A Process for Modeling the Analysis of Information Systems with the Unified Modeling Language

Joseph H. Daniel[21]
Computer Information Systems Management
Delaware Valley College, Doylestown, PA 18901 USA

Abstract

A two-phased process for modeling the analysis of an information system by means of the Unified Modeling Language has been developed from over two years of experience with a Systems Analysis and Design course. The first phase of this process, Requirements Analysis, involves the application of a Use Case Diagram and a Sequence Diagram. The second phase of this process, Domain Analysis, includes the application of a Class Diagram and a State Transition Diagram. This process is still evolving and, hopefully, will continuously improve in the future.

Keywords: Use case diagram, sequence diagram, class diagram, state transition diagram

1. INTRODUCTION

Currently, no standardized process exits for the modeling of information systems by means of the object-oriented Unified Modeling Language (UML). UML is not a process in itself, but primarily a graphical notation that is used to express the designs of an information system (Fowler 1997).

In spite of this limitation, I have developed a two-phased process for modeling the analysis of an information system as a result of over two years of teaching an undergraduate course in Systems Analysis and Design.

Throughout the two phases of requirements analysis and domain analysis, I have used a College Course Registration System as an example for the student assignments. The students relate well to this information system because they use it several times during the academic year.

2. MODELING PROCESS

The first phase of this process, Requirements Analysis, involves the application of a Use Case Diagram and a Sequence Diagram for analyzing the requirements of the information system. The purpose of this analysis is to create a better understanding of what the users want from the new system (Eriksson 1998).

The second phase of this process, Domain Analysis, includes the application of a Class Diagram and a State Transition Diagram for analyzing the domain of the information system. The purpose of this analysis is to achieve a better understanding about the world that the new information system is supporting (Eriksson 1998).

The four UML diagrams mentioned above are drawn by the students with the "AutoShapes" feature in the WORD 97 software. WORD was chosen over the two software engineering tools mentioned below because it was easy to use and readily available at a low cost.

The Object-Oriented Software Engineering package, which is free on the Internet for a demo version from the Rational Rose Company, was not used because it is limited to 10 use cases in the Use Case Diagram and will not draw a State Transition Diagram.

The Visible Analyst Workbench software, which is $99 for a student version from the Visible Systems

[21] danielj@devalcol.edu

Company, was not used because it will not draw a Use Case Diagram nor draw a Sequence Diagram.

3. REQUIREMENTS ANALYSIS

In the Use Case Diagram of Figure #1, the students are required to have 5-10 actors (stick persons), 10-15 use cases (ovals), and 2-3 extends (ovals). An actor, such as a Student, is an object that interacts with the Course Registration System, but is not a component of this information system.

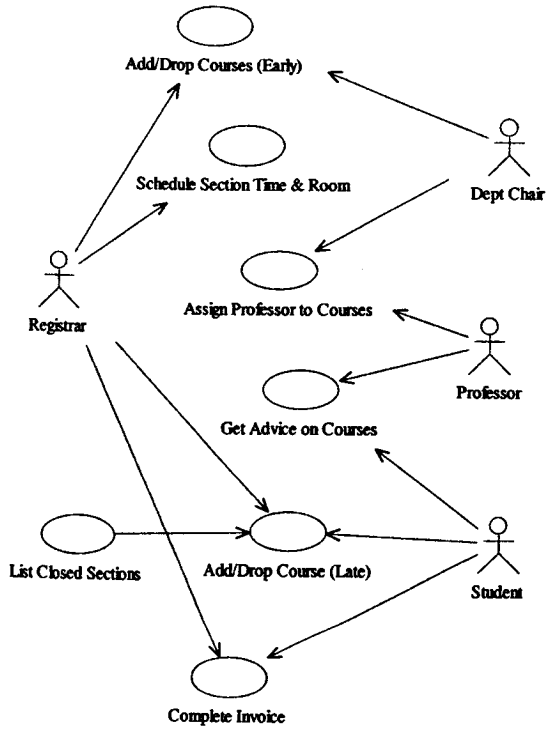


Figure # 1 - USE CASE DIAGRAM

This actor initiates a method or a use case, such as the adding or dropping college courses, in order to yield a measurable result. Interaction between two use cases is called an "extend", such as the listing of the closed sections for a course.

The Use Case Diagram concentrates on many methods, whereas the Sequence Diagram takes only one of these methods and looks at the interaction among many objects. An object, such as a Course Schedule, is a set of people, places, or things that performs common methods and shares common data attributes.

In the Sequence Diagram of Figure #2, the students are required to have one actor, one use case, 5-7 objects (rectangles), 5-7 time lines (vertical lines), 10-15 messages (straight arrows), and 2-3 self-messages (backward arrows). A time line represents one object, and the distance between two time lines represents the start and stop time between two objects during their interaction.

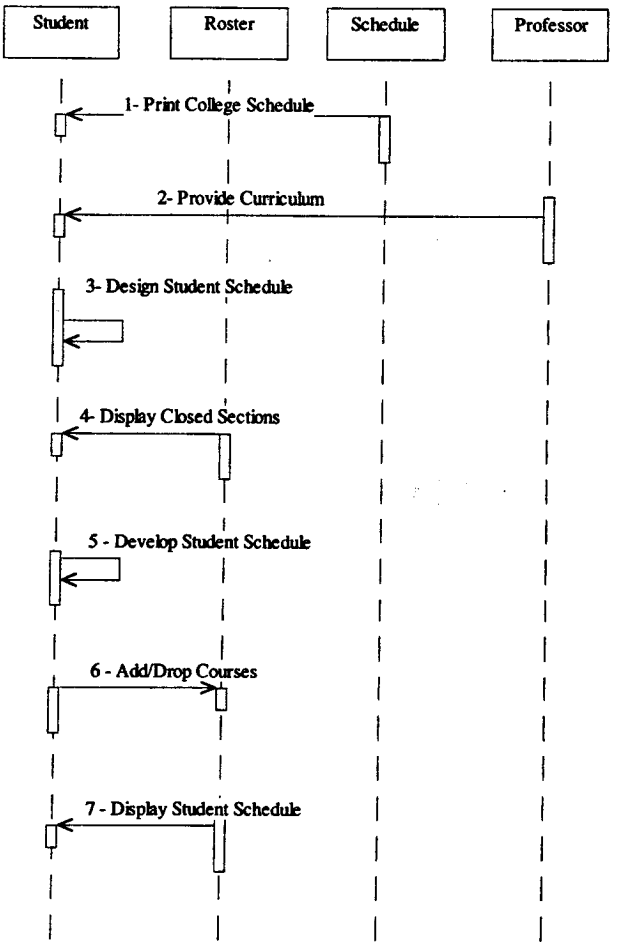


Figure # 2 - SEQUENCE DIAGRAM

This interaction between two objects is represented by a message, such as the printing of a Course Schedule for a Student. A self-message, on the other

hand, is an interaction that an object performs upon itself, such as the designing of a student schedule by the Student.

Briefly, the first phase captures and describes the user requirements for the new information system. This phase documents the requirements by a Use Case Diagram of many methods and many Sequence Diagrams of one method with many of its objects.

4. DOMAIN ANALYSIS

In conjunction with the first phase, the second phase defines the real world environment or domain, such as a College, which uses a Course Registration System. This domain is documented by means of a Class Diagram of many objects and State Transition Diagrams of one object with many of its methods.

For the Class Diagram of Figure #3, the students are required to include 10-15 classes (rectangles), 5-10 associations (straight arrows), 1-2 reflexives (backward arrows), 2-3 aggregations (diamond on arrows), and 2-3 inheritances (triangle on arrows). Each class in this diagram should contain about 2-3 data attributes and 1-2 methods.

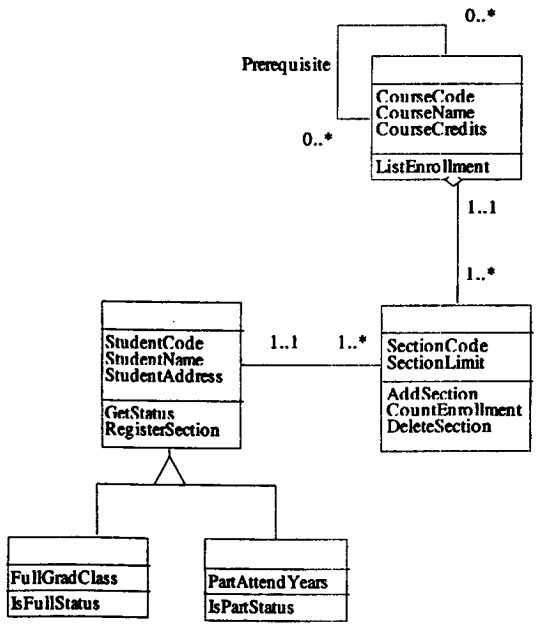


Figure # 3 - CLASS DIAGRAM

A class, such as a College Course, represents a group of objects, which contains instances of this class, such as the college courses of CM 2114, CM 3103, and CM 4146. When comparing the object oriented to the traditional systems approach, it is interesting to note that a class is analogous to a data file and an object is analogous to a data record.

The relationships in Figure #3, for example, involve (1) a bi-directional association between the Student and Section, (2) a reflexive relationship of the Course upon itself, (3) a "whole-part" aggregation among a Course and its component Sections, or (4) a "parent-child" inheritance where a Student is either Full-time or Part-time. Each of these relationships, except inheritance, has their multiplicity, such as one (1..1) Student is taking one or more (1..*) College Courses.

In the State Transition Diagram of Figure #4, the students are required to have one primary object, 15-20 events (arrows), 2-3 self-events (arrows), 10-15 states (rectangles), one start state (small circle), and one or more stop states (small double-circles).

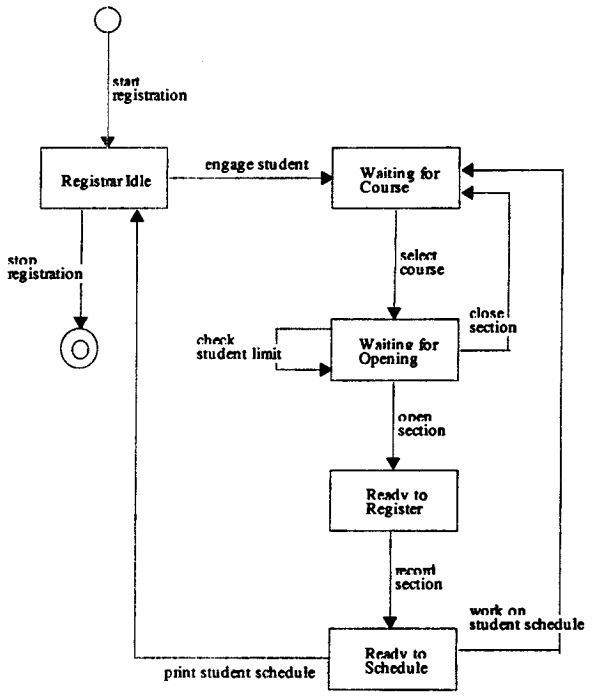


Figure # 4 - STATE TRANSITION DIAGRAM

An event represents a message sent between two objects, whereas a self-event is a self-message that an object performs upon itself. A message can also denote a request to perform the method of another object, such as the selection of a course from the Course Schedule. A state is the result of an event, such as waiting for a course to have an opening.

It is important to mention that the second phase is not a detailed design of the entire domain for the information system. For the student of a Systems Analysis and Design course, this phase, for example, should define only about 80 percent of the most significant objects and methods in the Course Registration System.

5. MODEL EXTENSIONS

In the first phase of this UML modeling process, the Use Case Diagram was used because it simplifies the traditional Data Flow Diagram (DFD) by including only the actor/end-user and use case/method in describing the requirements of an information system. The input documents, data files, and output reports are named only within the use case itself.

The Sequence Diagram was used because it details the Use Case Diagram by taking one use case/method apart to show the dynamics among its objects. Note that the Sequence Diagram is very useful in the early analysis phases of system development. As an extension to this diagram, a Collaboration Diagram may be used to display the same material, but with an overall design phase perspective.

In the second phase of this UML modeling process, the Class Diagram was used because it extends the traditional Entity Relationship Diagram (ERD) by including not only the data attributes, but also the methods of a class. As an extension to the Class Diagram, a Package Diagram may be used to organize classes into groups.

The State Transition Diagram was used because it details the Class Diagram by taking one object apart to explain the dynamics between its events/methods. As an extension to both the Sequence Diagram and the State Transition Diagram, an Activity Diagram may be used to illustrate the complex dynamic interaction among many objects and many methods simultaneously.

6. CONCLUSION

In an effort to make the assignments in the Systems Analysis and Design course more like the real world, an industry-based case study was also given to the student. The New National Bank case, for example, allows the student to learn more about the information processing inside a bank and its unique vocabulary. Other cases in wholesale distribution, manufacturing, and hospital administration are planned for this course in the future (Vitalari 1995).

It is important to point out that it is not necessary to model the business processes before the analysis of an information system. However, this modeling would help provide a better understanding of the user requirements when creating the Use Case Diagram in a business domain (Bahrami 1999).

In conclusion, this two-phased process for modeling the analysis of an information system is still evolving and, hopefully, will continuously improve with each course in Systems Analysis and Design in the future.

7. REFERENCES

Ambler, Scott; Rosenberg, Doug; and Fowler, Martin.
1998. "Focus on UML." Software Development. v6 n3 pSR1-SR22.

Bahrami, Ali. 1999. Object Oriented Systems Development. Boston, MA: Irwin/McGraw-Hill.

Eriksson, Hans-Erik, and Penker, Magnus. 1998. UML Toolkit. New York, NY: Wiley Publishing.

Fowler, Martin. 1997. UML Distilled: Applying the Standard Object Modeling Language. Reading, MA:
Addison-Wesley.

Melewski, Deborah. 1998. "UML Gains Ground." Application Development Trends. v5 n10 p34-44.

O'Brien, Larry. 1997. "Rational Rose 4.0 for C++." Software Development. v5 n6 p17-22.

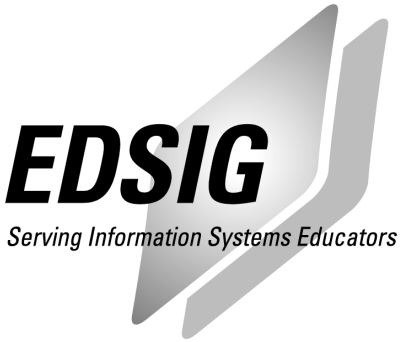
Quatrani, Terry. 1998. Visual Modeling with Rational Rose and UML. Reading, MA: Addison-Wesley.

Reed, Paul. 1998. "The Unified Modeling Language Takes Shape." DBMS. v11 n8 p46-52.

Shepherd, George. 1998. "When UML Meets FMC." Software Development. v6 n10 p51-56.

Vitalari, Nicholas, and Wetherbe, James. 1995. Cases in Systems Analysis and Design. Minneapolis, MN: West Publishing.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096