Course Outline
CS371S
Object-oriented Software Development

Instructor - J.C. Browne
Spring 2004

Course Approach and Goal

This course will introduce a model of software system development where an executable program is derived directly from an executable specification called an analysis model. No "code" is written except for a reusable software architecture.

The steps in the development cycle are:
a) The system is defined as an executable specification which is an object-oriented analysis model.
b) The program is validated at the analysis model level.
c) A software and execution architecture is defined as a set of class templates in an object-oriented programming system.
d) The executable system is realized by compilation of the validated analysis model to the software execution architecture.

This method of software development is now being used for high-reliability long-lived systems by leading embedded systems vendors such as Motorola, Xerox and Kodak.

Course Materials

The course materials include a textbook and some supplementary materials including papers and software system manuals.

The text is "Executable UML" A Foundation for Model-Driven Architecture" by S.J. Mellor and M.J. Balcer

The lecture notes will be distributed for the first few weeks and will then be available over the web and in hard copy through a distribution service.

Work Statement

This is essentially a laboratory class. The lectures will cover the xUML and the executable specification based development method in detail and other methods such as alternatives. The main goal of the course will be to carry through a complete development of a small software system using object-oriented development methods. There will be two class examinations but no final examination. The second class examination is scheduled for the last day of class. If there are objections to this date for the second class examination let me know by January 22, 2004.
The examinations are open-book and open-notes. Each student must take and pass both examinations. The course grade will be 50% on the project and 50% on the

examinations. Use will be made of commercial software tools which are used in industry.

Project Specifications

Project Structure - The project will be development of a small software system through the executable specification development methodology. The projects will be executed by small teams of co-workers. I have a set of possible projects. Each team will do a different project. A team can suggest a project of their own definition by preparing a requirements specification and getting it approved. The scale and scope should be similar to the requirements I will circulate.

Communication between Students and Instructor

The instructor is email oriented. He answers email almost every morning. This is by far the best way to communicate. Formulate your questions in writing and send them to browne@cs. Announcements concerning the class will be sent by email. Read your mail every day.

Standards for Conduct

Standard University of Texas rules for conduct of classes will be followed. Please make yourself familiar with those rules.

Appendix A – Analysis of Software Development Methods

Why Software Development by the Conventional Process is Difficult

Development of complex software systems has always been a challenging task. (We assume the reader is familiar with the conventional software development process based on manual translation of application designs to implementations in conventional procedural programming languages such a C or C++. The steady increase in functional complexity required for competitive capabilities in software products is compounded by implementation of these systems on distributed and networked systems. But the root causes of the problems of developing software by the convention process of manually programming application operations in procedural programming languages are:

(a) The wide conceptual gap between the operations defined in typical application domains and the operations defined in conventional programming languages. The results of this gap are: high complexity for the manual translations from application concepts to programming language concepts and high complexity and low readability of the programs which result. This conceptual gap also impedes validation.

(b) In the conventional development process end-user operations of the application are not validated until the programs in procedural programs are have been completed. When complex application operations are realized through complex transformations to complex programs in procedural programming languages, errors of translation are

inevitable and execution behaviors become unpredictable. The level of complexity of the program in the procedural language precludes detailed understanding of the entire system while at the same time the system must be validated in terms of application level operations. Additionally there is no provision for validation of the increasing important requirements for performance.

(c) Each software development project can make little use of artifacts from past development projects except at the lowest level of data structure library routines.

(d) Modifications of functional requirements expressed in application concepts must be done by modification of the procedural program in a different conceptual framework and at a much higher level of complexity.

(e) Modifications in execution environment lead to complex ports of the procedural program.

It is often difficult to estimate the effort necessary to realize products when the conventional process is used. The products often contain many defects due the difficulty of validation of the procedural program representations. The costs of modifications are high and error prone because they must be done on the procedural program.

It is apparent that a qualitative improvement in software development must automate translations from application specifications to procedural languages and that validation must be done in terms of application concepts.

Foundations of a New Paradigm

The preceding section makes it clear that qualitative improvement in software development process cannot be expected to arise in evolutionary enhancement based on the conventional process. And the analysis given in Section b identifies the steps in the development process which must be replaced.

(a) Manual translation from application operations to procedural programs must be automated.

(b) The application system must be validated in the conceptual basis of the application. Validation must include conformance to performance specifications as well as functional specifications.

The innovations which enable a process which removes these fundamental barriers are:

Separation of Concerns – Specifications for the application operations are done separately from specification of the execution environment.

Executable Specifications – The operations of the application are defined in a specification language with an executable semantics in the application conceptual domain.

Software Architectures – A software architecture is a specification of a set of operation and data structure templates to which the operations of an application can be translated. A software architecture is a virtual machine to which application level operations can readily be compiled. Execution environments for the applications are defined as software architectures in a standard procedural programming language.

Associative Objects – Associative objects are conventional objects extended and encapsulated to support automation of reuse and composition of systems from components. Associative interfaces replace and extend the concept of relationships in conventional object models.

The steps in the development process based upon these concepts are as follows:

(a) Requirements are captured in a traceable form in a database.
(b) An executable representation of the application system is captured in application concepts as an analysis level associative object model in which the constructs of the analysis model all have an operational semantics.
(c) A software architecture is selected or constructed.
(d) A simulation model of the execution environment is selected or constructed.
(e) The executable specification of the application is validated by execution over a workload (test suite) defined in terms of invocations of "end user" operations of the application.
(f) The functionally validated executable specification of the application is used as a workload generator for the simulation model of the execution environment to validate the performance behavior of the application operations.
(g) The fully validated executable specification of the application is translated to the software architecture by a compiler.
(h) The resulting program is compiled and linked by the standard compilers for the procedural languages to realize the production system.

This development process leads to a qualitative improvement over conventional development processes because it avoids the sources of difficulty listed in above. The characteristics of the new development process are:

(a) The application is validated in application concepts and with application based test cases. The immense complexity of validation of the procedural program is avoided.
(b) Modifications driven by changes in functional requirements are made to the executable representation of the application. Error-prone modifications of procedural program representations are avoided.
(c) Modifications driven by changes in execution environment are made to the software architecture and the simulation model of the execution environment. Production systems for new platforms are accomplished by compiling the application specification to the new software architecture. Error-prone ports of procedural programs to new platforms are avoided.
(d) Each software development task is accomplished by an appropriate expert. Application experts construct the executable specification of the application. Software designers develop the software architectures.

(e) Reuse of application level objects is enabled by defining the application objects as associative objects.
(f) The software architecture can often used for multiple projects with little or no change, particularly in the common case of a family of software products with similar characteristics.