Advancing Maturity of Software Engineering Discipline - A Case Study in Distributed Software Development

Thomas E. Potok
Oak Ridge National Laboratory
potokte@ornl.gov

Nenad Ivezic
Oak Ridge National Laboratory
ivezicn@ornl.gov

Kimberly D. Barnes
Oak Ridge National Laboratory
barneskd@ornl.gov

Abstract

Accelerating the maturation process within the software engineering discipline may result in boosts of development productivity. One way to enable this acceleration is to develop tools and processes to mimic evolution of traditional engineering disciplines. Principles established in traditional engineering disciplines represent high-level guidance to constructing these tools and processes. This paper discusses two principles found in the traditional engineering disciplines and how these principles can apply to mature the software engineering discipline. The discussion is concretized through description of the Collaborative Management Environment, a software system under collaborative development among several national laboratories.

1. Introduction

There are two major forces present in today's software development community, 1) the shortening of development life-cycles, and 2) the increased flexibility of the programming team. A few years ago it was common to develop a software project on a 2-3 year cycle with a dedicated team of full-time programmers on the project. Today, the development cycles are often a year or less, and the programming team is typically made up of consultants, subcontractors, and full-time employees. These two forces have begun to shift the focus of software development from the creation of new systems to the enhancement or merger of existing systems. For the most part, enhancing an older system, or merging two legacy systems is far less expensive than developing an entirely new one. These types of changes will most likely drive down the cost of developing software; however, these changes result in several issues. Fundamentally, how can systems be easily enhanced and merged, and how can people be brought together from various backgrounds and locations to collaborate on a project.

There are a wide number of technologies that focus on addressing these fundamental productivity issues. A commonly accepted approach today is to incrementally apply the latest technology to a given problem. However, it has been shown that the latest technologies do not necessarily produce the best productivity returns; likewise, they do not necessarily apply to a given development environment [1, 2]. We believe that software engineering will develop in much the same way as traditional engineering disciplines have. We are investigating the hypothesis that leaps of maturity in this field are achievable by taking advantage of successful business patterns from mature engineering disciplines.

The rest of the paper is organized as follows. Section 2 presents a summary of the related work. Section 3 outlines the working hypothesis of this paper. Section 4 describes the Collaborative Management Environment, a system to automate management of financial data from multiple research organizations. Here, we present our findings from the perspective of the stated hypothesis. Finally, Section 5 presents the conclusions of this paper.

2. Related Work

There is a great deal of speculation about what factors truly impact software development productivity. Some argue that productivity gains can be found through improvements in the personnel aspects of the software development team, while others claim it is either the methodology followed; the maturity of the software development process; or the effects of the governing business practices.

Briefly summarizing the related work, there appears to be no conclusive evidence that the makeup of the development team[1], or the methodology followed consistently effects productivity [2, 3, 4, 5, 6, 8]. There is some evidence to suggest that the maturity of the software development processes and the controlling business process can effect productivity [9, 7, 10].

[1] There have been numerous studies over the years on the effects of the team capabilities vs. software development productivity. Most studies show evidence of such a correlation, however, there are a significant minority that show no evidence. It is hard to draw a definitive conclusion based on these studies.

3. Hypothesis

Our working hypothesis is that it is possible to increase the maturity of the software development process, not only through organizational maturity as suggested by the Software Engineering Institutes Capability Maturity Model (CMM), but also through supporting and accelerating maturation of software engineering discipline as a whole [10, 11].

We are interested in the effects of providing tools and implementing processes within software engineering to mimic evolution of traditional engineering disciplines. In this way we hope to anticipate the emerging phenomena typical of other traditional engineering fields and take advantage of these phenomena to boost the productivity within software development practices. At a high level, these phenomena, can be summarized by widely adopted engineering principles. In this study, we investigate relevance of two principles found in the mature engineering disciplines to software engineering: (1) Enhanced engineering communication; and (2) Principled selection of technologies.

3.1. Enhanced engineering communication

Engineers work with a high level of understanding of the application domain. For example, a structural engineer can communicate about a structure in very precise terms with another structural engineer, an architect, or construction engineer with whom he or she has never met or worked.

In contrast, the software developers typically rely on the expertise of a "domain expert" to provide insight into a system being developed. In other cases the domain expert is taught the basics of software design and led through the design process with the help of software experts.

3.2. Principled selection of technologies

Engineers apply technology as needed to solve a given problem, i.e., the problem determines what technologies should be used, not the other way around. For example, a good structural engineer will carefully evaluate the risk associated with building a long-span bridge using a low-strength, low-durability, or a brittle material that could all lead to near-term or long-term catastrophe.

Most successful software development efforts are done based on the development teams knowledge or expertise with a given computer language, methodology, or process, not necessarily on how well suited the technology is towards solving the problem [12, 14].

4. Collaborative Management Environment

We are in the final stages of the development of a pilot system to automate the management of financial data from multiple independent research organizations. The system is named the Collaborative Management Environment (CME) and its development is a collaborative effort among several national laboratories: Oak Ridge, Lawrence Berkeley, Ames, Los Alamos, and Fermi.

The development of this system follows the above two key principles found in mature engineering disciplines. Our development approach is to (1) develop a precise understanding and definition of the concepts of the domain in which we are working, not to solely rely on outside expertise; and (2) devise a solution that is based on the most suitable technology available, not merely ones with which we are familiar.

This section gives account of the motivation, requirements, issues, overall findings, and future steps for the CME project in relation to the research hypothesis we are investigating.

4.1. Motivation

The pilot we are developing is based on the need at a congressional level to understand how research funding is allocated and spent. The Department of Energy is responsible for funding a vast amount of energy related research. There are a number of national laboratories that receive research funds, and these laboratories work in a very broad range of areas. Any request for information about performed research must be asked of each of these laboratories. Not surprisingly, each of these laboratories follow research management processes that are tailored to their expertise. Likewise artifacts resulting from these processes are in different formats and levels of granularity. For example, a multipurpose laboratory may track a wide variety of research expenditures at a very low level of detail, while a single purpose laboratory may track fewer expenditures in a high level of detail.

The process of submitting and funding research proposals is currently a paper process with many of the labs having electronic proposal submission capabilities. The goal of this research pilot is to develop a system that provides the support for automating the research proposal submission process and providing the capability to report at various levels how research funding is allocated and spent.

4.2. Requirements and Issues

At the outset this project seems rather straightforward; however, several previous development groups have not been successful. The first key challenge was addressing the changing and conflicting requirement which can be summarized as:

1) The labs own the data and will be very hesitant to supply additional data.

2) The system requires additional data for it to be acceptable. The current system was not acceptable primarily due to a lack of data integration. Without new data, the proposed solution would provide little benefit over the existing system.

3) The data must be kept at the labs. This data can be sensitive, particularly research funding proposals that contain new and potentially productive ideas.

4) The system must assure the financial data is accurate,

5) The system is scalable.

6) The system has good response time performance.

7) The system is highly secure.

We have developed several proposals to address these requirements based on a variety of technologies. At a conceptual level, the system needed to be capable of accessing and analyzing legacy data from several labs through a secure, distributed, and high performance system. A variety of issues were addressed in the development of this pilot system, with the two most significant being:

1) the development of a semantic data model used to represent individual laboratory data in a common format, and

2) the creation of a model driven development environment that supported the rapid development of databases and applications.

We further determined that an iterative development process was best suited to this environment, given the wide ranging requirements and the need to demonstrate results early in the development process[12].

4.3. Semantic Data Modeling

From a technological view point, there are many ways of dealing with heterogeneous data. We evaluated what we believed were the best technological solutions and worked with laboratory representatives to find an agreeable solution. We applied our hypothesis in two ways. First, by working with a wide range of potential users of the system to fully understand how the current system works. Secondly, by resisting the temptation to use familiar technology on a new problem. We summarize our technical evaluation of this problem below.

Probably the most popular way of dealing with legacy data is the notion of wrapping legacy data with software that enforces a known interface. We encountered several problems with this approach. Most notably in this case is that the data is independently owned, and there is little external motivation to share it. Why should a laboratory spend its time and effort to wrap a financial database when there is little benefit to them for doing so?

There was also a great deal of resistance to the laboratories loosing control of the data. One of the early project notions was to have agents traverse the independent databases and derive information structures for each. Besides being a challenging problem, the labs adamantly opposed having “outsiders” tramping through their systems.

The compromise that was eventually reached was to define a common data model that a lab could populate as they see fit. The choice of populating a data field was strictly up to the lab. This represents the notion of a federated database.

We spent a great deal of time working with experts from various aspects of the research funding system to understand precisely how this process works. We expressed this understanding in terms of a data model. The focus of these sessions was to thoroughly understand and document the domain and to accurately understand what common terms mean. These sessions revealed three major issues with the data: 1) The data is often informally grouped, such as projects, or research areas. These groups are not defined nor related to the current data. 2) The identifiers of the data can change from year to year so relating data from one year to the next may not be possible. 3) The data is presented in various levels of granularity, some as a collection of proposals, others as individual subproposals.

At the outset, these issues were not clearly understood by the experts with which we dealt. Through detailed modeling and analysis it became clear that we were gaining a broader view than the experts’ view of the overall process. Likewise we were able to tailor a technology to best suit our understanding.

This led to an expansion of our initial data model to address these three major issues. This expansion was a fairly simple exercise. The challenge is in gathering the needed data with minor impact on the labs. The information that we needed is contained in two places, the mind of the researcher who submitted the proposal, and the mind of the program manger who funds the proposal. The approach that we used was to allow these two people the ability to augment available information with their knowledge and to be able to store that knowledge in a structured way.

4.4. Rapid Application Development (RAD)

Having applied a level of enhanced engineering communication to the first part of the project, we now focus on the principled selection of technology for the remainder of the project.

By following the iterative development process, we needed to develop health chunks of software in a very short amount of time. The catch phrase for this type of development

is RAD or Rapid Application Development. The general approach to "RAD" is to develop prototypes of the desired system using software design tools built for this purpose. The idea being that the tools can automate some of the development tasks, such as the development of the user interface. The ideal case for this type of development appears to be one where the solution is driven by the end user view into the system, such as enhancing the interface to a legacy database.

In the case we are presenting, the system is driven by the data and the structure of that data. It is common for the structure of the data to change as the understanding of this structure increases. In this type of environment, one typically defines an agreed to data model from which an iterative develop process may begin [13,14]. In our case, we needed to demonstrate the feasibility of the project long before a universally agreed to model could be developed even though it is based on changing data requirements. Furthermore, any model developed would certainly change after the development of the system took place.

The solution we chose for this problem was to enable the generation of a database definition and object class definitions from the conceptual semantic model we developed. For example, we could make changes to a conceptual object model diagram and from this diagram produce corresponding relational database definitions and object class definitions. This allowed us to rapidly develop new data and class definitions whenever the data model changed. Obviously, significant changes to the data will require changes to the functional code. This encouraged the use of layering and encapsulation which further insulates the code from changes.

The most technologically innovative aspect of the system is its distributed nature. We reviewed three approaches to developing this system. The simplest was to bring all the data to a central database and build an interface to the database that provided the functionality needed. This is a typical application development approach that can provide good performance and security; however, it does not provide support for distributed data. The second approach was to build a distributed database system where every lab would be responsible for maintaining a database of their data, and our system would remotely access this data. This approach had several significant drawbacks, chief among them being the high cost to the labs to setup and maintain a database. The approach we chose was to keep the data at the labs, and represent the data in XML format so that it could be searched for and viewed by web browsers or parsed into a database if needed. The data can be easily stored and maintained by the labs, and the labs can benefit from easy access to this data. Through the use of architectural neutral languages (i.e., Tcl/Tk) and communication sockets, we can use the Internet to enable a robust distributed system. To enhance performance we developed a caching scheme to limit the need for multiple remote accesses for each query.

4.5. Summary of findings

Our preliminary results are qualitative in nature, but nonetheless, show promising findings. The biggest problem we encountered were directly related to the communication issues addressed by our first hypothesis. Several experts used what we initially considered to be commonly understood terms but under persistent questioning, we learned that these terms were vague and had different meanings for different people. For example, there is a concept of a "research project" that sounds very straightforward; however, the domain experts all have a different views of it. Representing a research project is a key aspect of the system we were developing, and even a slight misunderstanding of this term will most likely require significant changes. Had we followed the approach of relying on a domain expert for an understanding of the system, the system would not have worked properly and most likely would have required a large amount of rework. It is clearly premature to speak of productivity gains, but this does give us motivation to continue with our hypothesis.

We are not at a point to determine the benefit of the principled approach to technology mainly because the benefits or drawbacks of this approach will be seen over time. What we have observed is that working with well established technology is a safer approach for developing a project but also yields a higher risk of obsolescence in the future. We have spent a large amount of time weighting the risk of a not proven technology against a proven but fading technology. We have fairly consistently selected newer technology, primarily because of the capability of the technology and the promise that it holds in the market place.

4.6. Next step

Our next steps are the basic software development processes of testing and deploying the system. Following this, we expect to significantly expand the functionality of the system and deploy it to those interested in it.

In the light of our working hypothesis outlined in Section 3 and based on our experiences with the CME project, advancing the development framework to include collaborative information modeling capabilities is one of the most interesting aspects of the future work.

Our experience shows that building of the shared information framework for the distributed participants is perhaps one of the most costly development activities. We are motivated to make this step much less cumbersome. Our main vehicle for achieving this is development of a collaborative semantic modeling environment which would allow participants to propose, negotiate, and quickly implement the shared information framework. The basic assumption for such an environment is existence of a semantic meta-modeling language [15].

A semantic meta-modeling language of the CME domain captures a number of ubiquitous modeling approaches, is extensible, and provides a basis for information description and interoperability. Modeling languages that describe specific aspects of information systems, such as implementation details or interoperability requirements will be derivable from this meta-language (Figure 1). We believe, this common basis will enable efficient building of shared understanding among collaborating participants.

Our principal approach is focused on information modeling languages that are becoming, or have been promulgated as, modeling standards. For example, Unified Modeling Language (UML) is becoming the de facto standard in the Object Oriented Analysis and Design community and is endorsed by the Object Management Group. Knowledge Interchange Format (KIF) is a first order logic representation language developed within a DARPA program and is widely used by the software engineering community and is accepted as a standard by ANSI. Process Interchange Format (PIF) is a process representation language developed at DARPA with a potential to become another accepted standard. We will use the proposed meta-descriptions for these languages to develop the core of the unified meta-modeling language for CME. The standardization of modeling languages, together with other activities in the industry (e.g., component-based software development, software pattern-based development) seem to signal a start of a maturation stage in software engineering.

Figure 2 illustrates our objective -- to develop the mixed-initiative modeling and development process in which an integration engineer, the CME domain experts, and information providers (i.e., participating national laboratories) develop information models as a basis for making the information services interoperable. To achieve this goal, we plan to develop and experiment with a number of tools. We will provide an information modeling tool for participating laboratories to describe their information services. In the course of modeling, the information provider will interact with the integration engineer to request support for the modeling task. This support will be possible to request through a synchronous or an asynchronous channel. In most situations, support will come through asynchronous collaboration using the information modeling assistant. The modeling assistant will be created and maintained by the integration engineer. The integration engineer will use the modeling assistant workbench to develop modeling support specific for each participating information provider.

Domain experts, on the basis of the information models, will interact among themselves to negotiate a model to propose to the information providers. Using the domain modeling tool, developers and domain experts will be able to negotiate the shared domain model prior to its publication.

We are in the process of developing quantitative measurement and analysis procedures for the changes we are implementing. The natural approach would be to measure the output of the software development process and statistically compare this rate to a known output rate. From this we will be able to state whether the types of changes we are proposing really do provide productivity benefits.

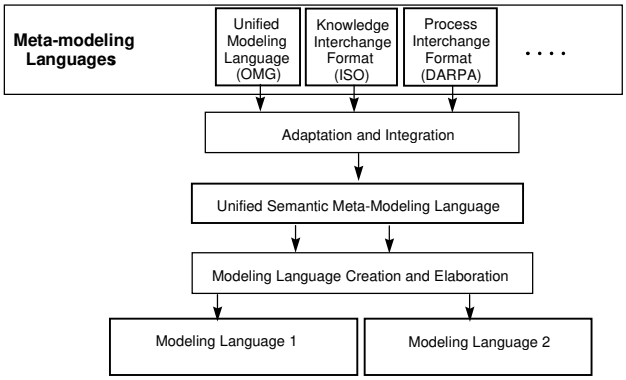

Figure 1. A common meta-modeling language will provide the basic tool for establishing shared information network.

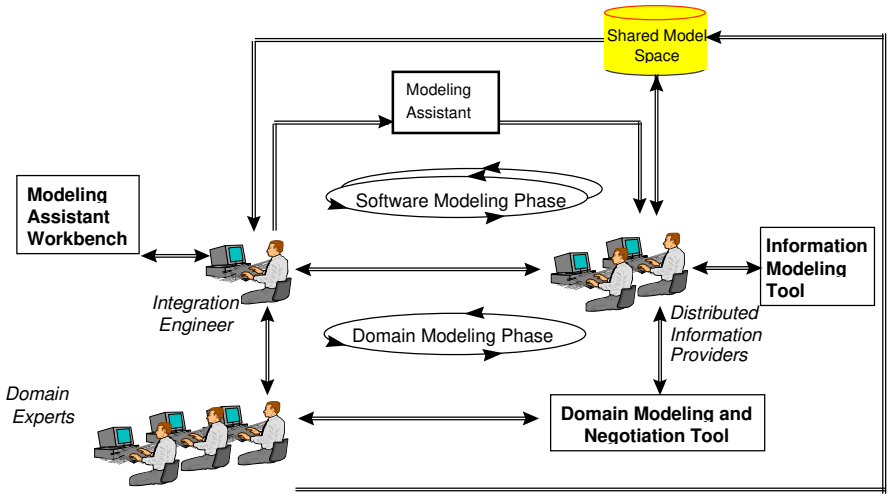

Figure 2.The mixed-initiative modeling and development process relies on supporting tools that facilitate collaborative information modeling and communication between all the actors in the process.

5. Summary

In this paper, we presented initial observations from a study into maturation of software engineering discipline through mimicking evolution of traditional engineering disciplines. We started by presenting related work in the area of software development productivity. Then, we presented two governing principles from traditional engineering disciplines. The application of these two principles was presented in detail for the case of development of the Collaborative Management Environment, a software system under collaborative development among several national laboratories.

6. References

[1] T. K. Abdel-Hamid. "The Slippery Path to Productivity Improvement," *IEEE Software*, 7/1996, pp. 43-52.

[2] T. E. Potok and M. A. Vouk, "Productivity of Object-Oriented Software Development," *Technical Report CACC-TR-96/31, Center for Advanced Computing Communications,* North Carolina State University, Raleigh, NC 1996.

[3] J. W. Bailey and V. R. Basili, "A Meta-Model for Software Development Resource Expenditures," *Proceedings of the Fifth Internations Conference on Software Engineering*, 1981, pp. 107-116.

[4] Boehm, B. W., *Software Engineering Economics*, Prentice-Hall, Inc. Englewook Cliffs, N.J., 1981.

[5] D. R. Jeffery and M. J. Lawrence, "Managing Programming Productivity," *The Journal of Systems and Software*, Vol.5, 1985 pp. 49-58.

[6] B. A. Kitchenham, "Empirical Studies of Assumptions that Underlie Software Cost-estimation Models," *Information and Software Technology*, Vol.34, No.4, 1992, pp. 211-218.

[7] T. E. Potok and M. A. Vouk. "The Effects of the Business Model on Object-Oriented Software Development Productivity," *IBM Systems Journal,* Vol.36, No.1, 1997, pp. 140-161.

[8] C. E. Walston and C. P. Felix, "A Method of Programming Management and Estimation," *IBM Systems Journal*, Vol.16, No.1, 1977, pp. 54-73.

[9] R. Dion. "Process Improvement and the Corporate Balance Sheet," *IEEE Software,* Vol.7, No.28, 1993, pp. 28-35.

[10] M. C. Paulk, B. Curtis, M. B. Chrissis and C. V. Weber, "Capability Maturity Model, Version 1.1," *IEEE Software,* 1993, pp. 18-27.

[11] Humphrey, W., *Managing the Software Process,* Addison-Wesley, Reading, MA, 1989.

[12] Boehm, B. W., *Software Risk Management*, IEEE Computer Society Press, 1989.

[13] Booch, G., *Object-Oriented Design with Applications,* The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1991.

[14] Rumbaugh, J,, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen. *Object-oriented Modeling and Design*, Prentice Hall International, Englewood Cliffs, NJ, 1991.

[15] N. Ivezic, T. E. Potok, and K. D. Barnes, "Achieving Maturity of Compositional Software Engineering Discipline" *Proceedings of the Workshop on Compositional Software Architectures, http://www.objs.com/workshops/ws9801/papers/paper100.doc,*1997.