristic of industrial-strength software is that it is intensely difficult, if not impossible, for the individual developer to comprehend all the subtleties of its design. He also affirms that the complexity seems to be an essential property of all large software systems, and we should master this complexity. The study of complex systems deals with problems that are difficult to solve and often hard to understand because of their indirect effects. In particular, beginners, as students, for example, may face serious difficulties in understanding the details of such systems.

Given this situation, the role of software engineering, according to Jennings [6], is to provide structures and techniques that make it easier to handle complexity. Based on
these demands, the main purpose of this research is to use 3D visualization technologies to help the understanding of UML models of a complex system. The tangible benefits of achieving this goal is to support the understanding of models, by developing Visar3D (Software Architecture Visualization in 3D), which aims to provide the user the ability to manipulate and analyze a large amount of data from multiple perspectives through an attractive experience. This approach allows students to first participate and explore large scale systems and later be prepared to build their own diagrams. For the purpose of this paper, a large system, or complex system, is defined as any software system composed of a great number of parts that have many interactions among them.

Related Work

Previous research has shown that there are practical advantages to use 3D for UML visualization. For example, McIntosh et. al. [7] focused on the “visual” aspect of UML as the means of communicating information to the viewer. The main difference regarding the VisAr3D approach is how it explores the 3D environment to provide a means for understanding complex software system models. Geon [8] is a new diagrammatic method for visualizing UML class diagrams. The authors indicate that the Geon diagram is both easier to understand and easier to remember than traditional UML class diagram. This important requirement, i.e., to be easier to understand, is also present in our approach. Gil and Kent [4] describe a 3D graphical notation that combines several familiar 2D diagrams. In VisAr3D, the basic idea is to make 3D visualization as familiar as 2D, and also make it intuitive. As it happens in VisAr3D, Radfelder and Gogolla’s work [9] exploits static diagrams (but their work also incorporates dynamic diagrams).

Dwyer [10] uses a Force Directed Algorithm to layout three dimensional UML class diagrams. The prototype incorporates the construction and edition of UML models as well as simple visualization. Visar3D only uses simple visualization.
The GEF3D framework [11] builds on the Eclipse Graphical Editor Framework and as such enables existing Eclipse 2D UML editors to be extended into 3D. Finally, Lange et. al. [12] propose a framework that supports task-oriented modeling and have developed views and visualization techniques. Our approach uses diagram views to visualize the system from different perspectives.

**VisAr3D Approach**

This paper presents VisAr3D (Software Architecture Visualization in 3D) approach designed to provide a 3D visualization of a UML model automatically generated from an existing 2D diagram. This solution explores different perspectives, relationships, abstractions, and contextualized documentation, where the user should intuitively understand the modeling elements in this 3D environment. The idea is to reuse all the existing and relevant information related to the modeling elements of a system. These important data are associated to the modeling elements and the user explores and interacts with them in the third dimension.

By developing a 3D visualization system for the display of models in a new perspective, it is possible to deal with more elaborate and complex designs, similar to those developed by industry. An intuitive exploration and interaction by software modeling students and by the teacher, using resources and facilities such as the manipulation of models in a simulated learning environment by means of virtual elements, is supposed to provide an attractive experience to the learning process.

Currently, VisAr3D exploits a static view of the system model, emphasizing its static structure, although there is a plan to represent its behavior through a dynamic view as well in the future.

**Basic System Principles**
In [13], a set of general design principles for creating learning environments and tools is described to help students understand scientific perspectives on complex systems, based on recent constructivist models of learning and considering the successes and challenges identified in recent education and complex system projects. Those principles confirm VisAr3D basic principles in several points. Jacobson et al. [13] believe that by integrating complex systems knowledge and methodologies throughout the pre-college and college curriculums will foster trajectories of students’ learning that will lead to conceptual growth and deeper understandings over time, grade levels, and topics.

VisAr3D is guided by the following eight principles:

(1) It must support the development and participation of students in complex projects. They need opportunities to directly experience complex systems. The designs found in the labor market are generally large, with requirements such as performance, reliability, low cost and high quality. This approach must support the development and students’ participation in such context. It should enhance the modeling of a system by a large group of students and support the understanding of the parts, to facilitate the understanding of the whole.

(2) It must reduce the gap between theory and practice. Allied to theory, supporting practice brings a greater dynamism to the classroom and results in a more meaningful participation of students. Students often learn theory best when it appears in concrete examples and has immediate relevance to them. It must also facilitate students’ autonomous learning and creative self-expression, supporting simple tasks such as reading and writing.

(3) It must support the activities of students in the process of assimilation of knowledge and skills. It is intended to help improve communication between group members, help in organization and division of tasks within the team, allowing the integration of individual results as a group activity at work, and collaborate with the resolution of complex problems, through cooperation, i.e., to propagate knowledge.
among participants. These are some issues identified in the systematic review previously conducted [14]. Besides, Jacobson et al. [13] state that, through collaborative interactions and the construction of shared artifacts, opportunities naturally arise for students to articulate or reify their ideas, and to reflect on the possible limitations of their ideas and theories.

(4) It must be attractive to students. It must provide an exploratory and intuitive environment that is conducive to discoveries and challenges, and responds to their interaction. It should provide many opportunities for students to obtain feedback on their ideas, and to evolve and refine them over the course of their projects. While students’ experience succeeds, and consequently a sense of competence is achieved, their motivation is enhanced to pursue further learning [15].

(5) The simplest is the best. The main goals for this visualization tool are simplicity, ease-of-use and effectiveness, due to the fact that both software engineering teachers and students have limited time to learn new concepts and will most probably lose interest in using the tool, if it imposes a steep learning curve as a prerequisite for getting familiar with it.

(6) The target 3D diagrams must have a similar 2D source diagram vision, so that it becomes a familiar landscape over extended use. The user must easily recognize his/her own diagram, even in a different guise.

(7) Visual enhancement is necessary to differentiate additional information to be explored and interacted. Using depth and color represent new views provided and presented in Section 3.3. It is important that relevant complex systems concepts be made salient and explicit to the learner.

(8) Details such as attribute, method and association names are initially hidden to reduce visual pollution. With very large diagrams, it is essential to obtain clarity. However, they are shown when requested by the user.

**Supporting System Overview**
This section provides an overview of the VisAr3D supporting system. It also includes a discussion of the important components and subsystems that were used to build the 3D class diagram.

The VisAr3D tool (the tool has the same name of the approach) was built using Java, X3D [16], Xj3D [17] and XMI (XML Metadata Interchange) [18] version 2.1. XMI is a standard from OMG (Object Management Group) that uses UML information to create XML documents. By using XML, UML information can be more readily transferred between tools. It also allows the information to be more readily transformed into new forms, such as 3D.

X3D is an open standard and a trademark of the Web3D Consortium. X3D has an XML encoding and allows users to download, display, and interact with 3D content via a web browser plug-in or a standalone viewer. X3D is the successor of the Virtual Reality Modeling Language (VRML). The forms are described by geometrical shapes, and behaviors can be controlled from the scene, both internally and externally, by the X3D file for programming languages or script.

Figure1: VisAr3D Supporting System Overview.
Xj3D is the Web3D Consortium’s open source project that provides a toolkit for writing X3D content developed in Java. By combining Xj3D and Java, one can easily add 3D capabilities to any application.

VisAr3d is composed of three main modules (Figure 1): Architectural Module, Augmented Reality Module and Virtual Reality Module.

On a regular course, the teacher exhibits a large and complex system diagram projected on the wall or on a printed folder. It details the features, components and connectors, in different levels of abstraction, possibly using various architectural styles. Before using VisAr3D, the software model in study must had been created and documented within a software model editor. On the Architectural Module, the diagrams must had been built using an external UML editor and exported as an XMI file. VisAr3D is able to read this stored UML diagram.

On the Augmented Reality Module, VisAr3D uses the 3D technology to capture and recognize the 2D projection (teacher’s diagram), helping the teacher and students in identifying and quickly accessing the model. By using the ARToolKit libraries [20], a graphic pattern is captured by the webcam into the student’s computer. VisAr3D uses this feature to identify the model that is being studied. The Augmented Reality Module turns the object of study more accessible to the user and integrates it in the classroom activities. Besides being easily usable, it ensures the display of models in the latest version.

After recognition of the software diagram displayed in 2D, the Virtual Reality Module automatically generates an equivalent 3D UML diagram from it. The resulted diagram is displayed in an unlimited space with a slight depth.

The minimum resources to support VisAr3D are a datashow to project the diagram on the wall or on a printed paper with a graphic pattern, and a mobile device with webcam, connected to a computer network.
VisAr3D Functionalities

VisAr3D, as a virtual environment, should enable the exploitation of the environment by the student with the help of a mouse, moving in space to the right, left, up, down, stay away or get closer to the plane and rotational movement.

The student should be able to visualize the 3D diagram in different angles and distances with the aid of icons and colors. VisAr3D must also allow students to get closer and realize the presence of more contextual information, and to go away until this information disappears. This feature allows the analysis of software models from a new perspective, and thus it is possible to discover similarities between parts of that diagram, to understand more complex relationships, different techniques, and architectural styles.

The amply space for data exploration in a 3D visualization environment facilitates the comprehension of diagrams with different levels of abstractions, or versions that have been designed by different people.

The functionality of viewpoint facilitates the movement of students at pre-defined points of interest. The teacher or the student can select (define) a position in the diagram as a new viewpoint and the user will be able to quickly move between these points, even when they are far away in the diagram.

Another facility provided by VisAr3D is a search agent that allows looking for documents by keywords or filters. The result can be a link list or a graphical visualization through colors or flags. This sub-system allows quickly and easily finding and accessing all kinds of information associated to the software model in study.

Zooming technique is also available, providing more details of the data shown on a certain view.

Besides allowing the navigation through all diagrams, there are defined views in the context of this work that represent different forms of visualization, intending to improve the quality of the course, in line with what is stated in [21]. VisAr3D uses
diagram views to visualize the system from different perspectives. No complex system can be understood in its entirety from only one perspective. The main idea is that different aspects of the system can be independently focused by choosing the right set of views. These views are:

“Relationships with Other Diagrams” View: One class in a model diagram can be presented in several diagrams. It is often difficult and tedious to find out on which place a class is in the model, because this relation is not explicitly presented in the diagram. To fully understand this class element, it might be necessary to explore this modeling element: its attributes, methods, and associations that can be found in different diagrams. In this kind of visualization, the selected class element is centered on the screen as the user changes from one diagram to another. It is well-known that the comprehensibility of a system is mainly determined by the possibility of studying the system one part at a time [22]. Since the part of the system that is relevant for the user purpose is shown, the target of the learning process is greatly reduced [23].

Relationships with Other Types of Diagrams” View: This view highlights the modeling elements that belong to other types of diagrams using colored links. For example, the class "Rule" in a class diagram can also be explored and studied in a sequence diagram. The user will access this same class in another type of diagram by just clicking a link.

Package View: Package View gathers the modeling elements in one Package. If this package diagram is available, the user can access this level of abstraction by just pressing a key.

Metric View: VisAr3D combines the existing layout of UML class diagrams with the visualization of metrics. Since applying metrics to a UML model can result in an overwhelming amount of data, and these data are usually presented in tables, it is a hard work for a software engineer or a student to make the mapping between metrics values in a table and classes in UML diagrams manually. VisAr3D supports this activity by integrating the model and metric visualization, using color, size and shape to visualize values.
Attributes/Operations View: There is information that only appears when requested by the user through the Attributes/Operations View, to avoid pollution of information associated with large diagrams attributes, methods, and names of relationships. When the user moves the cursor over a modeling element, its attributes, methods and relationships names are displayed in a readable size.

Author View: Likewise, VisAr3D allows visualization of the authorship of the diagram. In the distance, colorful icons indicate the authors of the architectural elements. Figure 2 shows the Author view.

Documentation View: The development of an architectural design often produces a large amount of documentation, as well as records of the initial decisions about the project to ease the communication between stakeholders. This information is very important for software modeling students, especially for the novice ones, to understand the whole process of creation and software development. By using VisAr3D, 3D architectures are automatically generated from 2D and displayed with contextualized information, containing a multimedia documentation in audio, image and video forms. While navigating through the diagram, students will have access to this superimposed information, as they move through the model space. It should be easy to perceive its correspondence to its content, identified by icons. As the user chooses to read one document, more than one modeling element, associated with it, is marked.

This feature is interesting for displaying information associated with graphic objects, which support the classroom, help student’s understanding or constitute a rich material for discussion. They also become essential for the study of large volumes of scattered data on the network. They can be: descriptions of architectural details, notes of a meeting, indicators of any kind, sketch diagrams, messages between users to discuss about modeling, assembly video, audio reports or any documentation that contributes to the student’s learning about that architecture. In many cases, data such as these do exist, but may be loose or lost, without being associated to the architectural model. This is a good way to retrieve relevant
information that in other way would be lost in time, and make them available for use. The intention here is to value good documentation and encourage students to always document.

**Figure 2: VisAr3D prototype screenshot.**

**Annotation View:** VisAr3D contains information generated within the space. It serves to facilitate the understanding of software model, and consequently better communicates information among team members who build the system. Through spontaneous and sometimes even playful situations, a student can explore, create, and especially communicate with other students or the teacher to obtain new information and to build his/her own learning process by sharing findings, doubts, or simple messages using notes. The participants can add arbitrary annotations to particular artifacts.

**Exercise View:** This view allows the teacher to interact with students through exercises that can improve their theory knowledge. This view is similar to the annotation and documentation views but is categorized as exercises. The students can be asked to choose between design alternatives, be required to make choices to address customer needs, be asked to reuse components, to adhere to
standards and also to correct erroneous elements, be encouraged to compete in using "Case Studies", or be challenged to refactor a piece of architecture.

**Final Considerations**

It is well-known that visual presentations can facilitate the understanding of complex systems. For this reason, this paper presented VisAr3D, a Software Architecture Visualization in 3D. Several important aspects were described to exploit the advantages of the 3D space to support the understanding of complex systems through a more attractive experience for the user.

VisAr3D proposes to meet these challenges by combining features of Virtual Reality and Augmented Reality. It is a novel way to visualize and understand UML models of a complex system. With this approach both teachers and students have a great opportunity to explore a 3D environment that emphasizes the intuitive understanding in a didactical context. There is a focus on the need of a practical experience (students feel greater motivation to see practical results in operation), bridging the gap between practice and theory; on the construction of a collective reasoning to understand concrete situations; and on the investment in the application of graphical interfaces and documentations to encourage students and to reduce the learning curve.

The approach lets users explore the various views without forcing them to follow a specified path. This should allow for more creativity and therefore interesting observations. They can find errors, examine them and learn through them.

To make the software modeling discipline more interesting and stimulating, VisAr3D aims at regaining the students’ taste for mastering the knowledge in a pleasant way. The idea is to bring a greater dynamism to the classroom, resulting in a more meaningful participation of students, mainly applicable to a large-scale software system.
An experimental study was planned and conducted to evaluate the feasibility of the VisAr3D approach and provided positive evidences of the benefits of its educational support [24].

References


**Sobre as Autoras**

**Claudia Susie C. Rodrigues**


**Claudia Maria Lima Werner**

É bacharel Em Matemática Modalidade Informática pela Universidade Federal do Rio de Janeiro (1985) e doutora em Engenharia de Sistemas e Computação pela COPPE/Universidade Federal do Rio de Janeiro (1992). Atualmente é professora associada da Universidade Federal do Rio de Janeiro. Tem experiência na área de Ciência da Computação, com ênfase em Engenharia de Software, atuando principalmente nos seguintes temas: desenvolvimento baseado em componentes, orientação a objetos, reutilização de software e engenharia de domínio. E-mail: werner@cos.ufrj.br