International Journal of Information Technology and Knowledge Management
January-June 2008, Volume 1, No. 1, pp. 131-137

PRESENT RESEARCH SCENARIO OF SOFTWARE REUSE

Anil Kumar

Abstract

Software reuse plays a significant role in the development of new software. The idea of software reuse was projected in late sixties. History of software development describes that area of software reusability was not so innovative. But it is believed that software reuse is still in the development phase and has not achieved its full potential. This paper describes how far we are with software reuse research.

Keywords: Software reuse, artifact, software component

1. Introduction

Software Reuse is defined as the process of building software applications and systems from previously existing software. It has been associated with software engineering since its early days as the NATO Software Engineering Conference in 1968 marked the birth of the idea of systematic software reuse. Since then, reuse has become an important research area in software engineering. Software reuse has been a buzz word in large companies for some time now, with its potential for achieving good quality systems in short time scales by the reuse of currently available components. Many success stories have been quoted, from Raytheon's 50 increases in productivity due to a reuse rate of 60%[1], to GTE's saving of $1.5 Million from a reuse factor of 14%[2], to the Japanese software factories' claim of annual productivity increases of 20% by implementing a software reuse program[3]. It would be foolish to claim that software reuse is the solution to all the problems that have caused the current software crisis. Achieving software reuse on a level at which substantial benefits will be gained is a difficult task, and requires a great deal of commitment and effort. Tracz emphasizes that "there is no free lunch when it comes to software reuse[4]." There are, however, techniques that can help a company to maximize its resources and improve its productivity. It has been shown that reuse offers great benefits if used effectively in the right environment. This however raises the questions: how are software reuse techniques best employed and what is the right environment for software reuse to prosper? All the published examples quoted have been large, well structured companies, with top level management support for the reuse program. This suggests that software reuse tends to prosper in such an environment, but what about the small, less structured companies, whose livelihood depends on the ability to produce their product as quickly

as possible, while trying to keep standards high enough to keep their customers happy and their maintenance costs low. To them, the benefits of software reuse could be invaluable.

The reuse of software in systems development is a strategy that increases productivity and quality[9]. The component based development is a paradigm aligned with this strategy because it focuses on the reuse of already existing parts (components) and the development of new reusable components[10]. It is very noticeable the research done in the past years on tools that promote software assets reuse. When taking intra organizations into consideration we have examples of digital assets repositories like the Digital Assets Manager[11] and the integration of development environments with these repositories. In terms of inter organizations; we have the XML Web Services standards in the mainstream and the Component Sharing Peer-to-Peer Network, the RCCS[12].

Large worldwide enterprises have several available applications (also called legacy systems) developed in a wide variety of technological platforms directly affecting the potential of immediate reuse of the investments already made in the past. Inside this trend of maximizing the potential for reuse, legacy systems are existing software systems, usually mission-critical and typically still in operation that often represents a crucial drawback. The main characteristics of the legacy systems are: the size, in average of millions of code lines; they encapsulate significant business logics and normally present a reduction on the quality of the original project due to the maintenance and evolving activities .In this paper it has been discussed that how reuse technology is used and at what extent we are using its potential in the present dynamic world.

1.1 Scope of Software Reuse

Software reuse can be divided into product reuse and process reuse according to the reuse object. The product reuse means the reuse of software component, getting a new system from component integration and construction. The process reuse means the reuse of past software development process, automatically or half automatically producing the system using the reuse generator. The process reuse depends on the technical development in the software automation, only applicable to some special applied domain currently, but the product reuse is a realistic and essential path now.

1.2 Mode of Software Reuse

The software reuse can be divided into black box reuse and white box reuse according to the reuse mode of reuse information. The black box reuse means to use the component directly without modification while the white box reuse means that the component is not according to customer need hence can not be used until be properly

modified according to the customer need. But in the mostly applied construction process, the adaptability modification of the component is essential[14].

1.3 Process of Software Reuse

i. Domain analysis phase: This phase is to come certain whether deserve to reuse the infrastructure for the domain development mainly through the definition and analysis of application domain.

ii. Domain engineering phase: This phase is to acquire general system structure according to the domain commonness getting from domain analysis phase and regulate how the property match the system structure and how to bind variable point.

iii. Property obtaining phase: This phase includes development of reuse, may also include some exterior adopt of reuse property.

iv. Property categorizing phase: This mission is a database management mission actually, including categorizing and saving reuse property.

v. Property maintaining phase: This mission is a maintenance mission actually

1.4 Making Reuse Attractive

There are two main challenges to effective reuse within a company: technological and organizational[7]. As technology has advanced, and the methods and tools to support reuse have become available, the technological challenges facing reuse have been surpassed by the economic and organizational issues that face a company intending to implement a reuse program[8]. One of the major challenges with software reuse is that of introducing a reuse framework and method into a company. There are several steps that we have used in this process. The first, and perhaps the most important, step is to gain the support of the top level management for the reuse program[9]. This is crucial, because the introduction of a reuse program affects all parts of the software production process in the company. Therefore, the support of the high level management in charge of all aspects of development must be gained to allow changes to be supported and implemented and to allow changes to company policy to be made if needed[10].

Hence we conclude that the support of the high level management by giving a presentation on reuse, explaining how it could help in their company and how to best utilize it. This was a good opportunity to present the case for reuse, stressing the benefits that it could bring to the company, and the approach to introducing reuse into a small company that we were recommending. It was also a good point at which to get feedback from the management on what they expected from the reuse program, and how they wanted the company to change for the future. Suggestions were then made

how to set up a reuse program into the company, along with the costs that would be associated with the setting up of this program, and a consideration of the risks involved.

1.5 Implementing a Reuse Program

This research has recommended an incremental approach to introducing a reuse program within a small company. There are many reasons for this. One of the major reasons is that it is not wise to jump straight into a new development strategy that will change the way that the company works without first being assured that the strategy is applicable to the company, its staff, its domain and its working environment. Although software reuse can offer major productivity gains when used in the right way in the right environment, it is not the solution to all problems. There will always be times when it will be more effective to write new code than to try to find and reuse old code. The key is to recognize which techniques will be most effective in different development environments, and utilize the most efficient development strategy in each case.

1.6 Present Reuse Scenario

The *Standish Group* has published a report to describe and analyze the projects in the software industry. Since this industry is so immature, the results presented in the report are so incredible. In that first report about 15% of the projects were classified as successful projects. Almost 15 year later, the successful projects are only a third of all projects in the software industry. Considering that the global economy tends to demand more and more software over the next years, emerges the necessity for a new way of develop software, a new modus operandi (mode of operation) in which we can move the software industry from the prehistoric age, from the craftsmanship to a new context based on manufacturing. The *software factory* approach leads us to put beyond all empiric methods that brought the software industry to this point. This approach address the economy of scope (completed by the economy of scale) in order to increase the Return on Investment (ROI). A software factory also allows the introduction of systematic methods developing software. *Greenfield* defines a software factory as "A software factory is a software product line that configures extensible tools, processes and content, using a software factory template based on a software factory schema, automate the development and maintenance of variants of an archetypical product, assembling and configuring framework-based components " Thus, since we know that specific approaches are more efficient (and more limited) than generic approaches, emerges the concept of Domain Specific Languages (DSL). *Deursen* defines a DSL as a: "A domain-specific language (DSL) is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain". Furthermore, in this context, the developers are focused in creating products based on the definition of

the problem domain (not being worried about memory variables). So, it's believed that the productivity, reliability, maintainability and portability of the software will increase so much. Then, the developers can associate semantic transformers to generated code from the models expressed in a DSL form. Therefore, DSL can be an important tool to allow the software factoring become actually successful. However, we must be critic and introduces some questions:

- How efficient the DSL are to be used in the software factory context?
- Why using DSL instead executable models in UML + Semantics?
- How many successful case exist using DSL in the industry?

The results gained so far have been varied. There have been some successes, but the challenges and difficulties encountered during the course of the project have also been interesting. Small companies are unique in their need to keep up with market trends, and succeed in every project that they undertake. Unlike larger companies, and even single project teams within a large organization, a small company cannot afford to fail in any project, because the livelihood of the company, and every employee, depends upon keeping their market share. In the company with which this project was associated, their business was dependent upon a single product. If that product failed, then the company would cease to exist. This does not compare with even isolated parts of a large company, because although the project may fail and cause difficulty within the company, this would not cause the collapse of the business. The stakes are much higher in a small company, and their willingness to take unexplored risks is much smaller.

Some success has been achieved in this study thus far; however, it could be considered that much of this success can be attributed to the new technology which the company adopted. New technology will only provide a platform for making improvements. It is only when the opportunities provided by this technology can be identified and exploited that benefit will be gained. In exploiting these opportunities, there were two major problems which were identified during the course of development. These were the lack of experience in using an object oriented design method, and the lack of tool support for the developers. It was felt that with suitable tools, object oriented development could be better employed, and it would be easier to collect written code into a reuse repository for use by other applications. Based on the above said technique HP (Hewlett Packard) realized that software reuse plays an important role in increasing efficiency of developer and helps in decreasing the cost and time of development of software. It was found that with adoption of reuse 6 %to 40% increase in productivity was observed and 12 %to 42% less time was required for the development. In spite of this quality was improved by a factor 24% to 76%. It also helps the specialists to

optimize software architectures and assets that may then be reused by others who are focusing on meeting product features and markets needs. Along with these factors reuse provides consistency and interoperability across products.

CONCLUSION

The above article describes how reuse is effective in current scenario of software development in spite of having number of hurdles. The results gained so far have been varied. There have been some successes, but the challenges and difficulties encountered during the course of the project have also been interesting. Small companies are unique in their need to keep up with market trends, and succeed in every project that they undertake. Unlike larger companies, and even single project teams within a large organization, a small company cannot afford to fail in any project, because the livelihood of the company, and every employee, depends upon keeping their market share. All this lead to poor reuses activities in most software development organizations. One of the main reasons for the low reuse activity is that developers simply do not attempt to reuse because of a number of factors, including lack of knowledge about reusable assets and the notion that the cost for reusing is higher than the cost of developing new code. Reuse plays an important role in increasing efficiency of developer for producing large product within limited amount of time and with less risk, high quality. At this technique is adopted by large companies. Though small organizations has also started to develop software by adopting reuse, but still most of these organizations afraid, because development with reuse is costly. As for this specialist persons and large repositories are required, which causes extra cost to the organizations. Various tools and techniques have been developed to achieve the reusability. Hence it is concluded that in future reuse will play a vital role in development and will be the key term.

REFERENCES

[1] Lanergan R. G., and Grasso C. A., (1984), "Software Engineering with Reusable Design and Code," *IEEE Transactions on Software Engineering*, **10**(5) pp. 498-501.

[2] Prieto-Diaz R., (1990), "Implementing Faceted Classification for Software Reuse," In: *Proc.of 12th International Conference on Softwware Engineering*, IEEE, Nice, France, pp. 300-304.

[3] Matsumoto Y., (1989), "Some Experiences in Promoting Reusable Software: Presentation in Higher Abstract Levels," In: *Software Reusability. Concepts and Models*, **II**, Biggerstaff T. J., Perlis A. J. (ed.), ACM Press, Addison-Wesley, Reading, Mass.

[4] Tracz W., (1988), "Software Reuse Maxims," *ACM Software Engineering Notes*, **13**(4), pp. 28-31.

[5] M. Friedewald *et al*., (2001), "Status of the Software Development Industry in Ermany," In forMatik Spektrum, **24**(2), pp. 81-90.

[6] P. Louridas, (2006),"Using Wikis in Software Development," *IEEE Software*, **23**(2), pp. 88-91.

[7] B. Leuf and W. Cunningham, (2001), The Wiki Way: Quick Collaboration on the Web, Addison-Wesley.

[8] [3] S. Lauesen, (2002), Software Requirements: Styles and Tech-niques, Addison-Wesley.

[9] Boehm B., (1999), "Managing Software Productivity and Reuse," Computer, **32**(9), pp. 111-113.

[10] [2] Crnkovic I., Larsson S., and Stafford J., (2002), "Component-Based Software Engineering: Building Systems from Components," ACM SIGSOFT, **27**(3), pp. 47-50.

[11] Bacili K., and Oliveira M., (2006), "DigitalAssets Manager: Sharing and Managing Software Development Assets," *OOPSLA'06 Demo Session*, ACM, NY, pp. 700-701.

[12] Oliveira M., Garcia I., and Nunes A., (2005), "RCCS: Uma Rede De Compartilhamento De Componentes De Software," Brazilian Symposium of Computers Networks (SBRC), Fortaleza, Brazil.

[13] Zhang Qiu-yu, and Zhang Dong-dong etc., (2004), The Research and Application of Special Domain Software Reuse Technology *[J]. Computer Engineering and Application*, **40**(14), pp. 213-215.

[14] Li Ji-hong, (2004), The Software-reuse Technology Based on Reuse Component *[J]. Shanxi Coal Management Cadre Institute Transaction*, **17**(3), pp. 109-110.

[15] Hafedh Mili, and Ali Mili, etc., (2004), Reused-based Software Engineering – Techniques, Organization, and Controls [M]. Beijing: Electronic Industries Publishing Company, 1.

[16] Arango G., and Prieto–Diaz R., (1989), Domain analysis: Concepts and Research Directions [M], IEEE Computer Society Press.

[17] Microsoft Corporation, (1993), "OLE 2 Classes for the Microsoft Foundation Class Library," Microsoft Corp.

Anil Kumar
Guru Nanak Khalsa Institute of Technology and Management, Yamuna Nagar
E-mail: thakral8@gmail