December 1998

Knowledge Reuse - Insights from Software Reuse

Il Im
University of Southern California

Alexander Hars
University of Southern California

Knowledge Reuse — Insights from Software Reuse

Il Im
Alexander Hars
Department of Information and Operations Management
Marshall School of Business
University of Southern California

Abstract

We are moving towards an economy where competitive advantage will be determined by knowledge. In their knowledge management strategies, many companies currently aim to encourage knowledge reuse. In this paper, we examine insights drawn from a related field — software reuse — for their relevance to the emerging field of knowledge reuse. We first examine different types of reuse: components, patterns, frameworks and general principles. We then evaluate different kinds of reuse activities. Finally, we discuss lessons from cultural issues in software reuse.

Introduction

There is a widespread consensus that we are moving towards an economy where competitive advantage will be determined by knowledge rather than by access to raw materials and cheap labor [Liebeskind, 1996]. Business processes are becoming more and more knowledge-intensive while the time given to an organization for creating or acquiring knowledge is becoming more limited. Organizations are forced to create more knowledge in shorter time period. As a response to this demanding environment, some companies have created new positions such as "Chief Knowledge Officer" whose mission is increasing the knowledge base of their workers so that everyone can make better decisions [Carrillo, 1997].

Knowledge management is "getting the right knowledge to the right people at the right time so they can make the best decision" [Hibbard, 1997] — or "an attempt to put processes in place that capture and reuse an organization's knowledge so it can be used to generate revenue" [Dash, 1997]. As implied in these definitions, one of the ultimate goals of knowledge management is "reuse of knowledge" through sharing of knowledge among the members. Thus, shedding light on knowledge in the reuse perspective is needed.

In the software engineering area, valuable "reuse" experience already exists in the form of "software reuse". Since software can be thought as a type of codified knowledge, investigating the efforts for software reuse and understanding what is relevant to knowledge reuse could provide valuable insights for knowledge reuse. In this paper, the insights from software reuse are discussed.

Software as Codified Knowledge

One of the primary reasons of implementing knowledge management efforts in large companies is to promote sharing of codifiable knowledge such as transfer of best practice and customer/market information [Dash, 1998]. Codifiable knowledge is a primary target of knowledge management because it is feasible and easier to share than less codifiable and tacit knowledge. This paper therefore focuses on codifiable knowledge.

Software has many similarities with codified business knowledge. First, both are accumulations of rules for works. The are often represented in an event-action format such as: "If event X occurs then perform action Y". Secondly, different software often needs exactly identical components, for example, sorting algorithms and user interface components, in several different contexts. Similarly, companies need identical knowledge such as guidelines for action or document templates in several different contexts. Thus, opportunities for the reuse of software as well as business knowledge clearly exist. Thirdly, there might be multiple alternatives such as algorithms in software and ways of solving a problem in business, that have the same goals but are different in approach, efficiency and/or effectiveness. Thus, it is likely that the quality of software or business knowledge can be improved by selecting better software or knowledge components.

From a knowledge management perspective, software can be considered as "a special type of knowledge that is captured and frozen into a specific representation format (text or binary code) ".

Types of Reuse

Although there is no internationally recognized standard for how software reuse should occur [Rada and Moore, 1997], there have been many efforts to make software reuse more efficient and universal. There are three different levels of reusable software components [Tibbetts and Bernstein, 1998; Johnson, 1997] – components, frameworks, and patterns. Components are "black box" pieces of software that can be plugged into a program with minimal modification efforts. Frameworks are libraries of software that work together to serve a particular end. Patterns are completely conceptual. A pattern is a description of a problem, a sketch of its solution, and the context in which that solution works.

In addition to these three areas of software reuse, there is another software reuse area – general programming principles, which is usually taken for granted. These four areas can be mapped along two dimensions — level of abstraction and range of problem area as shown in Figure 1.

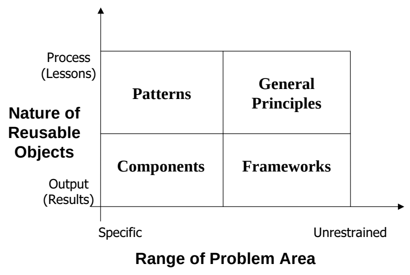

Figure 1. Reuse Map

The **range of problem area** means "the extent to which an object to be reused is applicable to different areas". The **nature of objects reused** represents "the types of objects reused – whether outputs are reused or processes/lessons are reused". For example, both components and patterns deal with specific problems but, components are outputs of the problem solving activities while patterns are lessons that have been discovered during problem solving processes. The dimensions are not discrete, which implies that in some cases the output and process of a problem solving activity are reused simultaneously and that there is a continuum in the range of applicable problem areas.

This map can also be considered for knowledge reuse. Organizations typically have a large amount of reusable knowledge that can be distinguished along both dimensions. In the knowledge reuse perspective, the examples of each quadrant are:

- Components – document templates, engineering drawings, last-year s tax report. These objects are output of problem solving activities and can be used in narrow problem areas.
- Frameworks — market report, ISO 9000, manuals. These objects cover a wide range of problems and are also outputs of problem solving activities.
- Patterns — best practices, worst practices, golden rules for a certain type of task. These objects are lessons learned during problem solving activities and applicable to narrow problem areas.
- General principles — general business principles, organizational vision, wise-sayings. These objects are processes/lessons that are applicable to broad areas.

In the following section, two areas of software reuse — component reuse and pattern reuse where a lot of research efforts have been devoted — are reviewed and implications for knowledge reuse are discussed.

Component Reuse

As object-oriented programming (OOP) is gaining popularity, software reuse by component-based development (CBD) is receiving attention. There are two key requirements of component-based development [Williamson, 1997b] :

- Communicate via an open interface.
- Existence of a component repository. A repository is a place to store information about a company's components, that also links to the components themselves.

In component reuse the representation methods for software components in the repository are critical [Frakes and Pole, 1994].

The problem in applying these concepts to knowledge reuse is that standards are needed for linking knowledge. Because knowledge is not executable ("conceptual"), this may be more difficult than software reuse. In addition, standards are needed for the structure of a knowledge component so that it can be added to a repository and retrieved. Thus, the use of templates is promising and classification approaches from library science etc. could be useful. Currently many companies are focusing on solving these repository-related issues for knowledge management implementations. A more difficult problem is the interoperability between knowledge components.

Pattern Reuse

There have been many efforts to develop robust design patterns for software reuse. "View" proposed by Novak [1995] and Cowan and Lucena [1995]'s abstract data view (ADV) and abstract data object (ADO) are part of those efforts. Although patterns do not contribute reusable code, they recycle something at least as important: design recoveries. Thus, teams with pattern experience are unusually productive and good at communicating within and across software development teams and facilitating training of new developers [Tibbetts and Bernstein, 1998; Schmidt, 1995].

Similarly, developing knowledge patterns will help companies train new employees and improve communications. However, because of the indirect benefits of pattern reuse, resources allocated to pattern reuse might be difficult to be justified.

Different types of software reuse approaches have different purposes and results. For example, the main purpose of component reuse is efficient software production, while pattern reuse aims at better future software development and developer training.

The purpose and type of knowledge reuse also need to be carefully analyzed before actions are taken. As discussed in software reuse the focus and activities will be different depending on the purposes of knowledge reuse and the types of knowledge to be reused.

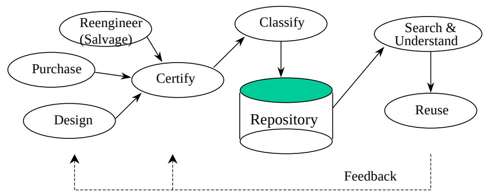

Figure 2. Software Reuse Cycle
(Adapted from Arnold and Frakes, 1991)

Activities for Repository-based Reuse

Arnold and Frakes [1991] propose a process for acquiring and reusing software parts as in the Figure 2. The main activities are:

- Reengineering: This is for recovering software assets for reuse.
- Certification: Certification is the process of establishing properties about assets - for example their correctness or timing behavior.
- Classification: Reusable assets are eventually put into a repository of some sort, and must be classified so that they can be found and understood.

This model provides an understanding of the procedures of repository-based software reuse. These activities are also applicable to knowledge management because a large portion of current knowledge management efforts are focused on building good knowledge repository [Anthes, 1998].

When companies plan to build a knowledge repository, they need to evaluate the challenges they might face. Certification is a major problem because it is not always easy to exactly measure the value and impact of a piece of knowledge to be added to the repository. If too many contributions are added to the repository, then users have an additional burden of evaluation, which will decrease the utility of using knowledge repository. There are some efforts to solve these evaluation problems in knowledge repository [Hars et al., 1996; Tsang & Hall, 1997].

Classification is also a potential problem, but many companies that built their own knowledge repositories seem to have less difficulty in classification than certification [Anthes, 1998]. Searching usually is not that difficult due to the availability of increasingly powerful and sophisticated search engines.

Company-wide Policies and Incentives

A significant inhibitor for software reuse is the "not invented here" (NIH) syndrome [Williamson, 1997a]. Developers are not sure whether they can trust software from the outside; they tend to underestimate the time and effort it takes to develop something from scratch and it may require substantial effort to find and test suitable software. In addition, developers fear integration problems. Another problem is communication among people in organizations. It is not unusual for distinct departments and business units to have their own vocabularies, values and ways of doing things [Williamson, 1997a]. Thus, software reuse might be difficult without improving communication between different functional units.

This problem may also occur when companies attempt to reuse their knowledge. People might have similar hesitation in reusing the knowledge from others, and it may not fit with the current perspective. The importance of common communication language has also been pointed out for knowledge management [Grant, 1996]. These problems can be referred to organizational culture . Recent survey shows that managers also recognize corporate culture as the biggest obstacle to knowledge management [Cole-Gomolski, 1997]. For successful knowledge reuse, companies need to encourage people by changing the culture and providing incentives to share knowledge with others.

Costs and Benefits

Software reuse does not come without cost. For a midsize to large application development organization (with 100 to 500 developers), first-year cost of reuse program may range from $535,000 to $1.15 million [Hunter, 1997]. The cost for knowledge reuse might be no less than that. Meanwhile, most of the expected benefits from knowledge management are intangible such as innovation , improved decision making , and flexibility [Hibbard, 1997]. These benefits are also difficult to measure.

It is to be expected that successful knowledge management should yield economic benefits. Moreover, as Frakes and Fox [1995] showed in software reuse, reuse is successful when people perceive economic value of reuse. Therefore, the success of knowledge reuse will be affected not only by actual economic benefits but also by perception of economic benefits. Managers need to develop measures for the impact of knowledge management. These measures are necessary for both tracking the actual benefits and communicating with the members on the benefits of knowledge management.

Conclusion

A company is, in a sense, an organization to integrate knowledge of its members and to convert it into economic value. Of the various different types of knowledge in companies, codifiable knowledge is the most easily feasible to manage. In this paper we investigated software as a kind of codifiable knowledge and drew insights for knowledge management from software reuse.

Knowledge reuse is one of the important aspects of knowledge management. For the repository-based knowledge management that mainly focuses on reuse of codified knowledge, software reuse provides many useful insights. Managers need to understand different types of knowledge to be reused, and devise procedures of knowledge reuse appropriate for their company. Policies and incentives to promote knowledge sharing will be essential for successful knowledge management. Knowledge management will also require financial justification and accurate measures.

References

References are available upon request from first author.