UNIVERSITY OF GOTHENBURG

Software Reuse And Offshoring:

A Study Of Benefits, Difficulties And Feasibility

Hui Zhou
Monan Yao

Bachelor of Applied Information Technology Thesis

Report No. 2010:035
ISSN: 1651-4769

University of Gothenburg
Department of Applied Information Technology
Gothenburg, Sweden, May 2010

Abstract

CONTEXT – Software Reuse and Offshoring are two promising approaches to decrease cost and increase productivity in software development. There have been many researches made in various perspectives on both subjects but none of them have forced on the relationship between them.

OBJECTIVE- The objective of the research is to explore the understanding of software reuse and offshoring, and to find out the possible outcomes of applying these approaches simultaneously in software development.

METHOD- Literature review is the main source of the research.

RESULT- It is very challenging to apply either of the two approaches in software development. When implementing both approaches at the same time, it requires tremendous efforts and up-front investment to make the change, and the result of integration would lead to increasing managerial difficulties.

CONCLUSIONS- By studying Software Reuse and Offshoring, we have found the potential benefits, difficulties and feasibility when combining both approaches in particular cases of software development.

Preface

Our sincere thanks to Faramarz Agahi, for help and guidance throughout the project. Last, but not least, we wish to thank all the interviewees who participated in this project.

Table of Contents

1.Introduction

Software engineering is an engineering discipline focusing on the cost-effective development of high-quality software systems (Sommerville 2006). In different organizations, researchers and scholars have devoted countless time and effort to numerous strategies, methods, and theories that support building high-quality software with lower budgets. Offshoring is one of these approaches and it is a growing phenomenon that takes advantage of globalized manpower towards higher productivity and lower cost in software development (ACM JMTF 2006). Nowadays many software products are regarded as valuable asset in companies. In order to increase the return on the investments, companies promote software reusing for lower software production and maintenance costs, fast delivery of systems and increased software quality (Sommerville, 2006). Therefore, software reuse can be considered as another approach towards the mutual benefits in software engineering.

As both two approaches can effectively reduce cost, enhance the productivity of software development, they can be compared to the "catalyst" in chemical reaction. Since either of the two can dramatically improve software development, it might be possible to multiply the benefits of both approaches by applying these two approaches parallelly in software development. Then it triggers the question of the research: ***What is the outcome of applying software reuse and offshoring simultaneously in software development?***

The objective of his paper is to explore the outcome of this experiment. Software is abstract and intangible. There comes rising difficulties to investigate the apporaches, since these appoaches are at least equally abstract and intangible as software. Therefore, the research is taken in a step-by-step fashion. The first step hereby is to gain an intensive understanding of software reuse and Offshoring. Both subjects are generally complex and sophisticated, and it requires extensive research on both subjects to gather constructive information to establish the theoretical framework for later research and discussion. The second step is to form the theoretical framework by choosing specific theory or model within the related research. The theoretical framework of the research consists two models that are reasonably selected from the approaches. Together both models represent the theoretical framework. The third step is data collection and data analysis. Data collecion covers the methods that have been used to retrieve data in the research, while the data analysis is based on the criteria defined by the theoretical framework. The final step is comprised of discussion and conclusion discovering the potential benefits, difficulties and feasibility of applying the theoretical framework in software development.

2.Related Research

2.1.Software Reuse

Reuse is often described as not “reinventing the wheel” and the first step at succeeding at reuse is to understand that you have more than one option at your disposal (Ambler 2005). Software reuse which is the reuse of existing software, or software knowledge in order to build new software. Table 1 clearly shows different categories of reuse.

Reuse Category	Examples
Architected	Service domains Domain components Internal open source
Pattern	Architecture patterns Design patterns Analysis patterns
Framework	**External open source Technical frameworks Business frameworks**
Artifact	COTS application Legacy application Domain model
Module	User interface components Technical components Web services
Template	Use case template Project plan template Document template
Code	**Class libraries Functional libraries Copy & paste**

Table 1 Types of reuse in information technology (Ambler 2005)

There are four main reuse types existing in industry software development. Opportunistic reuse which means while the development team starts a project, developers find that there are existing components that they could reuse directly. Besides that, planned reuse is one kind of reuse that a developers strategically design components which are reusable in future projects. Internal reuse and external reuse are normally implemented by individual programmer and organisations. Individual programmer reuses its own components which might be easier and quicker for the internal software development. External reuse always implemented by a team who may choose to license a third-party code. Licensing a third-party code typically would reduce the cost. This type of reuse mostly benefits from the Free/Open software development

2.1.1.Advantage

Cost-saving

> *"Difficult economic and market conditions are forcing software development teams to do even more with less, and to become even more responsive to customer needs."(Eran Strod, Quantifying the cost savings of using open source in software development)*

In software development industry, whether an organization relies on traditional software development methodologies or more flexible agile development processes, currently the reuse of software or source code is the key to dramatically speeding software development and lower the cost of it at the same time. For instance the use of open source code in reality. Table 2 expresses the cost save specifically by the comparing two different development methods.

Lines of Code	100000
Finished lines of code developed/day	20
Work days/year	222
Development staff years to complete	22.52
Developer cost per year	84660 USD
Savings by using software reuse(open source code)	1906757 USD

Table 2,Lines-of-Code Cost Calculator (BDS 2009)

There are fundamental cost aspect in the creating phase of software development: Firstly, the cost of developers time (Full time employees) Beside that, the average number of debugged lines of code produced and related costs such as maintenance, documentation. In this case, open source software has the potential advantages to save developers from reinventing the wheels. The obstacle of the open source software reuse is how to properly manage it according to corporate policies and procedures.

Benefit	Explanation
Increased dependability	Reused software, which has been tried and tested in working systems, should be more dependable than new software because its design and implementation faults have already been found and fixed.
Reduced process risk	The cost of existing software is already known, while the costs of development are always a matter of judgement. This is an important factor for project management because it reduces the margin of error om project cost estimation. This is particularly

	true when relatively large software components such as sub-systems are reused.
Effective use of specialists	Instead doing the same work over and over, these application specialists can develop reusable software that encapsulates their knowledge
Standards compliance	Some standards, such as user interface standards, can be implemented as a set of standard reusable components. For example, if menus in a user interface are implemented using reusable components, all applications present the same menu formats to users. The use of standard user interfaces improves dependability because users are less likely to make mistakes when presented with a familiar interface.
Accelerated development	Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time should be reduced.

Table 3, Figure Benefits of software reuse(Sommerville 2006)

More than code

Actually, although the practice is called code reuse, much more than code could be carried in reuse libraries. For instance assets could include Offshoring business-process rules, best practices, test cases, development models, patterns and specific code at all levels. Companies as well as individual programmers are increasingly seeing the benefits of reusing. Actually, there exists disadvantage of software reuse which needs to be considered.

2.1.2. Disadvantages

problem	Explanation
Increased maintenance costs	If the source code of a reused software system or components is not available then maintenance costs may be increased because the reused elements of the system may become increasingly incompatible with system changes
Lack of tool support	CASE tool-sets may not support development with reuse. It may be difficult or impossible to integrate these tools with a component library system. The software process assumed by these tools may not take reuse account.
Not-invented-here syndrome	Some software engineers prefer to rewrite components because they believe they can improve on them. This is partly to do with trust and partly to do with the fact that writing original software is seen as more challenging than reusing other people's software.
Creating and maintaining a component library	Populating a engineers prefer to rewrite and ensuring the software developers can use this library can be expensive. Our current techniques for classifying, cataloguing and retrieving software components are immature.
Finding, understanding and adapting reusable components	Software components have to be discovered in a library, understood and, sometimes, adapted to work in a new environment. Engineers must be reasonably confident of finding a component in the library before they will make include a component search as part of their normal development process.

Table 4,Problems with reuse (Sommerville 2006)

Hidden Costs

In reality, large-scale code reuse brings up exceptions than developer expecting. On the other hand, using reusable objects requires extensive analysis as well as plan. "*It is really hard to measure the reuse. And it do depend on how big project it is. Actually it is hard to see how much time or budget u have save during the project developing. it is also quite impossible to set up the goal and timetable about the reuse work in the beginning of the project.*" said Henrik, a software development professor in Sweden. Normally, the developer will need to invest extra time in testing and quality assurance and documentation to make sure the reused code is able to fulfill the functionality. All of this, as listed in the table 4, takes time which increases the hidden cost of the code. It also depends on the size of the development team as well as the organization construction.

Efficient Project Management(Team efforts)

There is a big obstacle impedes the project team to cooperate in achieving reuse in the first place. Few project managers are willing to spend their time, resources as well as project development risks to deal with their deadlines. Because of that, it is hard to be magnanimous and make the project developers work well enough for reuse. In this case, software offshoring environment provides chance for the organisation to deal with software reuse because of the mature management background.

Architecture Needed

Developers working with visual tools which are able to bring up components in quicker and easier way, but without a foundation of good architecture, it is most unlikely the components would work for the new application.

> *"Most projects have small start and when they get bigger the code turn into a mess. But personally i will agree that good structure will make reuse and maintenance easier in later phase of the software development."*
>
> *-Henrik*

Ultimately, a standard architecture would possibly help both vendors and individual programmers solve the reuse dilemma. This will force developers to spend more time thinking about the reusability than wrestling with current implementation. Many organizations prematurely give up on reuse when they don't get positive returns on their few projects or even on their very first one. Strategic reuse discipline is a long-term endeavor that has strategic returns instead of tactical ones (Ambler 2004). The goal of the discipline is to define how organizations could succeed at reuse by the architecture implementation. More in-depth information of this discipline will be expressed in the follow section as well as theoretical framework .

2.2.Software Offshoring

2.2.1.Definition And Terminology

According to Cambridge Advanced Learner's Dictionary (CALD 2010), Offshoring means "the practice of paying someone in another country to do part of a company's work". It seemingly resembles another term *Outsourcing* described as "if a company outsources, it pays to have part of its work done by another company". The Panel of National Academy of Public Administration in U.S. (PNAOPA 2006) has suggested more explicit definitions for these terms as followed:

> "***Outsourcing***— *firms contracting out service and manufacturing activities to unaffiliated firms located either domestically or in foreign countries.*
>
> ***Offshoring*** *— U.S. firms shifting service and manufacturing activities abroad to unaffiliated firms or their own affiliates.*
>
> ***Offshore Outsourcing*** *— a subset of both outsourcing and Offshoring in that it refers only to those service and manufacturing activities of U.S. companies performed in unaffiliated firms located abroad. (PNAOPA 2006)*"

Although these terms are clearly defined, it is still difficult to see through the differences by a glimpse, especially Offshoring and Offshore Outsourcing. The following sections will present the extensive research on these two terms exploring the similarity and difference between them.

2.2.2.Offshore Outsourcing

Offshore Outsourcing is the outsourcing process only contracting vendors outside of the border. In order to study Offshore Outsourcing in depth, it is approachable to begin with its ancestor, outsourcing. Outsourcing has been universally applied around world in many kinds

of business to attain the goal of cost-saving. It creates a win-win situation that brings both sides of the contract great values (Lee et al. 2003). As increasingly growing business or a scientific topic, outsourcing has been intensively studied in numerous perspectives such as economics, organisational sociology and management, but the process issues of outsourcing have been unknowingly neglected (Ring & Van deVen 1994). Zhu et al. (2001) has made research on the process view of outsourcing and defined a model illustrating the stages in the process.

The first stage is the "Planning" stage creating a sound business plan that covers cost of all outsourcing-related activities. The second stage "Development" includes choosing outsourcing vendor, making agreements, measuring the impact and benefits and making communication plan. The "Implementing" stage involves creating a transition plan and checklist preparing the project transferal to the supplier or third-party. The final stage "Surviving" involves assessment of the result of outsourcing to determine if the objectives were attained. It can be seen that this model is driven from the outsourcer's perspective which only shows the activities the outsourcer operates during the process. According to the process, the outsoucer can easily decrease the development cost by choosing the outsourcing vendors that offers less price.

The Zhe et al.(2001)'s model is originated from general business background, which might not be exactly the same model in software outsourcing. Many organizations have studied the process issues in Offshore Outsourcing software development, and some of them are outsourcing vendors. These vendors have dedicated great efforts to create appealing process models for business presentation. BelHard is one of the vendors from Belarus. BelHard's Outsourcing models shows that in software Offshore Outsourcing there are more involvement between the customer and the vendor. Except for identification and confirmation of the requirements, the outsourcer is requested to supervise the project during the whole software development cycle and resolving risks in the early stages. However, the communication between onsite and offshore is only through both project management team, there is no sign of direct communication between the development teams from two sides.

2.2.3.Offshoring

Offshoring describes the process that companies relocate or shift their services or manufacturing activities to low-cost destinations. Originally Offshoring is applied as a way to reduce cost, but in resent years researchers and scholars have revealed that Offshoring is not just a way to get bargain, it has made business entering a globalized era (CIBER 2006). Besides taking advantage of cheap labor at low-cost countries, the Offshoring process gives access to more talented people and their gifts. CIBER (2006) called this phenomenon as "next-generation Offshoring". Labor arbitrage or global labor arbitrage is the foundation of the next-generation Offshoring where results in the removal of barriers to international trade, moving low skilled jobs to low-cost nations while moving high skilled jobs to nations with higher pay (Roberts 2004). As one type of offshore software development, Offshoring can reduce the cost of development, provide abundant quality human resources and create an efficient platform to integrate separated talents.

NAOPA's definition of Offshoring includes both unaffiliated and affiliated firms as Offshoring destinations. However, Offshoring most generally refers to the relocation inside a single multinational firm, which is a subset Offshoring defined as "in-house Offshoring" or "offshore insourcing" (Prikladnicki et al. 2007). Olsson et al.(2008) have gone further in this area defining the term "two-stage Offshoring". The definition of Two-stage Offshoring is quoted here:

> *"a company offshores to one location, which then offshores work further."*
>
> *- Olsson et al.*

They also found out that the shifting can keep

moving from a low-cost destination to a even lower one, a term "multi-stage Offshoring" is suggested hence. The two-stage Offshoring model creates a bridging site between the high-cost site and low-cost site, but the function of the bridging site is defined differently. According to Olsson et al.'s case study, the bridging site is regarded as the management or communication center between the two sites. But in any of the cases, the bridge makes contributions to decease the cost overall development in the organisation. And this discovery enables in-house Offshoring to achieve lower cost in software development without involving outsourcing (Olsson et al. 2008).

2.2.4. Offshore Outsourcing V.S. Offshoring

There are some similarities between Offshore Outsourcing and Offshoring. First, they all require offshore location to deliver tasks. Second, both methods usually choose low-cost locations to shift tasks. Third, in either case Offshore Outsourcing or Offshoring, the company requires a reasonable scale of business to conduct these operations. And finally, both methods encourages globalization in a variety of perspectives.

The major difference between the two is the points of view. Olsson et al. (2008) explained that Offshoring is about the location of the main operation ,while outsourcing is about who takes charge of the operation. There is no direct opposition or contradiction between these two methods, there is possibly an intersection between the two approaches.

For outsourcing, Olsson et al. (2008) has revealed its two implications, either contracting parts of the process or tasks outside of the firm or delegating the entire process to an outsider. And it seems that delegating the entire process to an outsider is very similar to the Offshoring that shifts service or manufacturing abroad to unaffiliated firm. The outsourcing vendor BelHard has also identified full project outsourcing as an exception to their Software Project Outsourcing Model (BelHard 2010), they declares this due to the possible hidden risks (BelHard Outsourcing 2010). Or it can been seen in this way, when BelHard get a full project, BelHard is actually delegate the entire development of the project. While the outsourcer of BelHard just simply pass the project from to them, which is very likely an Offshoring process.

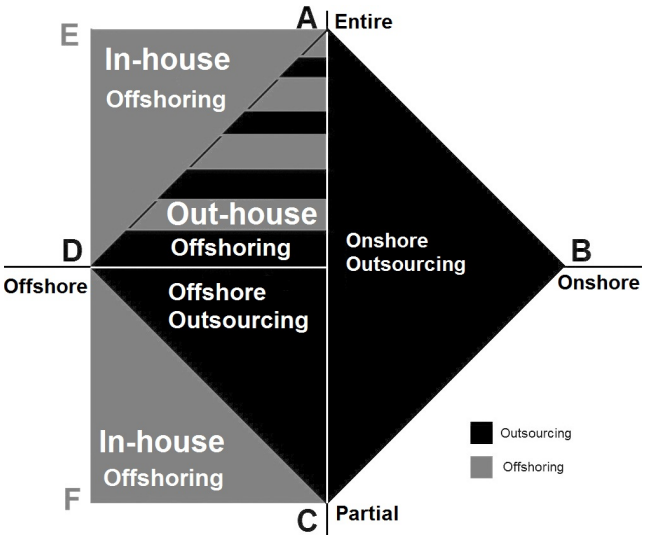


Figure 1: Intersection of Offshore Outsourcing and Offshoring

In another word, hypothetically, Customer asks Company A to develop Project X. Then Company A contracts an entire Project X abroad to Company B. When Company B finished the development, it passed the project to Company A, then Company A presents Project X to Customer. This process can be rephrased as company A Offshore Outsourcing or Offshoring the project X to Company B. Therefore, there is an possible intersection between Offshore Outsourcing and Offshoring as Figure 1 illustrated. In addition to this point, the there is another model showing close connection with both methods which is defined as nearshore outsourcing. Nearshore outsouring is a mode of offshore outsourcing reliant on geographical proximity between client and vendor countries (Abbott 2007). Like offshoring, this mode decreases the difficulties in communication and coordination by taking advanatage of the shorthen distance between two sides.

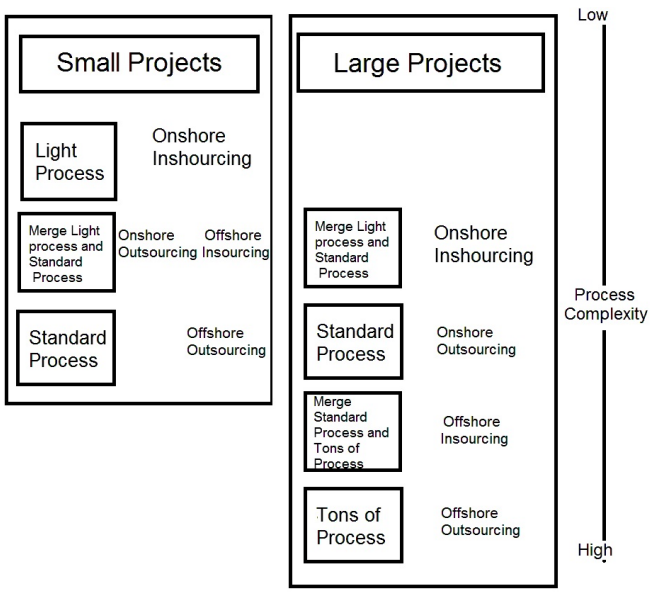


Figure 2, Process Compexity in differnt prjects (Prikladnicki et al. 2007)

Despite the intersection between offshoring and offshore outsourcing, it is revealed that offshore outsourcing is more complex than in-house offshoring in the perspective of process (Figure 2), which means in-house offshoring may be more efficient for software development at offshore destinations than offshore outsourcing. With similar advantages of outsourcing, two-stage offshoring is a promising approach to achieve low cost and efficient software development at offshore destination.

2.3.Software Reuse & Offshoring In IBM

Software reuse and offshoring are almost two unrelated subjects in software engineering. In order to explore the relationship between the two, it was decided to find an organization that has involved with both approaches in the system. IBM is one of the companies that have devoted tremendous efforts on both subject, which could bring valuable insights for the further research and discussion.

2.3.1.Description

From the merge of three companies back to the 19 century to the gigantic enterprise with almost 400,000 employees in over 200 countries, IBM is unadulterated a Globally Integrated Enterprise (Palmisano 2006). As one of the world's largest technology company, IBM covers a variety of businesses such as manufacturing and selling computer hardware and software, infrastructure services, hosting services, and consulting services etc. (IBM 2010). IBM has been actively involving with offshoring for many decades and have resulted in a massive complex multinational structure.

The reason of introducing IBM to the related research is originated by the the company's great achievement in these two areas. It is evident that IBM have reached the stage of Next-generation offshoring described by CIBER (2006) and taken advantages from talents all over the world with sophisticated labor arbitrage system. On the other hand, software reuse is heated subject in the company. There are have been numerous models and theories created to illustrate the importance of software reuse as well as the practical approaches that implements the theories. As mentioned above, systematic reuse requires sophisticated plans to carry out, and these signs can be spotted in IBM.

However, IBM is not quite transparent in terms of organisational structure. There are no brief documentations clearly showing how this gigantic company operates locally and internationally. Knowing IBM's attempt on patenting offshoring (USP&TO 2007) it was realized that IBM may not generously present their offshoring workflow freely on the website. In order to find how offshoring was managed in IBM, the research is carried out mainly based on reviewing published documents, brochures, articles and web-pages on IBM's websites.

2.3.2. Offshoring In IBM

As one of the largest IT company in the world, the structure of IBM is rather sophisticated and somehow cryptic where there is no clear illustration of the relationship between different departments and how they interact.

By analyzing the services provide by IBM, there are mainly two types of offshore sites in IBM: subsidiaries of IBM divisions and

affiliated firms, and some of the affiliated firms have been integrated with the divisions. For example, IBM Rational or IBM Rational Software was acquired by IBM in 2003, this affiliated firm became part of the IBM Software. It leads to the notion that theses affiliated firms are working independently as peers under the overall structure, which is quite similar to one case in Olsson et al.'s case study on Two-stage Offshoring.

When comes to offshore destinations, it appears that IBM has at least two separate divisions in each country. In China, IBM owes two firms namely IBM Global Service and IBM China. The IBM Global Service division covers many services such as business consulting, technology consulting and outsourcing. While IBM China manages the Delivery Centers in China (Appendix II). The Delivery Center or Global Delivery Center (GDC) is one of the fundamental units in IBM, GDC is where most labor gathered and participate software and hardware manufacturing. Global Service, on the other hand, is dedicated to consulting in various areas and system integration (IBM 2010). It feels that the Global Service department plays the role as the bridging site which is related to management and coordination.

IBM has set up 41 GDCs in 16 countries, and most of these countries are emerging countries providing vast amount of labor (IBM Global Briefing 2010). There is another kind of delivery center named as "Integrated Delivery Center (IDC)" in IBM. These centers are also considered as strategic location or strategic delivery center (IDC in Brno 2010). IDC is not only an offshore development center, but also get close to IBM's clients providing solution with IBM's integrated technology. By comparing the GDC and IDC with Two-stage offshoring, there are some similarities. In Two-stage offshoring, the bridging site focuses on management or some sort of the integration of the two other sites. While IDC does not directly managing GDC, but in some direction it integrates the assets developed on different GDCs. Also IBM has numerous research centers around world, and these research centers facilitates other divisions in terms of economical and technological resources, and these research centers can also be regard as some source of bridging site managing research-based resources.

2.3.3. Software Reuse In IBM

software reuse has always been a heated topic in software industry, and especially for big company like IBM. There are some prominent theories invented inside IBM describing how to manage systematic software reuse. A team of specialists in IBM have written a book titled *Strategic Reuse with Asset-Based Development* (Ackerman et al. 2008). This book has introduced a strategic reuse approach named Asset-Based Development, and made intensive research on the subject. This approach is very related to Ambler's SRD models. If saying Ambler's models are the blueprint of the systematic reuse in IBM, then Asset-Based Development is the instruction book explained how it works. Basically, Asset-Base Development is a component-based development customized by IBM. This approach requires a specific development infrastructure that includes planing sectors, management sectors and various tools that facilitate the development. The company that adapted the approach has to invest significantly on changing the development infrastructure as well as adapting or purchasing tools from IBM. But the benefits are evident. Asset-Based Development has been widely applied in IBM for example services in Service-Oriented Architecture(SOA) (Balaji 2007). And it has many success factors (Appendix IV) to be considered in order to implement the method.

3.Theoretical Framework

3.1.Two-stage Offshoring

Two-stage offshoring is a very promising approach that can greatly decrease development cost while take advantage of the organisational benefits. However, Two-stage Offshoring is not an approach based theoretical inferences, but discovered empirically. Since Two-stage

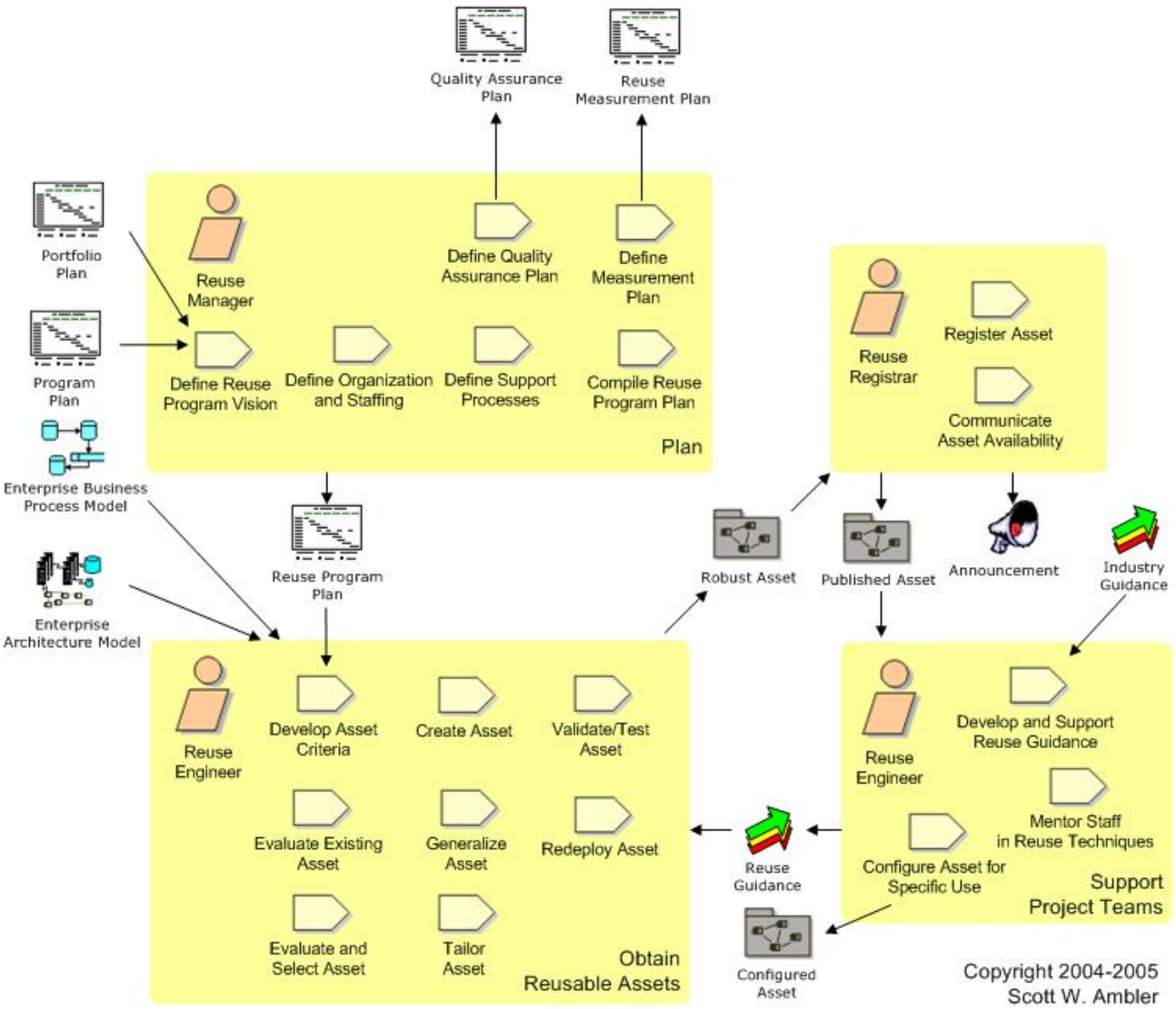


Figure 3: The amalgamated workflow of the Strategic Reuse discipline (Ambler 2004)

Offshoring is also in-house offshoring, it enables communication in all levels among different sites from code to human resources. In a individual view, developers can possibly share the company's resources in development such as reusable components and libraries. In an organisational view, two-stage offshoring models allows the company to gather talents at all sites along with development, and reduce cost by offshoring further to even lower destinations.

The bridging site in this model is quite delicate in the model. According to the case study by Olsson et al., the bridging site is mainly regarded as managerial outpost that maintains the communication and coordination between other sites.

3.2. Strategic Reuse Discipline

The related research in software reuse area, has provided some outstanding software reuse models. Ambler's Strategic Reuse Discipline is one of them which expressed software reuse in theoretical way. In next step, Strategic Reuse Discipline would be integrated with case 1 in real industry software development.

Reuse isn't free; it isn't something that happens simply because you're using certain tools or working with certain technologies. Instead, reuse is something that you have to work at very hard, as you can see in the high-level workflow for the Strategic Reuse

discipline in Figure 3 The detailed amalgamated workflow is depicted in Figure 3 (Ambler 2004).

Our perspective: In the field of offshoring environment, code is able to be reused across multiple applications as well as diverse organizations. For instance the department of the company in another place would reduces the amount of actual source code which they need to produce, potentially decreasing both development and maintenance costs. Simultaneously, the disadvantages are its scope of the effect is limited to programming as well as it often increases the coupling within and application. The point here in the thesis is trying to figure out software reuse which could potentially be part of the whole software offshoring process.

Consistent reuse requires a change in mindset. Developers must be willing to work together, to reuse each other's work, to help the reuse efforts of their organizations and to plan to reuse items whenever possible (Ambler 2000). When the offshore subsidiary starts a project, it should firstly consider what parts of the application could be reused from elsewhere. Perhaps another department of the company or the other organization has built what this project need. The flip side of the coin is that the offshore subsidiary must be willing to share its own work with other organizations which are outside the boundary of its own. In this case, there is one role called reuse registrar in the Strategic Reuse discipline who takes the responsible for publishing assets as well as announcement of the reuse work. This situation is sort of familiar with open source project environment. Reuse eventually will be an attitude, not a technology.

From organisation's perspective, there are many types of software development in association with offshore destination, and it is more possible to apply Strategic Reuse Discipline in the cases of in-house offshoring than offshore outsourcing. For outsourcing or offshore outsourcing models, there is an organisational barrier that significantly prevents the process of software reuse. The model of in-house offshoring provides a direct path between different sites of the company for software reuse activities. The model of Two-stage offshoring is potentially adaptable comparing to the model of Strategic Reuse Discipline. The bridge site in the two-stage offshoring model is mostly considered as an management and communication enhancement during cross-site development. Tasks are divided and distributed to different sites, while the bridging site coordinates the operations at different places. The SRD model illustrates a detailed managerial approach towards systematic reuse inside a single organisation. According to SRD, the company should assign a team to be fully in charge of the all reuse-related activities from planing, implementation and maintenance. In order to accomplish this settlement of this reuse team, the size of the organisation should have reached a reasonably large scale. Similarly, offshoring is also practices in larger companies. The isolation of distance can dramatically enlarge the difficulty of management and coordination between the associates at sites, and

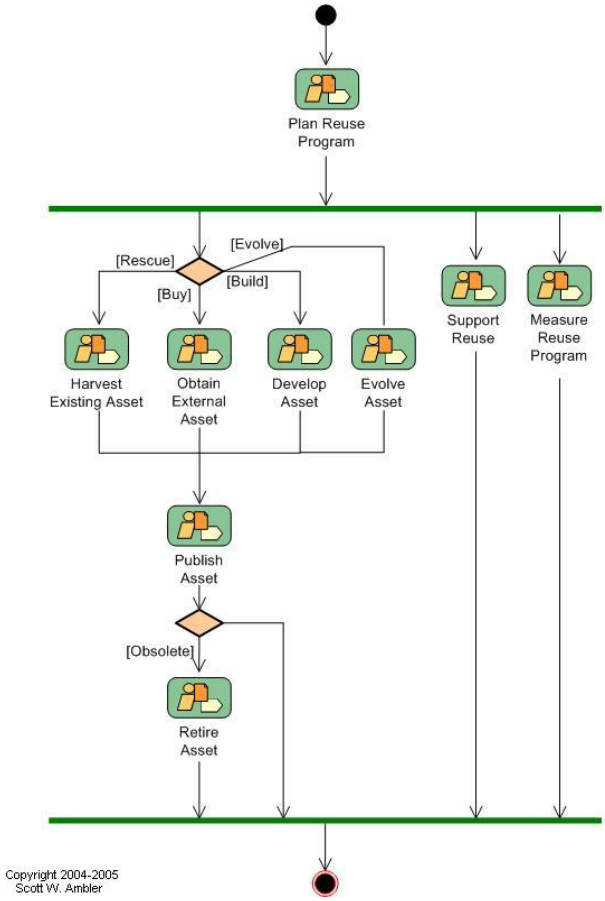


Figure 4: The workflow of the Strategic Reuse Discipline (Ambler 2004)

it would lead to vital disorder in smaller companies that lack of financial and managerial assets.

Strategic Reuse Discipline also reflects the labor arbitrary in the organisation. As the offshoring process getting matured, labor or jobs are delicately categorized according to the level of skills. in distributed development such as outsourcing and offshoring, it requires increasingly more effort to integrate the distributed work from different sites when the size of the project rises (Prikladnicki et al. 2007). Strategic Reuse Discipline could be a possible solution to simplify the complicity. With a reuse team at bridging site in charge of the planing,support and management of reuse activities, the organisation could redefine the tasks at different sites. the offshore sites(except the bridging site) can take responsibility of development and quality assurance of reusable assets while the onshore site can focus on defining the reuse standards and criteria.

4. Methodology

The proof of concept study as well as the reasons behind the decision to chose it will be introduced in this section as a research method for the paper. Moreover different data collected are identified by the specific approach for analyzing data.

4.1.Method Of Choice

The focus of the research is studying the benefits, difficulty and feasibility of applying software reuse and offshoring simultaneously in software development. Literature review is chosen for the research method. This paper aims to find out the outcomes while these two aspects cooperating with each other. Research data is collected from articles, papers and published case studies.

The research is consisted from following phases:

-Searching and finding related literature

-Choosing and categorizing related literature

-Few more relevant literature sources

-Analyzing the related literature.

Related literature for the research is consisted of papers, articles, books, conferences and publications on related content. The searching phase mainly based on online resources which including scientific search engines, library catalogs or database. Two main areas are constructed for the search phase: software offshoring and software reuse.

The goal of Strategic reuse discipline is to define how organizations can succeed at reuse. In order to have deep understanding of the discipline, one case study is essentially needed to ameliorate the data sources. Case study is a research methodology common in social science. It is base on an in-depth investigation of a single individual, group, or event to explore causation in order to find underlying principles (Baxter & Jack 2008 ; Dul & Hak 2008). For strategic reuse discipline, a case study based on software development industry would be an excellent opportunity to obtain significant insight into the model as well as enable the researcher to gather data from a variety of sources which converging the data to express the model. The case is to chosen to discover the actual software reuse model in individual perspective and verify the feasibility of SRD in practice.

4.2. Data Source

The primary data source for the research was the literature materials consist of articles, books and online publications. For example, Table 5 shows the literature that reviewd in the related research of IBM. And secondly, interviews materials conducted with one company Sandklef GNU Labs. There are two interviews has been accomplished. The interview questions are based on the related research outcomes which are able to enhance the understanding of these two subjects.

Articles	Type
Infrastructure Services, IBM Global Briefing	brochure
The future of IT application development	brochure
IBM Global Services	website
IBM Rational	website
IDC in Brno	website
Strategic Reuse with Asset-Based Development	book
Apply asset-based development to services in an SOA	website Article
Standards and reuse (Fay, 2004)	published Article

Table 5 Reviewed IBM Articles

Interviews are planed to be divided into elementary interview and feedback interview. The data would be recorded in a wide variety of ways including audio recording and written notes. The purpose of the interview is prove the ideas of the interviewees about the phenomenon of software reuse and offshoring in software development field. Ultimately, case study in this thesis is a method which enhance the understanding of the strategic reuse discipline. Therefore, it is difficult to conduct a perfect case study by various limitations of time and personal experiences. Interview content will be expressed in the appendix.Different sources of data collected in during this research is summarized with their advantages and limitations in the Table below.

Source	Data Collection Type	Advantages	Limitations
Sandklef GNU Labs software reuse specification	Document, architecture specification	Real industrial developers' experience	Subjective opinions may affect objective reality
The Leader of Sandklef GNU Labs	Interviews	Real software industrial perspective	Personal thoughts may affect objective reality
Literature from IBM	Document, reports	Valuable information based on domain knowledge	Difficult to address all relevant sections
Recorded media of interviews	Audio-visual material	Reprocessable	Interpreted issue.

Table 6 Summarized Data Sources

Case 1 Sandklef GNU Labs

Sandklef GNU Labs are written several tools and documents and been actively involved in the free software community sine 1998.

Interviews

2010.5.05 Interview with the leader of the company.

Strategic reuse discipline implementation phase.

2010.5.17 Feedback interview with the leader of the company.

Several softwares developed by ***Sandklef GNU Labs*** have been reviewed which helps the reviewer understanding the software development process better. This also helps the reviewer to get more feedback data when the software reuse discipline has been implemented.

GNU Xnee is a suite of programs that would be able to record, replay and distribute user actions under the X11 environment. It acts as a robot that can imitate the job user just did. GNU Xnee is licensed under GNU GPLv3 (http://www.gnu.org/licenses/gpl.html)

Swinput is able to fake a mouse and a keyboard by using the Linux Input System. The swinput modules read from a device and fakes hardware event (mouse motion, key presses etc) as commands written on the devices. Swinput presents status etc on the proc filesystem. It was developed to use when testing Xnee (http://www.gnu.org/software/xnee).

The data sources have been collectted from these two different companies are not simply for comparing the companies themselves. It mainly because the Strategic Reuse Discipline need to be implemented in at least two different types o f companies which in order to gather efficient feedback from the reality. The data sources are collected simultaneously, which means there existing comparision but the goal is mainly about testing Strategic Reuse Discipline.

4.3. Data Analysis

4.3.1. A Study Of Sandklef GNU Labs

Sandklef GNU Labs are written several tools and documents and been actively involved in the free software community sine 1998. Sandklef GNU Labs is a one man company forming teams with other companies that suits the needs of the projects.

Henrik Sandklef is currently working as a teacher at IT University and as a consultant. At IT University Henrik is giving courses about embedded programming and free software. (http://www.sandklef.com/sgl/)

For Sandklef GNU Labs, framework reuse is most implemented software reuse type . The choice of different software reuse type also depends on the software will be build, there is no specific reuse type would be chose all the time. Different requirement of the software will lead the development team to do the right decision.

Different perspectives from henrik expressed that advantages and disadvantages of software reuse, for instance, cost saving really depend on how much code the software development team needs, how much functionality the team needs. In some cases, getting a feature fixed by import or reuse a library probably will gain money in a short term, but it also make the development team tight up by the library too. In this situation, software reuse will probably rise up more complex maintenance in later phase.

For knowledge exchange, it definitely helps the organization to exchange knowledge as well as software development process. It also depend on what the company need. Some of the companies just reuse the software and keep the improvement as private software development, this kinda of reuse would probably not do anything about the knowledge. Hidden costs of software reusing, for instance, reuse one API, the developer need to think about that is it worth be tight up by the API.

When a development team set up a project. In this case study, the programmers would like to go through the old programs or code they have wrote. But still, the requirements of the new project have the higher priority. Set up the architecture of the project still will help the late work. For instance find out old software or code which are helpful.

The Ambler's model is way too complex for small software development companies. The role setting is better for big project. The roles of the Ambler's code reuse model are always shared among normal developers. The software reuse plan manager, in some cases, is the project manager who is in charge the software development process.

Actually, task of those roles in the Ambler's model are usually done by experience programmers. And they do not just hack it, they will start to analysis the code or the software they gonna reuse. These kinda persons are not like three years software development experience, they are really expert into this field who are able to handle the decision of software reuse.

Ultimately, the measurement of software reuse is really hard to be clarified in the beginning. And it do depend on how big project it is. Actually it is hard to see how much time or budget a development team have save during the project developing. it is also quite impossible to set up the goal and timetable about the reuse work in the beginning of the project. Normally, the manager would possibly have a overview about the benefits of the code reuse or software reuse at the end of the project.

4.3.2. Findings From The Case(Without SRD)

Unfortunately, software reuse does not just happen (*Herndon 1995*).

Due to the interviews, data sources shows that it is highly managerial in nature and they will not be successfully utilized without management and process support. Reusable assets needed to be designed and built in a well defined with understandable documentation.

Software developers need to keep an eye towards future use. Normally, software projects development existing in out sourcing field are built as “one-time only”, without reuse in mind. Single project typically is tend to be tightly bound within itself, without open interfaces which ease the reuse process. The software development process consequently must evolve to include reuse activities.

With a reuse process in place, every new system can be built from a set of core assets rather than rebuilding a system from scratch for each new customer's requirements.(Baragry 1994) This approach consequently adds new challenges for the software development team. For instance, instituting a training program for reuse strategies in management, design, implementation and test phases of development process (Baragry 1994). In order to meet these new challenges brought out by code reusing, a software development organization must possess some key abilities and have a strong commitment to goals of reuse (Brownsword & Clements 1996).

Firstly, members of the software development required training to perform their technical assignments associated with software reuse. As Henrik mentioned in the interview, experience software developers have the advantage when they are facing the software reuse. Beside that, a development team that is responsible for the maintenance of the reuse infrastructure must exist. Ultimately, responsibility must be assigned on each project of outsourcing or Offshoring companies for the registration and maintenance of reusable components. In essence, the organization itself must have a strong software development process foundation before attempting to incorporate reuse into the software life-cycle.

Along with many of the current familiar techniques, such as layered architectures, abstract interfaces design. The biggest new technical challenge on a product line approach is the initial design of the software architecture for the robustness towards potential future expansions, and its subsequent maintenance to deal with technology changes. This is also means the design of the software architecture should be carried out by people with experience and a solid understanding of software reuse development (Govardhan & Premchand 2005). Three main stages has been inferred from the case.

Planning the reuse program The outsourcing organizations must actively plan as well as budget for reuse. From the offshore subsidiary perspective, the time and resources are necessary needed to allocate to make reuse success. There is a bottom line that it is difficult for individual programmer to operate a reuse program within a organization. Most organizations fail because they make their own programs too complicated which provides no chance for the others to reuse.

Measuring the reuse programs When the organization set up the reuse programs in place, it is necessary to define the goals for it. For instance improving quality, reducing cost, and reducing time to market. The grandiose goals are easy to set up, but it is difficult to prove that the organization or different organizations have achieved them in practice.

The architecture of the code reuse process The architecture of code reuse, basically is divided in three main modules. Firstly, Code Incorporation center is in charge of uploading source code files which take responsible for transforming them into documents. (Relates to the code reuse registrar) Besides that, documents retrieval database is the foundation of executing queries over the connection between code incorporation center and code retrieval module. Code retrieval module is connecting the processes of the results obtained by documents retrieval database and use. It is able get the source code related to each document as well as provide it to the end code reuse participants.

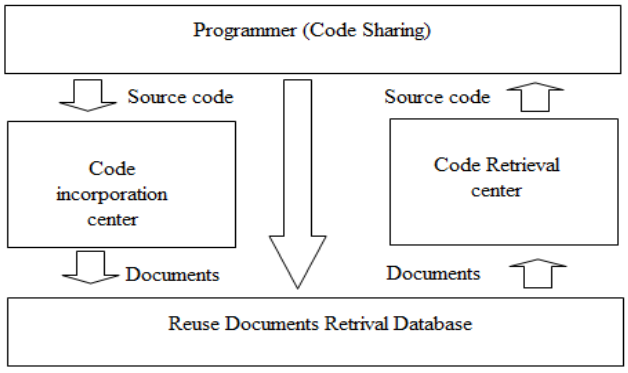


Figure 5, Basic Code Reuse Architecture (Adapted from Improving source code reuse through documentation standardization Basic architecture)

4.3.3. Finding Of The Case(With SRD)

After integrated the Strategic Reuse Discipline (SRD) with Sandklef GNU Labs case, findings are that there exists significant differences between the theory and software reuse process in reality software development process.

Firstly, from an individual software developer's perspective, the workflow of the Strategic Reuse Discipline is significantly helpful. For instance, from the plan of the reuse till the support as well as the measurement of the reuse are the essential steps for developers. And these processes usually would be neglected during the software reuse development by small group of developers.

From the small size software development company's perspective, specific process as well as reuse participants established in the strategic reuse discipline normally could not be completely implemented. In the reality software development field, especially for small companies the strategic reuse discipline, somehow, is too complex to be implemented. On the other hand, when software offshoring deal with software reuse during the development process, a reuse discipline would possibly enhance the development in some cases. Beside that, implementing the Strategic Reuse discipline brings up changes for the company. Change becomes to be an inherent section of the software development especially for software offshoring. As change occurs, the development team must have the most effective reaction for managing change with the software reuse.

Sandklef GNU Labs case also shows that the strategic reuse discipline potentially is serviceable for small size of software development company. Especially when the company is aiming sustainable development.

4.3.4. Findings Of Related Research

The related related research is based on literatrue review and has accumulated broad information on software reuse and offshoring. For software reuse, the major find is the challenges in applying SRD in software development. By reviewing IBM's study on this subject, it seems that the challenges have overthrown the benefits of the adaption. Asset-Base Development would not be fully functional even after all the changes successfully implemented. It takes time for the company to convert their already-made products into reusable assets or building more assets. Meanwhile, the cost of the change will results in all perspectives individually and organisationally. There will also be a rise of managerial challenges. The Asset-Base Development requires management focus on support, versioning and configuration of assets. In a view, Asset-Base Development reflects the division of labor in manufacturing different asset based on expertise. And this feature can be integrated with labor arbitrage in offshroing.

For offshoring, the major finding is the benefits and difficulties of Two-stage Offshoring. In brief, the benefit of Two-stage offshoring is continuously cost-reduction without involvement of outsourcing. The difficulty of applying Two-stage offshoring is the emerging challenges in controls of communication and coordination. The companies that have involved with offshoring are generally large firms with terms of employees on different sites. Two-stage offshoring is actually an evolved version of offshoring. IBM has even pushed this step to Multi-stage offshoring. From North America to Eastern Europe then to India, Philippines and China, all sites of the company can potentially function as either a development site of a bridging site.

5. Discussion

Introducing Strategic Reuse Discipline (SRD) to Sandklef GNU Labs case, it reveals significant conflict between the theory and reality. The theory defines several dedicated postitions (roles) for specific tasks during development. While the actual reuse process in the case, however, does not necessarily requires such amount of effort on controlling reuse, since most of the project are comparably small. For small companies, the strategic reuse discipline is beyond the capability of the firm and it is way too complex to implement. The most optimistic way of applying SRD in small firms is the adaption of the workflow. The Workflow of SRD describes the phases and activities in the reuse process. These phases and activities can be adapted individually, for example when one developer starts a new project, he or she can start with harvesting assets from various sources instead of building the wheel again. For larger firms , systematic reuse can be approachable but it takes enormous cost to make the change. The change also brings more difficulties which are mainly associated with management and planning of reuse. As for Strategic Reuse Discipline and the Asset-Based Development, this method has made the software development even more complicated. In order to take Asset-Base Development into practice, there will be a lot of changes. The development will In this method, There are many success factors in Asset-Based Development, and most of the them are associated with planing and management, such as identifying the asset to be built, management of support and versioning and configuration and so on.

It is obvious that offshoring is mostly likely to carry out in larger companies. Two-stage offshoring is a particular type of in-house offshoring. The bridging site in Two-stage offshoring is usually regarded as a communication hub between different sites, and managing the shifting of the task from one site to the other. The main difficulty in Two-stage Offshoring is how to ensure the communication, coordination, integration and conflict resolution (Olsson et al. 2008) in the development process. The Two-stage Offshoring has fully illustrated the theory of labor arbitrage or labor division inside the company. In addition Two-stage offshoring shares the benefits of distance proximity. Two-stage offshoring can be extended to nearshore locations with lower cost, on the other hand the overall structure of in-house offshoring would not break, and it can lessen the difficulties in communication and coordination in terms of operational processes. The company could assign different tasks on different sites based on the level of skills. In some cases, the software development process are distributed at different sites, for example one site focusing on coding and one site for quality assurance. In order to make the overall development successfully, the bridging site is vital due to its managerial function. And successful management at the bridging site becomes one major challenge in Two-stage Offshoring model.

Back to the research question of thesis: What is the outcome when applying software reuse and offshoring simultaneously in software development? It is quite critical to answer. For software reuse, the way that how company treat this subject is differentiated by the size of the company. Nevertheless, it requires great amount of cost to adapt systematic software reuse, for instance SRD. Offshoring is almost a "patent" for large companies. It takes investment to set up offshore subsidiaries while it needs more resources on managing these sites. Therefore, when using both approach in software development, it automatically increase the difficulties. The difficulties come from three directions: great amount of investment, tremendous difficulty of change and incremental managerial difficulties. If these worked out, the benefits are the union of both approaches. For feasibility, it is possible. IBM is a grand example, and surely IBM cannot stand for all companies.

6. Conclusion

It appears possible to let software reuse and offshoring co-exist in software development. But it depends on the capability of software development companies.

First benefits, both approaches has numerous features enhancing software development. Therefore, the combination of the two will unify the different advantages in both approaches, and extend cost-reduction and fast-delivery to a higher degree.

The for the difficulties, the adaption of either approach will result in increased managerial challenges. Applying both approaches simultaneously would probably multiplied the challenges of management in software development. Besides, the change itself cost great amount of investment and efforts.

Therefore, this article is also useful for those small companies who want to pursue a preliminary understanding about the implementation of a software reuse model. However, before suggesting how software reuse could be integrated with offshoring software development, it is also essential to look at the goal of the software development company. Small or medium size of organisation would prefer less investment with more returns.

Findings from the Sandklef GNU Labs case show that the strategic reuse discipline is still serviceable for small size of software development company. Based on the factors of feasibility, in the long run, strategic reuse also have chances to enhance the software development with the help of certain offshoring. As the successful example from IBM, there existing possibility of integrate strategic reuse discipline with offshoring. However, IBM case could not represent all kind of software develop companies. The feasibility of integrating them depends on how those companies deal with the difficulties expressed above.

All in all, implementation of software reuse and offshoring in software industry is not just another process improvement story of software development. However, like most good development methods, it requires a lot of planning, discipline, and up front investment. It is hard to move fast when people are running on a rough road. In the case of software reuse as well as offshoring, high quality of management in reality provides the smooth pavement for a faster ride.

7. Future Work

Future research are needed for several research aspects in this thesis. First of all, the goal of this paper is trying to find out the outcomes of applying two different software development methods. Therefore, more case study as well as literature review are needed to clarify the answer for the research problem. Secondly, the solution of by this research provides a preliminary theoretical guidelines. However, more empirical data analysis need to be explored to proof the solution. It is necessary to implement the Strategic Reuse Discipline in the real industry with all different size of companies. And observe how these companies could benefit from the model. Further more, how Strategic Reuse Discipline integrate with software Offshoring would be another interesting and advance research.

Reference

Abbott P 2007, *What do we know about distance in offshore outsourcing?* First Information Systems Workshop on Global Sourcing: Services, Knowledge and InnovationVal d'Isère, France 13-15 March 2007, Management Information SystemsUniversity College Dublin.

Ackerman, L, Elder, P, Busch, C, Lopz-Mancisidor, A, Kimura, J, Balaji,R 2008, Strategic Reuse with Asset-Based Development, viewed April 25th 2010, <http://www.redbooks.ibm.com/redbooks/pdfs/sg247529.pdf >

ACM Job Migration Task Force (JMTF), 2006, Globalization and Offshoring of Software

Ambler, SW 2000. Reuse Patterns and Anti patterns. Viewed March 24th 2010,<http://www.drdobbs.com/184414576 >

Ambler, SW 2005. Types of Reuse In Information Technology. Viewed March 24th 2010,<http://www.ambysoft.com/essays/typesOfReuse.html>

Ambler, SW 2004.The Strategic Reuse Discipline: Scaling Agile Software Development, viewed Feb 23rd, 2010 <http://www.enterpriseunifiedprocess.com/essays/strategicReuse.html>

Baxter, P and Jack, S. (2008) Qualitative Case Study Methodology: Study design and implementation for novice researchers, in The Qualitative Report, 13(4): 544-559

BelHard Outsourcing, 2010, Software Project Outsourcing Model, Viewed March 20th 2010, <http://soft.belhard.com/partnerships_1.html >

Black Duck Software 2009 Software Engineering Institute, Carnegie Mellon. Lines-of-Code Cost Calculator, viewed March 2nd 2010, <http://www.dwheeler.com/oss_fs_why.html >

Brownsword, Lisa & Paul Clements. October, 1996. A Case Study in Successful Product Line Development, Software Engineering Institute Technical Report, CMU/SEI-96-TR-016

Balaji RS, 2007, Apply asset-based development to services in an SOA, Viewed April 12rd, 2010 <http://www.ibm.com/developerworks/webservices/library/ws-soa-asset1/>

Baragry, Jason.1994. Summary of the ICSE 16 Panel on Software Reuse, Sorrento, Italy.

CALD 2010, Cambridge Advanced Learner's Dictionary, 2010, Viewed March 10th, <http://dictionary.cambridge.org/dictionary/british/>

Dul, J. and Hak, T (2008). Case Study Methodology in Business Research. Oxford: Butterworth-Heinemann.

Duke Center for International Business Education and Research (CIBER) 2006, Next Generation offshoring: The Globalization of innovation, Duke Univeristy, the Fuqua School of Business.

Govardhan, A & Premchand, P, 2005, A Pragmatic Apporach to Software Reuse, Journal of Theoretical and Applied Information Technology, viewd May 10th 2010, <http://www.jatit.org/volumes/research-papers/Vol14No2/3Vol14No2.pdf >

Herndon, VA, 1995. Software Productivity Consortium Services Corporation. Reuse-Driven Software Process Guidebook Product Description, SPC-93146-N,version 01.00.04

IBM, 2010, IBM Global Services <http://www-935.ibm.com/services/us/index.wss>IBM Rational Software <http://www-01.ibm.com/software/rational>, viewed 10th May 2010

IBM Global Briefing, 2010, Infrastructure Servi

ces, <http://www.ibm.com/investor/events/global0606/summaries/Infrastructure_Services.pdf >, viewed 18th June 2010.

IDC in Brno, 2010, viewed May 4th 2010<http://www-05.ibm.com/employment/cz/idc_brno/index.html>.

Lee,J, Huynh, MQ, Kwok,RC, Pi,S 2003, IT outsourcing evolution: past, present, and future, Volume 46 , Issue 5 (May 2003) Wireless networking security Pages: 84 – 89 Year of Publication: 2003 ISSN:0001-0782

Olsson, H, Conchuir, E, Agerfalk, P, Fitzgerald, B 2008, Two-stage Offshoring: An Investigation of The Irish Bridge, MIS Quarterly Vol. 32 No. 2, pp. June 2008.

Palmisano S, 2006, The Globally Integrated Enterprise, viewed 10th May2010, <http://www.ibm.com/ibm/governmentalprograms/samforeignaffairs.pdf>

Panel of the NATIONAL ACADEMY OFPUBLIC ADMINISTRATION(PNAOA),2006, Off-shoring: An Elusive Phenomenon, National Academy of Public Administration

Prikladnicki R, et al. 2007, Distributed Software Development: Practices and Challenges in different business strategies of offshoring and onshoring, international Conference on Global Software Engineering 2007.

Roberts, PC 2004, *"Global Labor Arbitrage" Dismantling America*, July 28,viewed March 10th 2010, <http://vdare.com/roberts/labor_arbitrage.htm>

Ring, P.S. & Van de Ven, A.H., Developmental processes of cooperative interorganizational relationships. Acad. Mgmnt Rev., 1994, 19, 90–118.

Sommerville, I 2006, Software Engineering 8th Edition, Chapter 18 Software reuse, Addison Wesley.

United States Patent & Trademark Office (USP&TO), 2007, United States Patent Application 20070162321,viewed May 1st 2010. <http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1&u=/netahtml/PTO/srchnum.html&r=1&f=G&l=50&s1="20070162321".PGNR.&OS=DN/20070162321&RS=DN/20070162321>

Zhu, Z., Hsu, K. and Lillie, J., Outsourcing – a strategic move: the process and the ingredients for success. Mgmnt Decision, 2001, 39, 373–378.

APPENDIX I

Table: Data Analysis Of Two-stage Offshoring And Strategic Reuse Discipline

			Two-stage/Multi-stage Offshoring	Strategic Reuse Discipline	Two-stage/Multi-stage Offshoring & Strategic Reuse Discipline
SoftwareDevelopment	SmallCompanuy	Sandklef GNULabs	N/A*	B: potentically increase Effitiency	N/A*
				D: Change Management	
				F: Prtially, Individueal Level, the SRD workflow could be applied individually	
	LargeCompany	Pennysoft*	B: Decrease Cost, Increase Productivity	N/A*	N/A*
			D: Communication, Coordination, Conclit Resolution, Intergration		
			F: Already done		
	GloalIntergratdCompany	IBM	B: Decrease Cost, Increase Productivity, talents, ect	B: Fast Delivery, Decrease Cost, Increase Productivity, ect	B: Fast Delivery, Decrease Cost, Increase Productivity, ect
			D: Change Management in organisational perspective	D: Management, Training, Measuring productivity, Mantenance, ect	D: Management, Training, Measuring productivity, Mantenance, ect
			F: Already done	F: Already done	F: Already done

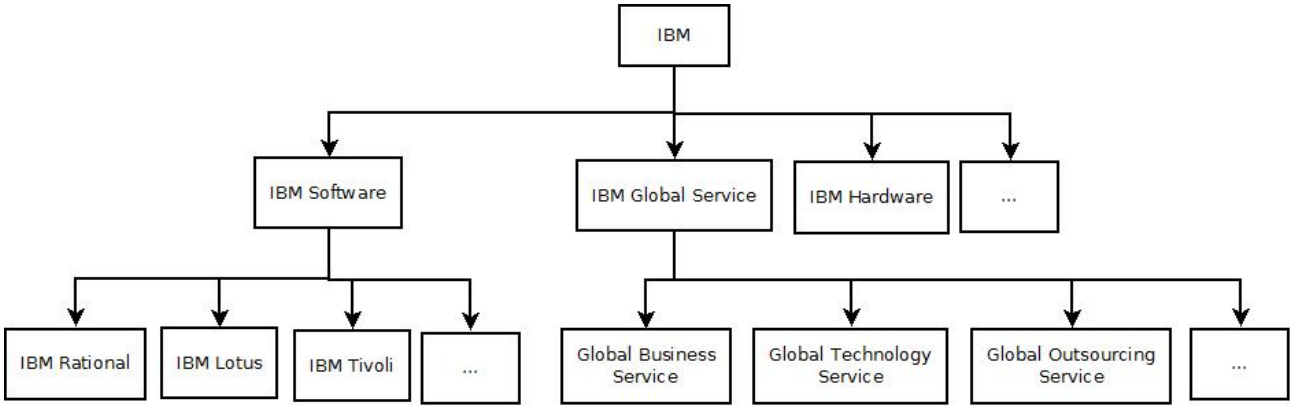
Table : Data Analysis of Two-stage Offshoring and Strategic Reuse Discipline
B: Benefits, D: Dificulties, F: Feasibility

**** Pennysoft is from the case study by Olsson et al. In Two-stage Offshoring: An Investigation of The Irish Bridge***

N/A: some of the data is not available.

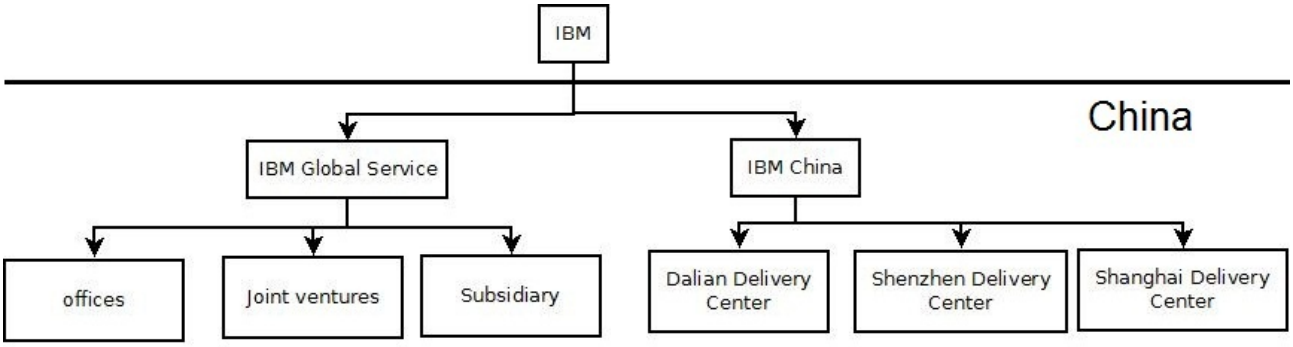
APPENDIX II

IBM Offshoring Structure



Appendix_Figure 1: IBM Structure

IBM In China



Appendix_Figure 2: IBM in China

APPENDIX III

Reuse Success Factors In Asset-Based Development (Ackerman Et Al. 2008)

Management support: As with most initiatives, it is important to have executive support.In the case of reuse, this is critical. However, we also find that the development ofreusable assets is most often bottom-up. Therefore, it is important to have top-downsupport while implementing assets bottom-up.

Identifying the right asset to build: Next to management support, this is the next mos timportant factor to consider. In this case, we need to decide what assets must be built. A strategy that only looks at measuring success based on the number of assets in the Repository is unlikely to succeed. The correct approach is to measure success based on the value that assets bring to your organization. When selecting what assets to build, look for recurring problems and best practices that need to be captured and ensure that there is indeed a need and a consumer for the solution. An asset that does not get used provides little value to the organization.

Versioning and configuration management: Typically an asset is composed of multiple artifacts. As we build, maintain, and then support assets, it is expected that there will need to be new versions created. We end up seeing that there are two types of versioning that are needed - one to keep track of the versions of the artifacts, and then as we build and create the asset that contains the artifact, we will specify versions for the asset.

Training for producers and consumers: For reuse to be successful, those people producing assets must have skills in producing assets that are reusable. The Asset Consumers need to understand how to find assets, use them in the environments, and if appropriate, customize the assets for a particular situation.

Measuring productivity and quality: Understanding asset usage, asset quality, and the cost and impact of assets is key to the program.

Sponsorship and maintenance of assets: Much like a software application, the cost to maintain and support an asset does not end at the first release. There needs to be a plan and support for the continued maintenance of the asset.

Communication of asset status to Asset Consumers: The Asset Producer needs to understand where and how the asset is being used. This will assist the producer in prioritizing the repair of defects and how to introduce and manage changes to the asset.

Commitment to high quality assets: Without a focus on high quality assets, a Repository will become a junkyard, a dumping ground for any old thing that might be reusable. This environment is unlikely to find success as the consumers of assets quickly learn that it is quicker, easier, and less risky to proceed on their own.

Well-understood domains: Assets are meant to contain and represent best practice solutions. To be able to build and support such assets, the Asset Producer needs to understand the domain for the asset.

Customizable, coarse-grained reuse: Successful assets tend to have built-in support for customization. In cases where there is unlimited access to changing and customizing the asset, the consumer will find the asset's flexibility overwhelming and lacking in guidance.

Architectures established to create and use assets: Related to understanding thedomain, we also need to understand the architectures that are targeted by the assets being built.

Process and organizational structure to support reuse: In order for reuse to succeedwithin the organization, there generally must be a team that focuses on manufacturingassets. The challenge of this is that there are many technology domains for which reusable assets can be created, and it is difficult if not impossible to have one team that has all of the experts needed. Therefore, you generally have a core team that understands the principles of asset manufacturing, and you have visiting experts that join for a time to give guidance on the development of specific asset types. The enterprise needs to support this activity of harvesting expert knowledge into reusable assets, when and where it makes sense to the business.

APPENDIX IV

Interview Contents

Henrik, Sandklef GNU Labs

Q:What kind of softwar you work with, free or proprietary?
A:FOSS, comercial applications, never proiatory.

Q: Are you familiar with the Strategic Reuse Discipline model? which is part of Enterprise Unified Process.
A: Never heard of them.

Q:What company you have worked with?
A: Ericsson, Volvo trucks, Consta Engerring

Q: How big is the team you been work with normally?
A: Around 10-15 people.

Q: How many people were working on the project in that specific department?
A: Maximum 25 people.

Q: Do you think code reuse is important?
A:Yes!

Q: How do you reuse code?
A: - Harvesting existing assets (From the software/code you made before) from free software, own code,libruray of companies.
- Obtaining External Assets (From FOSS)

Q: **Is there a position called "reuse engineer" in a development team/organization based on your experience?**
A: Not really, there is not such title there in the company, but about one or two people will take care of reuse.

Q: Systematical software reuse does not happen spontaneously, have you ever made plans for reusing code?
A: When you have years of expirences, you can quickly find out a solution.

Q: Do you consider re-usability as an important software quality when you start a project (so you can reuse the code in the future)?
A: Yes, but most projects start smalland when they get bigger the code turn into a mess. But I will surely agree to make good structure to make reuse and mantainance easier.

Q: - You said structure, do you mean the structure of code or the structure of the company(organisation)?
A: - both of them.

Q: Do you evolve and support your assets(code) to meet new requirements? How do you evolve your robust assets over time? Do you often fix bugs and make new updates to your old stuff (or let someone else do that for you)?
A: Yes, I constently. But sometimes I make sacrifaices for the code in order to meet the requiremnets. But this sacrifices usually can be fixed later.

Q: Do you publish your assets so someone else can reuse them? Have you heard of something called " Reuse Registrar"? And does Reuse manager exsit in this world? Since the author of this model has defined a few specialists that carefully control the reuse process, and I am wondering if these roles really exists.
A: yes, I have published a lot of code. But most cases, they are small scale end-user application. People use them as the way they are.

Q: Have you ever thought about retiring an asset? For example, delete the old versions that you cannot support any more.
Quite a lot of times. So I do not need to support the code anymore.

Q:Have ever done that?
Well, some of the project just died out, since the number of user is zero.

Q: Have you ever measured Reuse? How much time/budget saved? (maybe not the case for engineers)
A: No, but Interesting question. It is very difficult to measure in this case, but I would like to know if

Q: How much percentage of code reuse u have done in your whole programming life?
A:The code reuse is kinda depend on the requirement of the software u gonna develop. If u meam the whole life of programming, 60 to 70 percentage of code reuse i have done.

Q: When a development team set up a project. Would them start to go through old code or software, or just start to write the new program

instantly.

A: Personaly, i would like to go through the old programs or code i have wrote. But still, the requirements of the new project have the higher proiety. Set up the architecture of the project still will help the late work. For instance find out old software or code which are helpful.

Q: What is your personal idea about code reuse. R u insterested in it.

A: Ofc, it doest matter what the new project is, developers probably would have done quite a lot work on reusability, like code reuse, module reuse as well as architecture design etc

Q: Have you ever measure reuse? How much time or budget saved?

It is really hard to measure the reuse. And it do depend on how big project it is. Actually it is hard to see how much time or budget u have save during the project developing. it is also quite impossible to set up the goal and timetable about the reuse work in the beginning of the project. Normally, the manager would possibly have a overview about the benefits of the code reuse or software reuse at the end of the project.