Is Object-Oriented the Panacea?

ISRI Practicum Report II
April 25, 2003

Lu Luo
Institute for Software Research International
School of Computer Science
Carnegie Mellon University

Is Object-Oriented the Panacea?
(ISRI Practicum Report II)

Lu Luo
Institute for Software Research International
School of Computer Science
Carnegie Mellon University
April 25, 2003

Abstract

The reengineering of legacy systems is considered as one of the most significant challenges facing software engineers. Object-oriented analysis and design methodologies are often chosen by design-teams to re-engineer large complex legacy systems. Failure to reengineer can hinder an organization's attempt to remain competitive. How effective are object-oriented methods for a specific system? What are the design tradeoffs on performance, deployment costs and market window? What are the organizational issues involved? There have not been many pragmatic examples that designers can refer to answer these questions when being under similar circumstances.

In this report, I present a case study on the re-engineering process of a large complex real-time software system using object-oriented design and analysis. Several software engineering issues are addressed from this real-world experience and the lessons learned from the analysis are summarized.

This report is submitted to the Institute for Software Research International of School of Computer Science at Carnegie Mellon University, to partially fulfill the practicum requirements of the PhD program in software engineering.

1. Introduction

Object-oriented (OO) development methodologies have introduced a software development model that promises remarkable benefits including reusability, thus providing potential savings on development and maintenance costs. While being globally adopted into new software development projects, the advantages and disadvantages of using object-oriented methodologies to update existing software systems, especially legacy systems[1], are unknown. In this report, I present an industry case study on a re-engineering project that updates one part of a large complex legacy system using object-oriented analysis and design methodologies. The case study is based on the author's one-year involvement in the project as a developer. For nondisclosure purposes, I shield the identity of the organization under this study and use the name "*Company B* " instead.

Company B has been one of the major players in telecommunication and networking technologies for many years. The company's research and development arm is considered to be one of the world's most prolific invention factories. In response to the rapidly growing telecommunication market in China during the last decade, Company B has been actively adjusting its strategies towards establishing a stronger China-based engineering process. In addition to providing products, Company B has established research institutes, research and development (R&D) labs, manufacturing factories, and service-providing offices throughout China.

[1] I define legacy systems as those software and hardware systems that have evolved over many years.

Among these Chinese branches, only the R&D labs of Company B have a focus on the research and development of high quality software for the company's telecommunication systems. The case study in this report takes place in one of the Chinese R&D labs of Company B. For the convenience of discussion, the project under our study is going to be called "*Project H*" in the rest of this report.

The mission of Project H is to develop the control software for a new hardware unit with updated feature that is to be added to one of Company B's large telecommunication systems. The entire hardware and software system has evolved over 11 years when Project H is initiated in 1997. In order to embrace the by-then new object-oriented design and analysis methodologies, the management team and technical leaders decide to adopt object-oriented techniques when developing this new piece of software. Many problems arise during the object-oriented re-engineering process of Project H. Lacking previous experience of object-oriented techniques, the leaders of Project H immediately embrace object-oriented methodology without considering the design tradeoffs before the project is started. Some essential questions, such as "Is it really worth the reengineering effort?" "Is it necessary to use OO methods?" and "What is the tradeoff between cost and productivity?" are not raised until half way through the project, or even after the project cancellation.

The purpose of this report is to identify and analysis several software engineering issues from the project, in hopes of providing portable experiences to similar situations elsewhere. The report is organized as follows: Section 2 sets up the context for the discussion by introducing background information with relevant details of Project H, based on the author's direct involvement in the project as well as follow-ups. In Section 3, reflections on the experience are given by discussing several software engineering issues and the lessons learned. Section 4 summarizes the practicum report.

2. Description of Experience

In this section, a detailed description of the author's experience with Project H is given. The description includes both technical and non-technical facts that are helpful to set up the context of the discussion in Section 3 on software engineering issues. The composition and background of the team are given first. The technical facts about the project, including an overview of the system and the evolution of both hardware and software systems, are then briefly introduced. The technical description is limited to setting up the general framework for the sake of discussion, and no details of the company's business are given.

2.1 Human Resources

2.1.1 Team Composition

Project H takes place in Company B's R&D arm in China. The R&D arm also serves as the Asian-Pacific software development center of the company. When the company's R&D branch in China is first established, its high-level management team is composed of employees promoted from the US headquarters, while all developers and researchers of its R&D arm in China are recruited locally. As part of the company's localization policy, more and more local employees are promoted to management positions as time goes by. Project H is among one of the earliest projects in the company's R&D arm in China, and the management team is totally USA trained.

The organizational structure of Project H team is shown in Figure 1. The positions circled by dashed line are the members working on the project. The lines connecting positions indicate the reporting relationship: a position reports to a higher-level position it connects with. The grayed boxes indicate those who work at the US headquarters, including two project managers and the system architect. On the Chinese side, there are in total 30 local team members including the team leader, developers, testers, and system engineers. The management team members, including the directors, project managers and the architect, are all native Chinese speakers with US education background. As part of the company's localization policy, this arrangement of the team structures can largely bridge the potential cultural gaps that the company might encounter when entering a foreign market and establishing local technical arms.

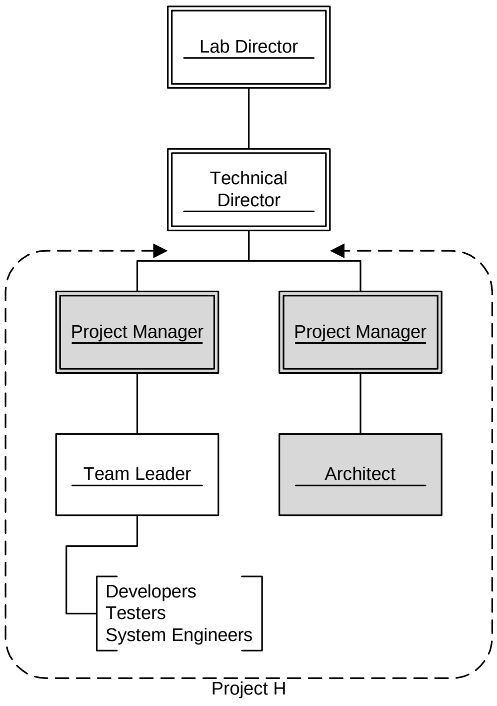

Figure 1. The organizational structure of the R&D arm of Company B

2.1.2 Education Background

In general, the R&D arm of Company B is highly selective when recruiting local technical employees. Among all the technical employees in the R&D arm, about 24% have a PhD degree, 68% a Master's degree, and only 8% have a Bachelor's degree. The recruitment process focuses not only on a candidate's education background, technical skills and teamwork, but also on his/her oral communication ability in English. This serves as an essential means of guaranteeing effective and efficient communication between local employees and US employees who do not speak Chinese.

Being the software center of a telecommunication company, the R&D arm in China tends to maintain a beneficial balance of the employees' field of interest. Most of the local employees are equipped with postgraduate degrees in Electrical Engineering, Computer Science and Computer Engineering. Based on the author's evaluation, among all the local technical employees, about 30% major in Computer Science and Engineering, and 70% of them major in Electrical Engineering. This evaluated percentage does not have any statistical support though.

2.2 Technical Background

2.2.1 Product Overview

The product that Project H team works on is a large Digital Cross-Connect System (DCS). DCS systems direct and manage digital network traffic from a multiplicity of sources at different speeds. DCS has come to play an increasingly important role as the heart of the telecommunications link. A DCS system can cross-connect any input line to any outputline, adjusting for line protocol and speed varieties. For the simplicity of our discussion, when I refer to the DCS system that the Project H team works on, I use the name "*System D*" as its identity. System D is a product line of digital cross-connect systems that are available in a complete range of configurations. System D has a DCS for network applications of every size. System D is designed to meet several requirements:

- Reliability. A high reliability carrier-level is provided through robust hardware, redundancy of critical equipment and service paths, and ruggedized software system core.

- Scalability. Start-up configurations can be deployed at the start of a growing network, which can be expanded with new features and higher capacity, at cost levels that are sensitive to the user's business needs. This supports cost-effective network growth as the numbers of end users increase or as the demand for more services rises.

- Innovation. The System D product line continuously adapts to meet new service and network applications. This is supported by the upgrade schedule, which consists of major feature releases every 12 to 18 months and updates scheduled between major releases.

The first release of System D was developed and shipped in 1986. By 1997, when Project H was initiated as an upgraded release with new features, more than 3,700 System D's in various configurations had been deployed in 100 different networks worldwide.

Project H was initially planned in 1997, when the company intended to expand its market share in Europe and build a new feature to System D according to European standards. During this phase, a couple of engineers from the original System D were prompted to technical leader and project managers, and were assigned to start the initial design of Project H. No local employee was involved at that time. In 1998, the project was officially launched. As part of Company B's localization policy in China, the higher-level management decided to move all the development work of Project H (the new feature) over to the newly established Chinese R&D arm. Therefore, 10 local developers were recruited as the first group of Project H team members. The size of Project H team increased to 18 people in 1999, when the author joined the team with seven other new employees. In early 2000, six months before the project was cancelled, the team enlarged again as number of developers reached 30.

2.2.2 Hardware System Evolution

System D is a software-based multi-processor controlled system that interfaces various telecommunication signal lines. In order to keep up with the changing customer needs on different network configurations and services, the system has evolved at a frequency of upgrading with a major feature every 12~18 months. Figure 2 illustrates how System D has evolved over the past decades. A real system structure may vary greatly from what is shown in the diagram. The initial release of the system has the ability of interfacing multiple of DS1[2] signals. It contains a DS1 unit processor that can accept DS1 signals, de-multiplex them into

[2] Digital Signal X is a term for the series of standard digital transmission rates or levels based on DS0, a transmission rate of 64 Kbps, the bandwidth normally used for one telephone voice channel. DS1, used as the signal in the T-1 carrier, is 24 DS0 (64 Kbps) signals. More information on the T-carrier and E-carrier systems can be found in Appendix A.

various tributary signals, switch the lower bandwidth signals, and/or multiplex them back to DS1 output lines.

As the networks grow worldwide and the market needs in European countries increase, it is necessary to upgrade System D so as to provide the capabilities of dealing with the European transmission rate standards, i.e., the E-carrier system, which is equivalent to the American T-carrier system. A new E1 unit processor, as is shown in Figure 2, is added as an upgraded feature to this new release of the system. Similarly, more unit processors, such as the STS1 Unit, etc. have been added to the system in different releases.

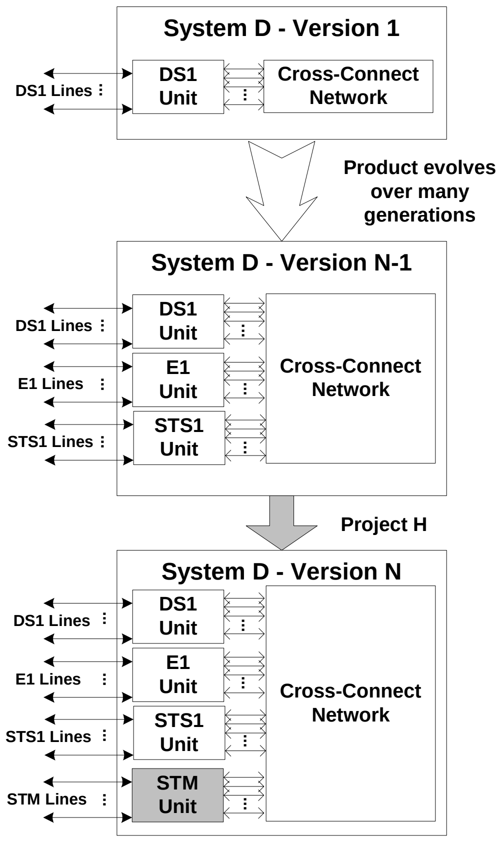

Figure 2. Evolution of System D

Project H is initiated in response to the proliferation of optical networks and the needs to expand European market. Company B decides to promote System D to support Synchronous Digital Hierarchy (SDH[3]) to acquire larger market share in Europe. Consequently, a new unit processor (referred to as *Unit H* in the rest of report) is to be added as the new feature of the next release of the system in 1997. Unit H is designed to accept optical lines carrying the Synchronous Transport Modules (STM) and rates, to transform optical signals into electrical signals, to de-multiplex and cross-connect the converted electrical signals, and transform them back to optical signals. The shaded arrow and the shaded box named "*STM Unit*" in Figure 2 illustrate the addition of this new unit processor forms a new release of System D. The Project H team's

[3] See Appendix A for detailed description of SONET/SDH standards.

mission is to write the software for Unit H that supports STM interfaces. The project was initially planned in 1997, officially started in August 1998, and cancelled in May 2000.

2.2.3 Software System Evolution

The basic idea of Project H is to build software for the new Unit H hardware using object-oriented design and development methodologies with the aid of ObjecTime. ObjecTime is a development tool of Rational that supports graphical object-oriented modeling and high-level simulation. In order to differentiate the existing software system and the object-oriented part created in Project H, I call the existing software the "*legacy code*" or "*legacy software*" and the new software "*the model*." The legacy software for pre-project-H versions of System D distributes over more than ten unit processors and contains about 600 K lines of standard C code written under Unix.

Legacy systems are often defined as those that still have some value yet significantly resist modification and evolution. In the case of System D, "legacy" is not a very accurate classification, since it has evolved over many years and is still actively used. Before the new feature of Project H is added, only slight bug-fixing maintenance work is needed for the legacy software. In addition, though a new unit processor is added for each new carrier rate interface and the controller software has to be updated correspondingly, all units share a set of common central control processes and similar internal logical structures. In Figure 3, the basic infrastructure of legacy software is shown. The rectangular boxes are software procedures dealing with interface and management functionalities, which are common to all hardware units. The common procedures are written within the same source files and switch statements are used to modify the program perform for each unit.

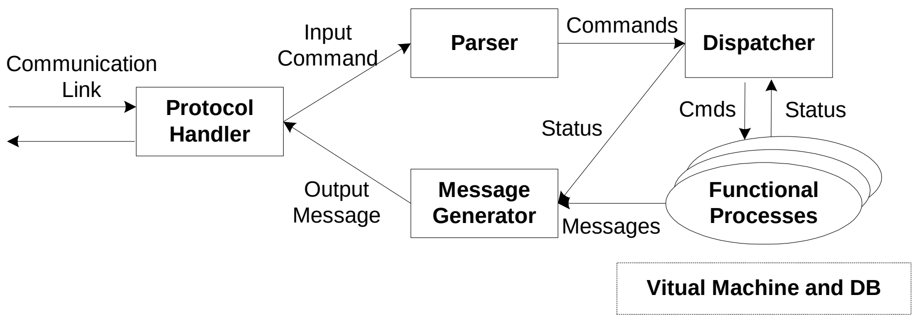

Figure 3. Legacy software structure of System D

The oval boxes in Figure 3 illustrate the functional processes that perform the major switching functionality network cross-connect jobs, such as *Frame configuration* and *Cross-connect*. There is a set of 12 functional processes in every unit and these processes communicate with one another via the Virtual Machine and a global database across the boundary of unit processors. For all units, the set of 12 processes contain similar functionality. The behavior of the processes depends on how the carrier interfaces and network schemes are defined or configured for each specific unit.

Throughout the evolution of System D, each new upgrade of the legacy software follows the same steps. For each major release, a new unit processor is often added with a set of new specifications. The controlling software is upgraded correspondingly: the interface/management procedures code is updated and new function modules are added to the functional processes set. The general software infrastructure is always kept unchanged, and most of the upgrading work is to modify all the interface/management and functional processes that are involved in the new specifications.

3. Issues identified

In this section, I discuss several software engineering issues as a reflection from this case study. The issues I will discuss are:

- When should a system be re-engineered?
- Is object-oriented methodology a great idea?
- How to select the right design tools?
- Tradeoffs in developers' learning curves and productivity
- Meritocracy or democracy?
- Issues on globalization and multi-site development

Most of the issues are organized around questions raised from the case study. Each of the subsections below focuses on discussing one issue, followed by some lessons learned from the case study.

3.1 Re-engineering Tradeoffs

The ultimate target of Project H is to implement the controller software for the new Unit H hardware of System D. Unit H is an integral part of the whole system and cannot work alone. The newly added part must work seamlessly with the existing system, hardware or software. In Project H, the software for Unit H is totally redesigned using object-oriented methodologies, with the aid of ObjecTime tool, and is implemented in C++ language.

As is mentioned in Section 2.2.3, it is beneficial to re-engineer a legacy system if: a) it is difficult or costly to maintain or enhance the system, b) the language and technologies used are obsolete, and c) there are known problems unsolvable without re-engineering. In the case of Project H, none of the above benefits seemed clear.

First of all, based the evolution process of System D, to upgrade the software with new features that support Unit H could be as simple as adding switch statements to the existing legacy code, because of the similarity of logical structure and functionalities among all units. It is also not difficult to maintain the existing system since a very small debugging team is needed to accommodate work such as bug-fixing and modification requests. A non-beginner engineer can usually fix a bug or modification request with turnaround often within a single day. There are very few known unsolved problems for the system as well.

Secondly, the programming language and technologies used in System D are not obsolete. The standard C language is still the most efficient language popularly used in large real-time software systems and is well supported under the Unix environment. On the other hand, all the carrier rates and signals supported in all previous versions of System D are still widely used. In fact, the upgrade process of System D is as simple as cloning a set of relevant functional modules with necessary modifications. No fundamental changes in the system structure are needed.

Not only are the benefits of re-engineering unclear, the potential harm that re-engineering might bring to the legacy system is underestimated as well. It is not clear whether the re-engineering effort will save future development or maintenance work, since only the Unit H software is involved. In addition, the model design automatically generates a skeleton of C++ code for Unit H using ObjecTime. The developers then write the functional and structural details for each object.

As a result of automatically generated code by a graphical tool, the model software contains substantial redundant code that increases the system's footprint.

Lessons learned:

- Make sure that a system is a real legacy before re-engineering
- Evaluate the pros and cons of the re-engineering effort beforehand
- Keep both new and old components in mind, they coexist and interact

3.2 The OO Chaos

According to the technical leaders of Project H, the motivation of using object-oriented analysis and design methodologies to rebuild Unit H software is that OO is trendy and promises to do a better job. The decision of using object-oriented methods to develop the new piece of software was made in 1997, when object-oriented techniques just started to be used widely by development teams. Object-oriented design and analysis methodologies were pretty mature at that time, and it would not be very difficult for the technical leaders of Project H to consider the design tradeoZffs carefully before the project was started. It is unclear to the author how the design decisions were made exactly, but the designers did not seem to think much about the tradeoffs in advance.

3.2.1 Problems and solutions

The OO re-engineering process of Project H turned out to be chaos. The scope of the project determines that object-oriented methodologies can only be adopted in the new software for Unit H, not the whole system. Compared with the set of 12 functional processes for each unit in the legacy code, the object-oriented model for Unit H contains more than 20 actors[4] at the top layer, with more detailed actors at lower layers. Each actor belongs to one of four major functional categories such as *Provisioning Entities* and *Board Control*. In addition, there are several managerial actors that take care of the coordinating among actors.

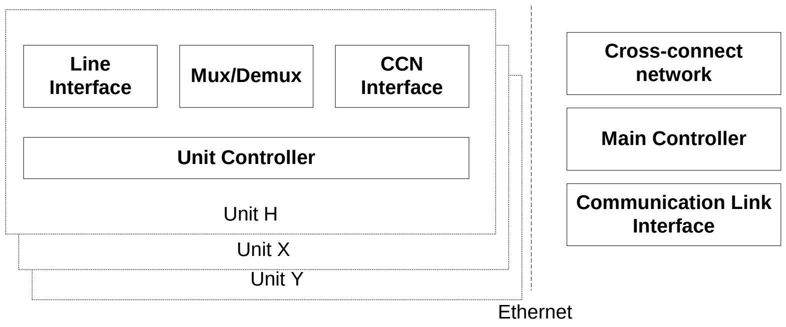

Figure 4. System D hardware architecture

The intention of designing such a system partition is to avoid centralized software control, and to make functional features independent to the system architecture. However, this design lacks

[4] **Object-oriented software architecture design and implemented in ObjecTime is a collection of collaborating actors. Actors are concurrent active objects that have their own logical threads of control. Find more description on architecture elements in ObjecTime in Appendix A.**

careful consideration of several key factors that are vital for the OO model to work seamlessly with the legacy system. The problems come from ignoring the dependency of OO software architecture on both the hardware and legacy software architecture of System D. I start from briefly analyzing the hardware structure to provide a better understanding of the problems. Figure 4 shows the hardware architecture of System D.

The hardware system of D is composed of three common components, *Cross-connect Network*, *Main Controller*, and *Communication Link Interface*, which are shared by all Units of the system. Each Unit deals with a specific input signal rate. Within each Unit, there are four sub-systems (each being an individual processor): *Line Interface*, *Mux/Demux*, *Cross-connect Network Interface* (CCN), and *Unit Controller*, that performing different functionalities. In the legacy software, the functionalities of each sub-system are abstracted as one or more functional processes. The processes on different sub-systems (thus different processors) communicate with each other using Inter-Process Communication (IPC) via *Virtual Machine*.

The functional categories and actors identified in the OO model design do not correspond to the existing hardware and legacy software systems. As a result, no matter how the objects are encapsulated, it is impossible to prevent some functionalities of an object from being involved in multiple hardware processors. For example, the functionalities of an actor in the Provisioning Entities group distribute between both the Unit Controller and Mux/Demux sub-systems (processors).

This problem obviously originates from the original OO model design, and it would be easier to solve it at a high level of the design. However, as I have explained, when the problem is exposed during the detailed design phase, the architect actually refuses to make any modification in the original design. In order to overcome this defective original design at lower levels, the developers introduce an artifact, namely the *half-object plus protocol* pattern. Figure 5 illustrates the basic idea of this pattern. An object is divided into two *half-objects* that cross the boundary of two sub-systems (processors). A *protocol* is then defined between two half-objects to talk to each other. The IPC mechanism is used to hide the details of communication services between half-objects. The actual executables on each processor bears the details for half-objects and IPC, therefore no change is needed to the existing partition of actors in the original model design.

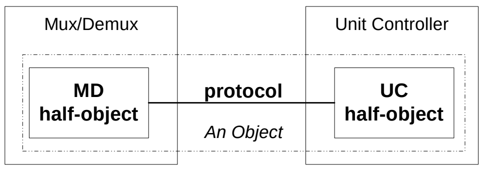

Figure 5. “Half-object plus protocol” pattern

The second problem comes from the coexistence and interaction between the OO model and the legacy system. The OO model only affects Unit H software, a small portion of the entire system. The software for all existing Units cannot be changed. The parts shared by all Units, which containing the common interface/management processes (the part to the right of “Ethernet” in Figure 4), should also be left untouched. To make Unit H work seamlessly with the entire system, the OO model must interact with all the unchanged legacy parts. Since the legacy processes and the half objects have totally different expectations regarding messaging, the two paradigms are not compatible in the original OO design. Again, the original design does not address this interface issue and low-level artifacts have to be introduced to amend for the problematic original design.

To solve this problem, the developers add a Legacy System Interface (LSI), a.k.a. the *facade*, to bridge the interface gaps between the model and the legacy systems. Figure 6 illustrates the LSI between the legacy and OO software systems. The messaging mechanism of the legacy processes and the model objects are different: on the legacy side, a mailbox is used and processes interact by sending messages, while on the OO model side, proxies are used for the communication between objects. Therefore, an *Adaptor*, a Mail Box Server, and a LAN manager are added as the post office between the model and the legacy parts. An adaptor has one or more associated mailboxes, and may talk to multiple half-object proxies. Adapter also translates legacy messages to OO proxies, and vice versa, and delivers the messages to/from the appropriate mailboxes or proxies. The Mailbox Server distributes the messages to appropriate mailboxes in a way similar to the Virtual Machine in the legacy part. The LAN Manager is responsible to send/receive messages to/from the communication service. With the LSI mechanism, no changes are needed for both the legacy software and the model.

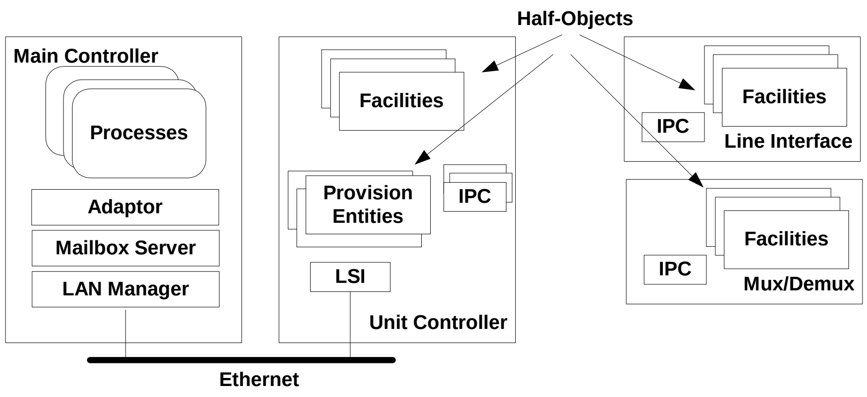

Figure 6. Artifacts introduced to overcome design defects

3.2.2 Discussion

The architect started working on the model with another engineer who left the team shortly after Project H was started. By the time the author joined the team in 1999, the initial model was almost finished and "ready" for the developers to work on. However, the complex issues mentioned above were not addressed until most developers started detailed design, and even implementation. In fact, the model has been discussed and questioned all the way through project cancellation and a tremendous amount of time was spent on solving the half-object problem and building the facade.

Even though the implementation of the model was almost finished before the deadline, the object-oriented design of Project H is not considered successful, which attributes to several defects in the project setup. The tradeoffs of adopting object-oriented design are not fully considered beforehand and there is no analysis or explanation on why re-engineering the Unit H software using object-oriented techniques is possibly more beneficial in comparison with following the legacy design and implementation. Secondly, given the team's inadequate experience on object-oriented design and development, the limited project timeline and market window, and the complex interaction between the model and the rest of the system, the time and cost of the design is not taken into careful consideration.

Conceptually, object-oriented techniques could be used on most software projects, as a way of thinking about problems using models organized around real-world concepts. Object-oriented models are useful for understanding problems, communicating with application experts, modeling enterprises, preparing documentation, and designing programs. Compared with conventional development methods, object-oriented methods may take more time because of the intent of promoting future reuse and reducing downstream errors and maintenance cost. Subsequent iterations of an object-oriented development are easier and faster than with conventional development because revisions are more localized. Object-orientated developments actually enable the development of reusable components such as class libraries, other than support software reuse. Reuse of objects is hindered by the large number of classes that are entangled in a web of association, aggregation and generalization relationships. The interdependencies make it difficult to reuse the classes elsewhere out of the context they are created.

It depends on both the nature of object-oriented methodologies and the situation of a specific project whether or not to adopt object-oriented methods in the project. Given the above characteristics of object-oriented development, different project teams should make individual tradeoffs on short-term and long-term benefits, cost and efficiency. Since Project H is cancelled shortly after the development finishes, it is not evident what benefit the object-oriented development has brought to the system. In general, the object-oriented development process provides every team member a much deeper understanding of the system, especially from the intensive debate during the detailed design phase. The OO object-oriented design process also promotes a better documentation for the Unit H part.

Lessons learned:

- Object-oriented method is not universally suitable for all development tasks
- Evaluate long term benefits vs. short term benefits
- Adopt OO development if the individual gains are warranted

3.3 Choosing the Correct Design Tools

The new object-oriented model for Unit H software is designed, documented, and partly implemented using ObjecTime, a Rational software development tool that implements the Real-time Object-Oriented Modeling (ROOM) language (a variation of Unified Modeling Language, UML). ObjecTime enables software developers to generate complete applications directly from their visual design models. The designs are specified using ROOM graphical charts of structure, finite state machines, and message sequences. ObjecTime includes a set of graphical UML design editors, a model compiler that generates C or C++ application code directly from the design models, and graphical symbolic debugger that supports animated simulation of object behaviors.

From the functional aspects mentioned above, ObjecTime serves as a suitable object-oriented development tool for Project H. It is convenient using ObjecTime to discuss design ideas and pass them from senior members to those who just join the team. It is also helpful to perform unit-level testing and debugging by using of the animated graphical debugger.

Many challenges were generated by using ObjecTime. As the first project in Company B’s China branch that adopts ObjecTime, the potential negative impacts of the tool have not been analyzed before. In fact, Project H serves partly as a test-bed of the applicability of ObjecTime for similar projects within the company. The identified problems include:

1. Bulky code size. ObjecTime automatically generates C or C++ code from the graphical design. It also supports animated simulation for debugging purposes. Those features inevitably lead to a great deal of redundancies in the generated code.

2. Instability. Since its first usable version was released in 1995, ObjecTime is not very stable when Project H team uses it. Developers often complain about the speed of

screen refreshment, screen freeze-up, and even system hangs. It also takes a long time to compile the model, with all the graphical features added.

3. Impact on system performance. Although Unit H did not undergo any field test, there have been numerous unit tests done by the developers including many integration tests with the existing legacy system. The overall system performance (running on software simulator for System D) is notably decreased.

4. High license fee and training cost. The author does not have access to first-hand data on the costs of ObjecTime license fee due to business reasons. As far as the author is informed, the group license for ObjecTime that Project H uses is on the order of millions of dollars. As for the training cost, all developers receive part-time training on ObjecTime for about two weeks and the trainer flies over from the United States.

Lessons learned:

- Carefully evaluate the advantages and disadvantages of new development environments
- Consult other uses to discover the hidden challenges of using a development tool
- Budget the tool cost appropriately

3.4 Learning Curve vs. Productivity

Before becoming fully involved with the development of Project H, all team members receive training from both the company and the project. The training time varies based on how early a developer joins the team. The first group of developers joining in 1998 constitutes the backbone of the team and receives more intensive trainings on almost all aspects of the system, including the legacy software and the new object-oriented design. The learning experience of each developer is listed below.

1. Bridge the background gaps – 3 weeks

   All developers recruited into Company B's R&D branch in China have Master's or PhD degrees in one of two areas: Electrical Engineering and Computer Science. Upon entering the company, every developer is given a self-training course in the field he/she is not familiar with. A CS major developer should try to become familiar with the essential concepts of telecommunication, and an EE major with software development methods.

2. Familiarization the legacy system – 2~6 months

   The first group of 10 developers spends six months with the whole System D team working on different sub-systems. Thorough and detailed introductions on every aspect of the system, especially the legacy software is given. The later groups start with easy jobs such as fixing small bugs based on Modification Requests from maintenance teams for the legacy software. These developers familiarize themselves with the system via reading specifications and test standards. In addition, engineers in the first group are assigned as mentors of the new comers.

3. Training on object-oriented development/analysis – 2 weeks

   All developers receive part-time training from a company consultant on OO technology. The training course focuses on analyzing use cases and scenarios, aided by Class-Responsibility-Collaborator (CRC) cards. Having worked on software architecture for large systems and object-oriented technologies for more than 15 years, the consultant also offers jump-start service to Project H with the architect and later with the developers.

4. Training on ObjecTime – 2 weeks

   As part of the development tool, all Project H engineers receive part-time training courses on the usage of ObjecTime and are certificated by the instructor.

5. Training on SONET/SDN – 1 week

   This training course is given to the whole organization. During the course a detailed explanation on the standard of synchronous data transmission on optical media is given.

In summary, the total development time of Project H is about 34 man-years. The time spent before everyone can start working on the object-oriented development is 17 man-years, 5 out of which is spent on becoming familiar with the legacy system and 12 spent on learning object-oriented techniques. I raise a question on the necessity of re-engineering the Unit H software. From the man-year data, it is possible to assume that without using the object-oriented method, the learning curve of the whole team is shorten to 5 years and the total development time is reduced to 22 years. Therefore, if Project H were developed using conventional methods as the previous units, the development time could have been 35% shorter. There is no data on the man-year spent on previous versions of System D under similar circumstance though.

Learning curves are all about ongoing improvements. Developers seldom begin with a project at full productivity. During the initial phase when developers slowly reach full productivity, the short-term productivity loss can dramatically affect the budget of a project. After the initial phase, developers slowly grow to the full productivity and the improvements during this phase can produce substantial financial savings.

According to the Raccoon's analysis[5] on learning curves in software engineering, the stability of a project is vital for productivity improvement. Stability minimizes the short-term losses from restarting the process and maximizes the long-term gains from learning. It is very important to minimize disruptions to the process, to minimize employee turnover, and to remove process bottlenecks.

In order to maximize the long-term improvement from learning curves, teams should be kept together on long-term projects. Slowly adding or removing team members disrupts learning. Developers who join the team late or leave the team early have less time to learn and benefit from their learning. If tradeoffs have to be made between scheduling more people and more time, and all other factors are equal, choose to budget more time since the long-term benefits will emerge eventually.

Upgrading tools and training developers can shorten learning curves and improve productivity faster. In the case of Project H, the company has done a good job budgeting the learning cost and investing in employee training and tool upgrading. The productivity improvement of Project H team is not fully realized due to the project cancellation and team dissolution. It would be more beneficial for the organization to take advantage of the productivity gain on similar projects.

Lessons learned:

- Evaluate long term benefits vs. short term benefits of training and learning curves
- Teams and processes should be as stable as possible to maximize productivity

[5] **http://www.swcp.com/raccoon/raccoon/learncurve.html**

- Productivity benefit from longer schedules that render learning curves

3.5 Meritocracy vs. Democracy

A good architect is the key for the success of any software project. According to Brooks, conceptual integrity is *the* most important consideration in system design, and there are wide productivity variations between good programmers and poor ones. The variations are even larger between good and poor architects.

There is one chief architect for the entire Project H team. It is mainly the architect who initiates the object-oriented design for Unit H software, based on the claimed benefits of object-oriented methods at that time. Although the architect has been an experienced engineer working on System D and other industry projects for more than eight years, it is the architect's first time being a technical leader as well as working on an object-oriented design. Without any previous experience, the architect actually starts learning object-oriented techniques at the same time as the project is planned.

Most developers in the team do not have experience in industry object-oriented development. Although object-oriented methodologies have been widely known and praised, they are not adopted by any project in Company B at that time. The developers gain knowledge on object-oriented techniques from the training provided by a company consultant, who has worked on software architecture for large systems and object-oriented technologies for more than 15 years. The consultant not only gives training courses to new team members, but also offers help to the architect on the design of Project H.

Given the above facts of the team, the object-oriented movement of Project H meets considerable difficulty. The object-oriented architecture ends up being questioned and discussed all the way through the project. The architect's experience with the object-oriented design is barely one year more than the other developers. The architect creates the initial object-oriented design in a little more than six months and assigns the architecture to the developers.

As the developers gain more and more expertise on both the system and object-oriented techniques, they gradually find more defects in the initial object-oriented design during detailed design. Instead of promoting a new round of design iteration to address these defects, the architect insists that the initial design is correct and should not be changed. The developers have a difficult time trying to persuade the architect to make some changes to the initial design and solve the identified problems and, in the end, are not successful. The company consultant, who is the only object-oriented guru has only indirect connection to the project, provides very good suggestions to the architecture but none of the suggestions is actually accepted. Struggling to live with the design defects, the developers have to create the awkward object-oriented model described in Section 3.2.

According to Brooks, the chief programmer in the development team defines all the key elements of the entire system, including specifications, program design and documentation. The chief programmer is talented with at least ten years of experience, and considerable systems and application knowledge. In modern software development teams, many of the jobs mentioned by Brooks, such as writing specifications and documentation, do not necessarily have to be done by the architect, but he needs to make sure the architecture design is successful. The architect's experience and knowledge are essential to the success of the entire development task.

A small, sharp architecture team is needed under the circumstances where the chief person in the team is not equipped with adequate expertise to lead the project alone. In our case, the architect actively promotes object-oriented design, but strongly opposes iteration. As time goes by, many developers have gained as much knowledge and experience the object-oriented design as the architect. In addition, those developers have better understanding on high-level problems exposed during the detailed design process. The object-oriented development could have been much improved if some outstanding developers took part in the architecture design and had influence on project decision-making.

Lessons learned

- The technical leader's expertise is determinant to the success of the design
- In case the technical leader lacks adequate expertise, build a technical leader team
- Communication between technical leader and developers is essential

3.6 Globalization and multi-site development

The localization process of Company B in non-US countries such as China brings prominent benefits to the company. Establishing local R&D branches together with the manufacturing factories and service-providing offices combine the company's technology strength and the local country's broad market. The innovation of the R&D branch points specifically to the local market needs and can be adopted quickly. Secondly, though the average salary for local employees is very competitive, it is still much lower than the average level for employees with the same education background and experience in the US. In addition, the local human resource market in China is made up of numerous IT professionals with good educational backgrounds. US companies like Company B are very competitive in attracting excellent employees with many factors including competitive salary, advanced technology and management, and a world-class reputation. In order to best match the needs of both the company and future employees, the recruiting process of Company B, especially for the R&D arm, is very selective. All candidates must go through three rounds of interviews, held by different technical leaders and directors. The candidate's proficiency in both technical areas and English is intensively evaluated by giving a 45-minute presentation in English. Only about 5% of the applicants are offered a position in the R&D arm.

Language and culture difference are potential challenges for the company's localization process. The fact that almost all high-level management and technical leaders in the local branches are native Chinese speakers makes the communication between local employees and the management more effective. Although the employees are selected based on their English capabilities, the communication between local and US employees in English is not as efficient as that in native languages. The on-site training processes and discussions strengthen the communication between local and US employees.

Multi-site development has always been considered highly efficient, since a common sense is that one or more teams can work on the same project when the other teams are resting, thanks to the time difference between sites, which reduces needed development time and cost. In fact, it is not always the truth. For Project H, almost all team members are in China except for the two project managers and the architect. The developers do not have the right to make major decisions. Once there is project-wide question, the whole team has to wait until the second day for the managers and the architect's response and not much could be solved locally without notifying the higher levels of non-resident management. From time to time, the managers and the architect have to travel to China and discuss project issues. Instead of increasing the efficiency, it actually requires extra time and cost for multi-site development. The company has realized this problem. As the organization grows, many outstanding developers among the local employees are promoted to technical leader and management positions.

Lessons learned

- Globalization/Localization bring benefits to both the company and the local country

- Multi-site development is difficult
- Localization should be more broadly and more promoted

4. Conclusions

In this report, I describe and analyze some real world experience of the author. The analysis is based on an industry project which re-engineers one part of a large software system using object-oriented techniques. From the viewpoint of software engineering, several issues are raised and analyzed. The issues are hopefully “portable” to similar situations elsewhere.

Issues identified are:

- Effective validation of reengineering decision is difficult but important

  Make sure that a system is a real legacy before re-engineering. Evaluate the pros and cons of the re-engineering effort beforehand. Keep both new and old components in mind, they coexist and interact.

- Object-oriented methodologies are not always the best idea

  Object-oriented design is not the panacea. Evaluate long-term benefits vs. short-term benefits. Adopt OO development if the to individual gains warrants.

- Development tools are double-blade swords

  Figure out the hidden challenges of using a development tool. Budget the tool cost appropriately.

- Budgeting developers’ learning curves

  Evaluate long-term benefits vs. short-term benefits. Teams and processes should be as stable as possible to maximize productivity. Projects benefit from longer schedules.

- Choosing the right team, especially the right leader

  The technical leader’s expertise is critical. In case one leader is not enough, build a team. Communication between technical leader and developers is essential.

- Multi-site development is not as easy as one may think

  Globalization/localization bring benefits to both the company and the local country. Multi-site development is difficult. Localization should be more broadly and more promoted.

5. Acknowledgement

I would like to thank John Liu and Johnson Wu for providing important background materials. I deeply appreciate the advice from my advisor, professor Dan Siewiorek. Many thanks to Priya Narasimhan, who is the second faculty reader of this report, and to the ISRI software group at Carnegie Mellon University for providing valuable feedback on my presentation based on this report.

Appendix[6]

A-1 Digital signal X

Digital signal X is a term for the series of standard digital transmission rates or levels based on DS0, a transmission rate of 64 Kbps, the bandwidth normally used for one telephone voice channel. Both the North American T-carrier system and the European E-carrier systems of transmission operate using the DS series as a base multiplier. The carrier system is based on digital bits.

DS0 (64 Kbps) is the base for the digital signal X series. DS1, used as the signal in the T-1 carrier, is 24 DS0 signals transmitted using pulse-code modulation (PCM) and time-division multiplexing (TDM). DS2 is four DS1 signals multiplexed together to produce a rate of 6.312 Mbps. DS3, the signal in the T-3 carrier, is composed of a multiple of 28 DS1 signals or 672 DS0s or 44.736 Mbps.

A-2 The T-Carrier System

The T-carrier system, introduced by the Bell System in the U.S. in the 1960s, was the first successful system that supported digitized voice transmission. The original transmission rate (1.544 Mbps) in the T-1 line is in common use today in Internet service provider (ISP) connections to the Internet. Another level, the T-3 line, providing 44.736 Mbps, is also commonly used by Internet service providers. Another commonly installed service is a fractional T-1, which is the rental of some portion of the 24 channels in a T-1 line, with the other channels going unused.

The T-carrier system is entirely digital, using pulse code modulation and time-division multiplexing. The system uses four wires and provides duplex capability (two wires for receiving and two for sending at the same time). The T-1 digital stream consists of 24 64-Kbps channels that are multiplexed. The four wires were originally a pair of twisted pair copper wires, but can now also include coaxial cable, optical fiber, digital microwave, and other media. A number of variations on the number and use of channels are possible.

A-3 The E-Carrier System

E1 is a European digital transmission format devised by the International Telecommunications Union - Tele-Services (ITU-TS) and given the name by the Conference of European Postal and Telecommunication Administration (CEPT). It is the equivalent of the North American T-carrier system format. E2 through E5 are carriers in increasing multiples of the E1 format.

The E1 signal format carries data at a rate of 2.048 million bits per second and can carry 32 channels of 64 Kbps each. E1 carries at a somewhat higher data rate than T-1 (which carries 1.544 million bits per second) because, unlike T-1, it does not do bit-robbing, a technique used in signaling on the T-carrier system, and all eight bits per channel are used to code the signal. E1 and T-1 can be interconnected for international use.

E2 is a line that carries four multiplexed E1 signals with a data rate of 8.448 million bits per second. E3 carries 16 E1 signals with a data rate of 34.368 million bits per second. E4 carries four E3 channels and E5 carries four E4 channels.

[6] **All materials in A-1 through A-4 are extracted from www.whatis.com**

A-4 SONET and SDH

Synchronous Optical Network (SONET) is the American National Standards Institute (ANSI) standard for synchronous data transmission on optical media. The international equivalent of SONET is Synchronous Digital Hierarchy (SDH). Together, they ensure standards so that digital networks can interconnect internationally and that existing conventional transmission systems can take advantage of optical media through tributary attachments.

SONET provides standards for a number of line rates up to the maximum line rate of 9.953 gigabits per second (Gbps). Actual line rates approaching 20 gigabits per second are possible. SONET is considered to be the foundation for the physical layer of the broadband ISDN (BISDN).

SONET includes a set of signal rate multiples for transmitting digital signals on optical fiber. The base rate (OC-1) is 51.84 Mbps. Certain multiples of the base rate are provided as shown in the following table. Asynchronous transfer mode (ATM) makes use of some of the Optical Carrier levels.
Synchronous Digital Hierarchy (SDH) is a standard technology for synchronous data transmission on optical media. It is the international equivalent of Synchronous Optical Network. SDH uses Synchronous Transport Modules (STM) and rates: STM-1 (155 megabits per second), STM-4 (622 Mbps), STM-16 (2.5 gigabits per second), and STM-64 (10 Gbps).

A-5 Architecture Elements in ObjecTime

ObjecTime is developed by the Telos Group at Bell-Northern Research. The ObjectTime tool was ready for usage internally in January 1990. In 1992, an independent spin-off (ObjecTime Limited) was founded, which brought the tool to the open market. ObjecTime was merged with Rational and then later with IBM.

An object-oriented software architecture implemented in ObjecTime is a collection of collaborating actors. Actors are concurrent active objects with their own logical threads of control. There are two major views of actors: structural and behavioral, which form the key items for modeling. The *structure view* resembles a black box view of the model. It describes communication with other components (actors) via ports, bindings and protocols. It also supports the encapsulation of actors within other actors. The *behavior view* can be thought of as a glass box view, and are represented by extended state machines. At any given time the actor is in one of a set of pre-defined states. Transitions between those states are triggered by incoming messages and may depend on specified logical conditions. States as well as actors can be nested, which allows different level of detail.