Implementing Systems Analysis / Systems Design Course Projects Using Rational Rose, the UML, and Visual Basic

Robert B. Sweeney, Jr.[1]
School of Computer Information Sciences, University of South Alabama
Mobile, AL 36688 USA

Abstract

There are numerous benefits derived from the use of visual modeling in object-oriented analysis and design. These include facilitating communication between clients and developers, managing complexity, promulgating a shared vision of the project's goals and objectives, and creating and maintaining an artifact upon which to enhance the current application or use to build future applications. Computer-aided software engineering tools incorporating the Unified Modeling Language, such as Rational Rose from Rational Software Corporation can automate and improve the development of object-oriented visual models. Rational Rose can be obtained without cost by qualifying educational institutions from Rational Software if they participate in the company's SEED program. The purpose of this paper is to review the utilization of the UML, Rational Rose , and Microsoft Visual Basic for project development in a systems analysis and systems design course sequence which has an object-oriented emphasis.

Keywords: Unified modeling language**,** UML, Rational Rose, visual modeling, systems analysis, systems design, Microsoft Visual Basic

1. INTRODUCTION

Information Systems educators are charged with implementing the IS '97 model curriculum guidelines. The guidelines outlined for the Analysis and Logical Design course promote the idea that students should develop the ability, "to show how to develop a logical design, and develop and analyze alternatives involving implementation using packages, tailoring of packages, constructing software, or CASE tools (Davis 1997)." The guidelines for the DBMS Physical Design and Implementation course recommend that students should be able, "to develop skills with use of a combination of code generators and language facilities to implement multi-user departmental or simple enterprise level systems (Davis 1997)." Both of these sets of guidelines have application to courses in systems analysis and/or systems design. As a visual modeling tool complete with code generating capabilities and multiple language support, Rational Rose provides the means to help achieve these goals.

This paper reviews an attempt to integrate visual modeling and application development using the Unified Modeling Language (UML) and Microsoft Visual Basic into a Systems Analysis / Systems Design course sequence. The catalog description of the Systems Analysis course is, "A continuation of ISC 350 (Information Systems in Organizations) with emphasis on object-oriented analysis methodology," while the description of the Systems Design course is "A continuation of ISC 360 (Systems Analysis) with an emphasis on object-oriented design methodology." The topics covered in both courses include materials related to developing systems using the system development life cycle while at the same time incorporating material related to object-oriented analysis and design. Students generally take both courses from one of two instructors but they are not required to do so.

The courses must be taken in sequence but otherwise the prerequisites for both courses are the same including a number of lower division courses known collectively as the Professional Component. Professional Component includes courses in English, logic, statistics, math, business, software engineering, file structures, and two courses in Java programming. In addition, specific course prerequisite requirements include data communications, operating systems, and database design (in the case of Systems Design). Students are introduced to the major concepts of object-oriented programming such as inheritance, encapsulation, abstraction, polymorphism in the Java programming courses.

[1] sweeney@cis.usouthal.edu

2. RATIONAL ROSE AS A CASE TOOL

Rational Rose visual modeling facilitates and automates parts of the software development process through the use of the Unified Modeling Language. The UML is meta-language receiving strong industry support that specifies a standardized set of graphical notations and their syntax that is useful in object-oriented analysis and design (OOAD).

The Rational Software Corporation offers a Software Engineering Educational Development (SEED) program that provides copies of various software packages and training materials, including Rational Rose, to qualifying educational institutions. Information about the SEED program may be obtained from seed@rational.com or from the company web site at http://www.rational.com. The SEED program can provide both individual 1-year licenses that can be provided to both students and faculty to use the Rational Rose software as well as server licenses so that the software can be installed in a laboratory environment. In addition to Rational Rose, many other software tools are also available under the SEED program such as those for configuration management or real-time system development. The SEED program can also make available training materials for various courses related to OOAD, UML, and Rational Rose (Sweeney 2000).

Rational Rose has many advantages as a UML visual modeling tool, but its flexibility may be the most valuable. This flexibility is achieved by the software's ability to support many object-oriented languages including Java, MFC C++, Visual Basic, as well as Oracle 8i databases. Rational Rose promotes software component reuse and better utilizes scarce software development resources. Rational Rose supports both forward and reverse engineering. Support for forward engineering means that Rational Rose can generate code based on a visual model of the application, whereas reverse engineering support allows Rational Rose to convert existing code into a visual model. The combination of forward and reverse engineering is often referred to as round-trip engineering.

Successful OOAD processes are often described as having the characteristics of being both "iterative" and "incremental" (Fowler 2000; Reed 2000; Liberty 1998). In the context of OOAD, iterative means that the development proceeds piecemeal toward completion, in such a way that as Fowler notes, "each iteration builds production-quality software, tested and integrated, that satisfies a subset of the requirements of the project (Fowler 2000)". Each iteration builds incrementally on previous iterations during the construction phase of the project. Rational Rose supports this iterative/incremental development approach in part through its round-trip engineering capabilities and by virtue of its ability to be used in a team development environment.

3. USE OF RATIONAL ROSE IN A SYSTEMS ANALYSIS AND SYSTEMS DESIGN COURSE SEQUENCE

Rational Rose has been utilized in a junior-level Systems Analysis and a junior-level Systems Design course at the University of (name deleted). Both classes used the same primary textbook, Modern Systems Analysis and Design (Hoffer 1999), that included a chapter on the UML. However, since adoption of the textbook, the UML has undergone some revision and as a result this textbook alone was inadequate. Consequently, another, highly recommended text, UML Distilled (Fowler 2000) was used to supplement the Hoffer textbook and provide a more in-depth, yet still accessible, source of information about the UML.

UML Distilled was particularly valuable as a course textbook as it introduces the students to a methodology, based on the Rational Corporation's proprietary Unified Process, for performing object-oriented analysis and design. A fundamental concept of the UML is that it is not a methodology or process, but in fact can be used with multiple processes. The methodology introduced in UML Distilled involves four phases: Inception, Elaboration, Construction, Transition.

Inception involves activities that ensure that the customer and developer have a shared understanding of the objectives (business case), architecture, and scope for the project. In addition, the use cases are identified and the key ones are described. During Elaboration further examination of the project domain results in project refinement as the remaining use cases are described and a project plan is developed including an identification of the high risk project elements. In Construction, integration and development of program features occurs, program documentation is completed and software testing is performed. Finally, Transition results in the transfer of the application from the developers to the users and involves user testing, system conversion, system implementation, and training.

Selection of an appropriate methodology is critical to the success of an OOAD effort as is understanding the activities involved in that methodology. The UML can support this chosen methodology during the development process identifying, clarifying and capturing design analysis and decisions. In addition to choice of and appropriate utilization of an OOAD methodology, factors that affect the development process include the utilization of software design patterns. These are named, codified, reusable software "best practice" design concepts. Design patterns offer developers an efficient way to communicate and can be a valuable addition to the OOAD process.

Like many of OOAD methodologies, these phases are performed both incrementally and iteratively. One aspect of incremental development is that the project are often completed in small, independent pieces where the focus is often on the completion of the functionality of one use case at a time, usually beginning with what is considered to be the riskiest use case. Iterative development means that the Inception, Elaboration, Construction, and Transition phases are repeatedly performed for each piece of functionality completed with each iteration resulting in a deliverable, preferably an executable one. Iterative and incremental development allows for many advantages including the ability to effectively and efficiently respond when, almost inevitably, project requirements change during development.

Course projects were used in both the Systems Analysis and Systems Design courses to enable to students to practice OOAD development techniques. These projects are designed to expose students to the steps performed in the phases of OOAD as outlined above. The students analyzed a case study of a business situation and developed use case scenarios and diagrams in order to identify and understand the basic functionality and user interactions of a project. Students make a presentation of their development artifacts in both classes. Optimally, a course project that began in the Systems Analysis course could be extended and completed in the Systems Design course, however, due to the fact that students can take the course sequence from one of two instructors who do not have a standardized curriculum with regard to the use of class projects this is not currently done.

Students in the Systems Analysis course used the Rational Rose software to work a number of smaller problems before completing a larger, end-of-semester, project involving multiple diagrams and models. This project utilized a case study of a small business (a video rental store). Students prepared a complete set of diagrams to model an application for handling the store's inventory. This project was performed as a team assignment. Initially, each team met and decided on appropriate uses cases and wrote use case scenarios. As part of the development of their use cases, students defined the actors that interact with each use case, the preconditions that must exist before the use case can be initiated and the postconditions that must be true before the use case can end, and any alternative or exceptional pathways through the use case.

After development of the use cases and use case scenarios, sequence diagrams were created to model the object interactions present in each use case scenario. In addition, students developed conceptual class diagrams to document the static structure and relationships of their classes and used packages to divide the class diagrams into various system areas. Finally, the students created state diagrams to document the behavior and states of particularly complex or volatile objects.

One problem encountered during the course of this project was that students on the same attempted to build separate Rational Rose models with the intention of later combining them by utilizing Rational's *Rose Model Integrator* software tool. This application is designed to compare and / or merge model elements from different models so that differences between the models are noted and can be resolved. Using the *Rose Model Integrator* proved to be less than successful and in most cases, students ended up manually combining model elements from separate models onto a new or existing Rose model.

In the Systems Design class, the focus shifted to the development of an actual working system, albeit on a small scale. The idea was to give the students a chance to see how Rational Rose can actually be used to perform round-trip engineering. Round-trip engineering is the ability to create code from a visual UML model and also to create a visual model from existing source code. This feature greatly enhances Rational Rose's ability to support an iterative and incremental development process. Using Rational Rose, the students first proposed the design of a new system by writing a simple use case and a complete use case including the actors, preconditions, postconditions, etc. as defined in the previous paragraph. The instructions to students were to develop a use case and use case scenario describing a typical business application function. Examples of sample business function use cases provided to students included "placing an order", "maintaining inventory," "shipping an order," and "modifying an order."

On satisfactory completion of their use case scenarios, students then prepared at least one sequence diagram to show where messages were passed between objects as described in their use case scenario. For complex use case scenarios, such as those with many alternative pathways, the students prepared additional sequence diagrams as needed. Sequence diagrams provide a means for initially identifying the classes that make up the system, so in the next step, an initial class diagram was developed and relationships between classes identified through development of the sequence diagram(s) were modeled.

Microsoft Visual Basic version 6 (VB6) was selected as the programming language to be used for implementation. Several factors contributed to the decision to use VB6. One was the fact that, based on a poll of students in the systems design class, this was the language with the highest degree of familiarity within the class. Although Java is the language currently used in the school's required core programming courses, several students had taken those core courses prior to the use of Java or had transferred to the school with credit in another object-oriented language. Another factor was that part of the course content for the systems design class is the study of appropriate interface design, and VB6 is well-known and respected for its ease of use and functionality in this area.

At this point it is worth noting that the Rational Rose application is daunting to many students (and instructors) at first exposure due to its large number of features and options. However, the documentation CD-ROM (Rational 2000) that comes with the software includes a number of tutorials that are extremely useful when first learning the how to use the application. Particularly valuable is the "Rose Visual Basic" tutorial accessed via Rational Rose's Help menu. This is a complete model along with an associated Visual Basic project. Model elements can be related to code in the project and vice versa in order to see how one relates to the other providing a powerful teaching tool. Following the examples provided in the Rose Visual Basic tutorial which is provided with the Rational Rose documentation and in the book, Developing Applications with Visual Basic and UML (Reed 2000), the three-tiered class diagram option was selected. This requires going to the Tools menu, Options command, Diagram tab and selecting Three-Tiered Diagram from the Diagram section then restarting Rational Rose.

The Rational Rose help section states, "The Three-Tiered Service Model diagram supports the three-tiered architectural approach used when building Microsoft Visual Basic applications, as it separates the contents of the system into three conceptual tiers of services: *User Services* (left pane), *Business Services* (middle pane), and *Data Services* (right pane). It is a special kind of class diagram, called three-tiered diagrams." User Services are those classes used to represent interfaces for presenting and accepting data. Business Services are those classes that represent business domain entities. Data Services are for those classes used to provide persistence (storage and retrieval) of data as well as access to a DBMS. This delineation of classes categories into the Microsoft vision for software development and therefore Visual Basic is an excellent match for its use (Reed 2000). Once an initial class diagram was created, students utilized Rational Rose's forward and reverse engineering services. Access to these services is through the *Tools* menu, *Visual Basic* submenu, and the *Update Code* and *Update Model from Code* commands, respectively.

For particularly complex objects, those that for example have a number of changes in their state over their lifetime, students prepared state diagrams. These diagrams are useful in determining both the attributes that define an object and determine the state it is currently in and the operations the object can perform with in, or transitioning to, a state. This information in turn is used to develop and refine the classes that are documented on the projects class diagram(s).

In our simple example, interfaces defined in the User Services area of the class diagram are mapped to only to VB6 forms. Accordingly, at this point students, having modeled in Rational Rose an initial class diagram and forward engineered this model into a VB6 project outline, can begin to design the various forms, reports, and dialogs necessary for their proposed application. Even though VB6's competence in this area is strong, the students were asked to create a sample source document that would be used in the data entry process for their use case. They then prepared hand-drawn storyboard-type renditions of the interfaces they were proposing prior to actually creating them in VB6. The purpose for developing these initial drawings is to have the students think about the importance of matching a user's mental models of how an interface should look with the actual implementation of that interface. This part of the project also tied in well with material from the Hoffer textbook on design and development of forms, reports, and dialogues, particularly the guidelines presented which can be used as a checklist for developing and testing these interfaces.

After completion of the preliminary interface design, the project iterated between adding code to the VB6 project to implement the various operations defined for each class. Attention was now also turned to the problem of implementing data persistence so objects created can, if necessary, be stored and retrieved. Due to the complexity involved in attempting database connectivity and the fact that the systems design course was offered over the shortened summer semester, this aspect of project implementation was omitted. In addition, some students lack of prior programming experience with VB6 caused them to have difficulty developing code beyond that created by the Rational Rose code generator. Students iterated between developing code and documenting the code in the visual model until a working prototype was complete. Simple testing procedures were performed and project documentation finalized. Students completed the project by demonstrating their application to the class.

Class discussion at this point was directed toward the topic of incrementing additional use case functionality into the existing project. In a "real world" situation, development often starts by analyzing the risk associated with the various uses cases identified and then attacking the riskiest first. After that use case functionality has been produced and tested the next riskiest use case is developed, integrated with existing project components, before unit and system tests is performed. Students were asked to develop a second use case, related to the first and consider how it could be developed and integrated into their existing project.

4. CONCLUSIONS

One of the goals of this two course sequence in Systems Analysis and Systems Design is to describe several major alternative methodologies for developing information systems. Toward that goal, students are introduced to a variety of methodologies including those associated with OOAD. Students are instructed in the standardized and widely-used UML metalanguage that is useful for identifying and documenting project analysis and design decisions. The leading software application for utilizing the UML, Rational Rose, is also introduced. Furthermore, since the UML needs to be used in the context of an overall development methodology, an iterative and increment series of development phases is introduced. Student projects in both classes focus on developing student skills and confidence relative to leveraging the UML and its associated software to improve the software development process.

Quality of the projects submitted by students as well as the instructor's perception of the student's level of understanding of the concepts, however, varied widely between students. This indicates that not all students were equally adept at mastering the UML concepts that were presented or the software tools, such as Rational Rose and Microsoft Visual Basic, that were available. Problems noted previously regarding the degree of knowledge that students in the classes had with VB6 and the difficult task of learning a complex modeling language and a complex software application also contributed to this variance, in this author's opinion. Generally, however, comments from students were good regarding the use of Rational Rose and, in the case of the systems design course, VB6 in the course projects.

5. REFERENCES

Davis, Gordon B., Gorgone, John T., Couger, J. Daniel, Feinstein, David, L., Longenecker, Herbert E., 1997, IS '97: Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems. Association of Information Technology Professionals.

Fowler, Martin with Kendall Scott, 2000, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2nd Edition. Addison Wesley Longman, Inc., Reading, MA.

Hoffer, Jeffrey A., Joey F. George, and Joseph S. Valacich, 1999, Modern Systems Analysis and Design, 2nd Edition. Addison Wesley Longman, Inc., Reading, MA.

Liberty, Jesse, 1998, Beginning Object-Oriented Analysis and Design. Wrox Press Ltd., Birmingham, UK.

Rational Software Corporation, 1999, Inside the Unified Modeling Language. CD-ROM.

Rational Software Corporation, 2000, Rational Solutions for Windows: Online Documentation, Version 2001.03.00, November 2000. CD-ROM.

Reed, Paul R., 2000, Developing Applications with Visual Basic and UML. Addison Wesley Longman, Inc., Reading, MA.

Russell, Jack, 1999, "A Second Course in Systems Analysis and Design: A Rationale and a Proposed Course Outline." Proceedings of ISECON '99, October 14-17, pp. 84-89.

Sturm, Jake, 1999, Visual Basic 6 UML Design and Development, Wrox Press Ltd, Birmingham, UK.

Sweeney, Robert, 2000, "Utilizing the Rational Rose OOAD CASE tool for Visual Modeling using the UML in the Systems Analysis and Design Sequence." Proceedings of ISECON '00, November 9-11.