Development of Object-Oriented Frameworks

Joseph W. Yoder

August 15, 1996

1 Overview

The primary work described here involves a "Business Modeling" application that is being developed and deployed at the Caterpillar, Inc. Caterpillar, Inc. joined the National Center for Supercomputing Applications at The University of Illinois as an Industrial Partner in December 1989 to support the educational function of the University and to use the University environment to research new and interesting technologies. During the partnership Caterpillar has initiated various projects, including the evaluation of supercomputers for analysis and the investigation of virtual reality as a design tool.

The most recent project is pilot project to demonstrate how an appropriate tool might support financial analysis and business decision making more effectively. This Business Modeling project aims to provide managers with a tool for making decisions about such aspects of the business as: financial decision making, market speculation, exchange rates prediction, engineering process modeling, and manufacturing methodologies.

It is very important that this tool be flexible, dynamic, and be able to evolve along with business needs. Therefore, it must be constructed in such a way so as to facilitate change. It must also be able to coexist and dynamically cope with a variety of other applications, systems, and services [Foote & Yoder 1995].

Many Frameworks have been developed to provide for a "model" that will help build a Business Model. The current work focuses on the Financial Reporting application and the frameworks developed to build them.

2 Decision Making During the Design Phase

When we developed the Caterpillar Business Model, we quickly realized that every business unit had its own variations on the way that they did business. Therefore, instead of trying to build a single model, we needed to build a set of frameworks for building models. When we started, however, we had no idea what these frameworks would look like. So, we started with a dynamic programming environment (VisualWorks) that would make it easy to change our programs [Goldberg & Robson 1983].

Many architectural decisions were the result of using VisualWorks. VisualWorks separates the user interface from the application objects, and also separates the DBMS from the application objects. But we discovered that we needed some reusable application objects, and we found that we needed to specialize the UI and the DBMS components.

First, we built a program to model one particular business unit. During the course of writing the program, we noticed many places where the same problem arose over and over. For example, the business model would make many queries to the database and would then display the results in tables. Both the queries and the tables would vary. VisualWorks has tools for generating both queries and tables, but both were more generic than we needed, and required too much programming. We built more specialized tools to make these tasks easier.

Then we built a financial model for another business unit. Most of the changes were ones we had anticipated. The new business unit had somewhat different databases and wanted a few new tables, so we had to change the queries and the tables. We also quickly found out that the use of the business model needed to include additional components such as: allowing the end-user to incorporate their business plan into the system, providing error-correction capabilities, and to allow the user to look at "What If" scenarios.

Because we were using Smalltalk, it wasn't hard to change the programs we had written. As we learned which parts of the program were most likely to change and encapsulated those parts in objects, we made the system easier to change. However, it still required programming to change the system. As long as we depended on inheritance to make new components then it always required programming to make a new one, but when we started to make new components by composing existing ones then we found that we could design "builders" that would compose them for us without using Smalltalk. This is important because Caterpillar has too many business units for us to build a business model for all of them. We will have to provide tools to let people make their own business models, which means that we will have to let people make their own business models without really knowing Smalltalk.

For the financial model frameworks, we saw we needed basic reporting capabilities with printing support, the ability to drill down to summary and detailed reports along with a means to do error correction and graph reports. This lead to developing frameworks for these. This paper will describe some of the Reporting Frameworks along with the SQL Formula Creator and Query Objects Framework.

3 Reporting Framework

Caterpillar uses a common profit/loss statement which has its calculations based upon the DuPont Model [Johnson & Kaplan 1987]. The view of this can be either a graphical way of looking at return on investment or the common spreadsheet format.

The views of the profit/loss statements was based upon the logic of looking at sales, costs, inventories, etc. that produced the desired results. All of the top level numbers such as sales and costs can be broken down to sales by region, by model, by date, etc.

The financial model application discussed here allow for users to view the top level numbers and let them drill down into different summary reports of their sales and costs along with means to view detailed transactions from the database and graph out the results.

A framework was developed for building the graphical view of the top level numbers by extending the VisualBuilder of ParcPlace's framework which will not be discussed here. We also built a graphing framework that mapped into the graphing framework provided by ParcPlace. This framework takes either 2-D lists of values to display or Query Objects discussed below and generates a customizable graph which allows the user to print the results. The details of this framework will also not be discussed here.

The basic reporting frameworks I want to discuss here is the development of the reporting framework for the generation of summary reports and detailed reports. The domain logic was defined using the Query Objects and Formula Objects framework [Brant & Yoder 1996].

All top-level reports have the ability to "drill-down" to a first level summary that viewed sales, costs, etc in a summary fashion. The user can then view more detailed summary reports down to the level of viewing and editing the individual transactions that comprise these reports.

The framework we developed provided for the development of ReportValues objects that contain the domain logic needed for all of the formulas and SQL operations for accessing the data of interest. These objects also contain the "meta-data" describing the first level drill-downs, other summary/detailed reports, and any graphs of interest.

ReportValues are table driven thus allowing for the creation of reports without doing any coding. Basically the database can be queried for the different types of reports that need to be built. This allows for all of the reports to be created at run-time based upon the "meta-data" stored in the database.

Since most reports needed were well defined and we found that the majority of the financial applications developed at Caterpillar can be handled by a collection of a few basic types of reports, we could easily automate this process and make it easy to develop an application through table driven information from the database. Thus once the required data model and query objects have been specified, a new financial application can be quickly developed.

4 Formula Creator & Query Objects Framework

Originally, the users of the system provided us with a list of errors that they normally search the database for. We hard-coded these queries and developed an interface for the user to select these errors. However, it quickly became apparent that each business unit would have a different list of these exceptions and scenarios and that these will probably even change over time within each business unit. Therefore we focused on a way to automate the creation of these during run-time.

Our solution to this was to build a dynamic system that will allow the user to decide at run-time the specific logic and variables of interest. This lead to the Architectural Design Decision of developing a framework for building and selecting queries of interest. Theses queries map into a SQL Database.

VisualWorks by ParcPlace provides a framework for creating static SQL database queries. The framework allows the developer to graphically create SQL queries that map to Oracle and Sybase Databases. These queries then get converted into a Smalltalk method that can be called later when desired. Smalltalk objects can also be passed into the generated methods and conversions and comparisons are supported by the framework. This framework can also query the database for the current data model the developer is interested in and then create objects to map to the desired tables within the database. It is also easy to extend the framework to add undeveloped database functions or extend the mapping to other database vendors.

The problem arises when one wants to dynamically change or create SQL queries during run time. Since the SQL framework supplied by ParcPlace only provides ways to predefined static queries we built a framework of SQL Query objects that let you create dynamic SQL queries through the use of Smalltalk expressions.

Our solution to allowing the dynamic creation of SQL objects was to define GroupQuery, OrderQuery, ProjectQuery, SelectionQuery, and TableQuery classes which are all subclasses of the QueryObject abstract class. We also created the associated objects to allow for the developer to build Smalltalk like query expressions.

The Query objects know how to respond to the appropriate message to build the queries and wrap constraints to themselves during run-time. See the Reports Patterns from the PLoP '96 proceedings for more information on the related patterns [Brant and Yoder, 1996].

We were able to reuse all of the code from the original VisualWorks framework of parsing for a method into SQL and sending the SQL across the network and then creating objects representing the desired values returned from the database. Our dynamic SQL framework permits for late binding of constraints to the SQL objects by allowing the developer to build a parser for developing queries and "wrap" additional constraints to SQL objects as the application runs.

The SQL Formula creator takes these QueryObjects and allows the user to dynamically add additional constraints and then file these out for later use. These constraints are wrapped around a selection criteria object. Thus, no matter what selection criteria the user may be viewing later, they can access the SQL Formulas that were previously created and get their desired results.

This allowed for the dynamic creation of error correction screens since the end-user can create the queries of interest and then can query the database given the current selection criteria and return the results for correction.

It was then easy to take the basic reporting schema that ParcPlace provided for viewing and editing data and build it dynamically given the QueryObjects specified from the end-user. We simply mapped the desired query into the framework that ParcPlace provided and then dynamically built a dataset for viewing/editing the data based upon the given QueryObject.

One thing to note is that the users of this system are always stuck with the same look and feel. Even though they could dynamically create any SQL query of interest, the returned data was always in a predefined format. Therefore, customization beyond our common look and feel would have required either extending the framework or "programming" custom screens for the user.

5 Conclusion

This workshop paper has briefly described a few frameworks included in the financial model being developed for the Caterpillar Business Model. Framework development was an interactive process for us as we first had to develop some working applications in order to see the common themes.

This work revealed a well defined reporting format used by the Caterpillar organization, thus allowing us to create frameworks for building the financial applications. These frameworks allow for the dynamic creation of the desired queries and views of interest as long as the end-user is ok with the default look and feel of summary reports, detailed reports, error correction modules, and graphing views.

6 References

[Goldberg & Robson 1983] Adele Goldberg and David Robson; Smalltalk-80: The Language and its Implementation, Addison-Wesley, Reading, MA, 1983.

[Brant & Yoder 1996] John Brant and Joseph Yoder; Reports, Third Conference on Pattern Languages of Programs (PLoP '96) Monticello, Illinois, September 1996 Pattern Languages of Program Design 3 edited by TBA. Addison-Wesley, 1997.

[Foote & Yoder 1995] Brian Foote and Joseph Yoder; Architecture, Evolution, and Metamorphosis, Second Conference on Pattern Languages of Programs (PLoP '95) Monticello, Illinois, September 1995 Pattern Languages of Program Design 2 edited by John Vlissides, James O. Coplein, and Norman L. Kerth. Addison-Wesley, 1996.

[Johnson & Kaplan 1987] H. Thomas Johnson and Robert S. Kaplan Relevance Lost: The Rise and Fall of Management Accounting, Harvard Business School Press Boston, MA, 1987.

[Gamma et. al 1995] Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading, MA, 1995.

7 Biography

I graduated with High Distinction and Honors from The University of Iowa in Computer Science and Mathematics. Since then, I have completed a Master of CS degree in the study of problems of the computer-based recording of medical records at the University of Illinois. I primarily focused on the development of computer-based system for the collection of physical exam findings. This design of this system employs an object-oriented approach through the direct manipulation of graphical objects integrated with hypertext approaches and semantic networking to build a system that is more natural to the user.

I started working with Professor Golin at the beginning of 1993. His focus was Theory, tools and applications of Visual Programming. I was leaning towards providing a visual environment for the development of intelligent agents to assist in intelligent automatic user interfaces for my PhD work before he resigned from the University.

I am currently working on my PhD with Professor Ralph Johnson. His focus is on object-oriented technology and how it changes the way software is developed. In particular, he is interested in how to use and develop frameworks, which he believes is a key way of reusing designs and code using objects.

I am investigating "visual languages for business modeling". I am designing them, using them, and implementing them. My current focus has been on using frameworks to develop and implement visual languages for use with business modeling. This project is aimed at providing support for decision making during the business process.

I believe that Frameworks are both a way of coming up with visual languages and a way of implementing them, because if you focus on building something in an Object-Oriented language, then building a framework for it, then making the framework composable, and then making a direct manipulation tool for composing applications using a framework, you will automatically discover a visual language.

I am also interested in finding and describing the design patterns in visual languages and business modeling. To support this I am also researching "Domain Analysis and Engineering". This project is being funded by Caterpillar at the National Center for Supercomputing Applications.

My Research Interests Include: Computational Theory, Learning Theory, Human Computer Interaction, Software Engineering, Computer-supported Cooperative Work, Visual Programming (including grammars and parsing), Expert Systems, Intelligent User Interfaces (providing intelligent automatic semantic feedback), Object-Oriented Programming and Databases, Design of Reusable Software-specifically with the use of frameworks, Domain Analysis and Engineering, and Pattern Languages of Programming.

I enjoy learning and have worked many jobs during my academic career including: Teaching Assistant, Research Assistant, Consulting, Systems Analyst and Programming, Honors Research, Design and Implementation of Medical Computer Systems, etc.