Date of acceptance Grade

Instructor

# Software Engineering Challenges in Small Companies

Yiyun Shen

Helsinki 04.04.2008

Seminar report

UNIVERSITY OF HELSINKI

Department of Computer Science

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

| Tiedekunta/Osasto – Fakultet/Sektion – Faculty/Section | Laitos – Institution – Department | |
|---|---|---|
| Faculty of Science | Department of Computer Science | |
| Tekijä – Författare – Author | | |
| Yiyun Shen | | |
| Työn nimi – Arbetets titel – Title | | |
| Software Engineering Challenges in Small Companies | | |
| Oppiaine – Läroämne – Subject | | |
| Computer Science | | |
| Työn laji – Arbetets art – Level | Aika – Datum – Month and year | Sivumäärä – Sidoantal – Number of pages |
| Seminar report | April 04, 2008 | 12 |

Tiivistelmä – Referat – Abstract

This paper described three of the challenges met by small software companies and their corresponding solutions. It focuses on the challenges met by small software companies from software engineering's point of view, and introduces several approaches, which have been demonstrated helpful to solve the specified problems.

The paper is organized as follows. Chapter 1 gives an introduction of the current situations of the small software companies. In chapter 2, the major software process challenges of small organizations are discussed. In chapter 3, possible solutions to the challenges will be described in turn. Chapter 4 contains a brief summary and the prospects of small software companies.

Avainsanat – Nyckelord – Keywords

small software companies, software engineering challenges, software process improvement

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

# Contents

# 1 Introduction

Small companies, judged by number of employees, are generally under 100 employees in the United States while fewer than 50 employees in the European Union [Sma08]. Small companies, together with large companies, have coexisted for quite a long time, which makes it difficult to trace the origin of so-called small business. However, for the small software companies, the true spring can be dated back to 1970s, when the third industrial revolution happened. The rapid advance in technology, which opened the well-known "Information Age" since 1974, is undoubtedly linked to IT development, both for hardwares and softwares [Gre97]. Though we often concentrate on large companies and their famous and widely used products, we cannot ignore the importance of small companies in a national economic system. According to the software industry statistics taken in Ireland, small software companies play a fundamental role in many national economies' growth. They represent up to 85 percent of all software organizations in the US, Canada, China, India, Finland, Ireland, and many other countries [Sof06]. In Finland, the software industry is one of the key industrial segments. This industry is expected to employ more than 100 000 people in the year 2010 [Nuf99]. The number of small Finnish software companies doubles that of the large firms according to the questionnaire given by the Technical University of Helsinki in 2000, which pointed out that about 140 out of 200 software companies had fewer than 20 employees [Sal01]. Different from other counties, the software industry in Finland has concentrated on providing technological solutions to business-to-business niche markets since its 'rise' in the 1970s [Sal01].

Generally, small companies are supplementary to large firms from marketing view. Large companies, with enough financial and managerial resources to develop and market new technologies, aim to gain dominant share of a market. Contrarily, small software companies, in order to survive in crucial competition, mainly concentrate on a market niche, which is disregarded by large companies. Small companies cannot be simply seen as scale-down versions of large ones [Sto82]. For small companies, the significant advantages commonly adopted are their excellent responsibility and flexibility. They are selling innovative programmes with special features or offering particular solutions and services to their customers. While running their businesses, small software companies often meet difficulties in finances and staffing. A majority of small companies are independently financed and rather like a develop team than a company in size. Therefore, we cannot just apply the software engineering standards and solutions designed mainly for large companies to small ones. The smaller and less well known the company is, the less attraction it has to the experienced professionals.

The challenges in running small software companies seem not only to be networking, marketing, business issues, but also to creating and leveraging technological knowledge and know-how. Though having similar objective in providing high quality softwares and services to satisfy customers, small and large companies cannot both apply same development methodology or techniques without any modification and optimization. Actually, due to the limitation of resources and business issues, those best practices proved in large firms might be too expensive or time consuming to perform in small companies. Accordingly, the recent researches start to find special solutions to improve small companies' software processes in several aspects. In addition, the standardization organizations set up modified standards and improved approaches on SE Life-Cycle Profiles for Very Small Enterprises (VSEs refer to companies with fewer than 25 people) [Riw07].

This paper focuses on the challenges met by small software companies from software engineering's point of view, and introduces several approaches, which have been demonstrated helpful to solve the specified problems. The paper is structured as follows. In the next chapter, the major software process challenges of small organizations are discussed. In chapter 3, possible solutions to the challenges will be described in turn. Chapter 4 contains a brief summary and the prospects of small software companies.

# 2 Challenges

Before talking about the software engineering challenges in small companies, I would like to emphasize that the scale of the software companies is "small". Larry argues that it is the business goals, not the organization's size, that matter software development. Though given different characteristics of small and large companies, if they have the same business goal, then, they can apply the same software engineering techniques [Lum07]. Philosophically, that might be true under some particular conditions, because both of the small and large firms want to make benefits in marketing and to maintain good relationships with their customers. However, practically, the ways to achieve similar business goals can be greatly varied from large companies to small ones. This makes the whole story be different to small companies. The size of organization has influences on the choice of software development strategies and the use of SE technologies, because leaders have to take the constraints of staffing and budgets into consideration. According to this situation, three main software engineering challenges faced by small companies are described as follows.

First, using toolkits, the copyrights of which are reserved by other commercial entities, can be costly. To organize software process, project leaders need the help from lots of development and management tools. In large firms, they may have professional teams to develop and maintain their own development platforms and secondary toolkits, which can improve the whole work efficiency. On the contrary, this is not affordable to small software companies. Though small companies need tools to communicate with customers and to cover every step in software process, the high cost in purchasing new secondary toolkits will make the project easily over its original estimation.

Second, complex but helpful measurement can be time-consuming for small software companies. The benefits of measurement are usually found in analysing the effects on software process by improvement efforts. The large firms, of course, with enough resources, have published their practical experiences in apply metrics programmes in software process improvement. [Kau99] Even though, the managers and developers in small software companies are reluctant to measurement, partly because they doubt whether the metrics programmes can lead them to success after spending extra time on measurement.

Third, small software companies suffer from the lack of real-world publications from similar companies describing efforts on an improvement initiative [Kht00]. Adopting internationally accepted software process practices is essential for software companies

to compete in the global software development market [Gue04]. Although more and more studies on successful initiatives are available in recent years, they still seem to be restrictive.

# 3 Solutions

This chapter includes four sections of useful approaches in solving the corresponding problems described in chapter 2. Section 3.1 introduces several open source tools covering almost all steps in software process. Section 3.2 describes successfully applied measurement in small software companies. Since software process improvement is a demanding and complicated work, Section 3.3 will present an adjusted IDEAL model, which is practical for small software enterprises.

## 3.1 Open Source Tools Used in Software Process

As the large software organizations, small organizations aim to gain all the benefits through exploiting the software engineering practices. However, unlike large companies, small companies face with the lack of staff to develop functional specialties to perform complicate tasks supporting there products implementation. The software process designed needs to be lightweight, low cost in training and suitable for rapid development. Therefore, most of the tools used in the process should be open source. In Ken and Bill's open source approach applied to software development, particularly for long-term project, the software process is conducted into five parts, namely, communication and documentation, revision control, building management, testing, and release process [Mah07]. Fortunately, more and more open source tools covering almost all parts of software process are available these days.

During software development and maintenance, effective communication and sufficient documentation can lay a solid foundation in software process. The role of communication is also stressed with its influences on the perception of the innovation. In recent years, Wiki has become one of most popular communication tools in the world. The most famous application of Wiki technology is the Wikipedia. To introduce Wiki into software process, especially in communication aspect, has become a first place option for many project managers. Wiki, which is software that used more like a platform to allow users to create, edit, link, and organize the communication contents collaboratively, is a preferable place to put Q&As issued by either customers or developers, and to record new sparkling ideas freely [Wik08]. It helps to narrow the communication gap between developers and customers, even between developers themselves. As said, the contents of Q&As and important discussions can be taken into documentation if needed. Therefore, after a long time,

less important points will be missed or forgotten than ever before. The improved documentation helps developers to understand and manage their work, and meet the customers' need for accurate information.

Revision control is essential not only for large software companies but for small ones as well. No matter how small the project will be, the project team need the Revision Control System (RCS) to store, retrieve, log, identify and merge revisions automatically. With the RCSs, developers can easily track changes in codes and documents at any time and manage file versions. Concurrent Versions System (CVS) and Subversion are two open sources revision control tools widely used nowadays. Subversion is superior to CVS by providing http/https server mode and allowing users to move and rename directories or branch and tag files [Mah07].

The Trac system can be a good example of the open source tools used in software process. Trac combines information between wiki content, revision control, and a computer bug database by allowing wiki mark up in issue descriptions and commit messages, creating links and seamless references between bugs, tasks, changesets, files and wiki pages. Besides, it serves as a web interface to a revision control system, like Subversion, Git, Mercurial, and Bazaar. Another remarkable function of Trac is the timeline, which shows all project events in time order, making it more easily for developers to be acquaint of an overview of the project and tracking progress [Tra07].

## 3.2 Simplified Measurement

As mentioned before, the goal of measurement is to improve software process, and almost all software companies can benefit from the improved software process. Thus, theoretically, software companies no matter whether they are small or large should warmly welcome measurement and metrics programmes. Actually, the story is rather opposite to that ideal thought. The software developers, especially from small companies, tend to greet measurement activities with scepticism before they see true profits from metrics programmes [Kau99]. After the researchers had an insight into the doubts of software developers, they found the reasons as follows: the measurability of software work, the usefulness of data collection, the fears of being controlled by their employers, and the extra workload spent on measurement, which is the most considerable thing [Kau99]. For these reasons, the comprehensive, complicated metrics programs applied in large firms are meaningless for those small software companies.

In order to eliminate the doubts of software developers, new, simple, quantitative, small-scale metrics programs are required. Moreover, the metrics programmes need to be individualized for different companies with high diversities in their characteristics

and key objectives. Examples for varied key objectives of small software companies are to reduce dependency on the chief developer by having all team members' work on all parts of their product, to maximize the accepted change requests handled in a certain period, to minimize the time spent on handling customer change requests, etc [Kau99]. The next step is to gather data from all possible resources. Since the developers may have residual doubts about measurement benefits and increase in workload, it is better to aggregate figures from existing available data. The following step is to generate the results of measurement, which is an important step to make the whole evaluation be visible and fruitful to developers and customers. The key to successful measurement is to understand the objectives of small companies. It has been proved that technology transfer is a process happening in social environment rather than a context-free technical matter. Given different situations and characteristics, we should adjust metrics programmes correspondently. Though small in size, the software companies with fewer than 20 developers need some basic formal routines as well. Only if metrics programmes are carefully selected and the purposes of which are well dedicated, can we achieve the balance between mechanisms and documented procedures.

## 3.3 The IDEAL Model

The IDEAL Model is proposed by the Software Engineering Institute (SEI) to help the construct and implementation of software process improvement schemes [Kht00]. It is important to understand the meaning of the software process in order to comprehend the model. It is also necessary to understand that a software process model is not a mathematical formula. In this context, it is a description of how to conduct the process of software development. *Software process is defined as a set of activities that begin with the identification of a need and concludes with the retirement of a product that satisfies the need; or more completely, as a set of activities, methods, practices, and transformations that people use to develop and maintain software and its associated products (e.g., project plans, design documents, code, test cases, user manuals).*

### 3.3.1 The IDEAL Model

The IDEAL (Integrated Design, Evaluation, and Assessment of Loadings) Model has been developed in order to provide a routine of steps that constitute a software process improvement program. [Kht00]

The IDEAL Model includes five phases, as shown in Figure 1:

- the Initiating phase

- To build up the stimulus and infrastructure for improvement
- To initialize the roles and responsibilities of team members and allocate initial resources
- To define the business requirements based initiative
- To establish a management group and a software engineering process group
- To prepare an initial improvement plan for the next two phases

- the Diagnosing phase
  - To create a baseline of the current state of the company
  - To document an improvement action plan as a initial version which contains the results and recommendations from evaluation activities
- the Establishing phase
  - To prioritize the issues decided by the company
  - To form the paths along which to find solutions
  - To finish the action plan draft generated in the former phase
  - To develop measurable goals and metrics to control infrastructure
- the Acting phase
  - To create deploy solutions basing on the aspects of improvement found in the Diagnosing phase
- the Leveraging phase
  - To evaluate the data collected in the earlier phases, the lessons learned and metrics on performances

Then, the next pass through the IDEAL model begins with probable adjustments of the strategy, the methods and the infrastructure.

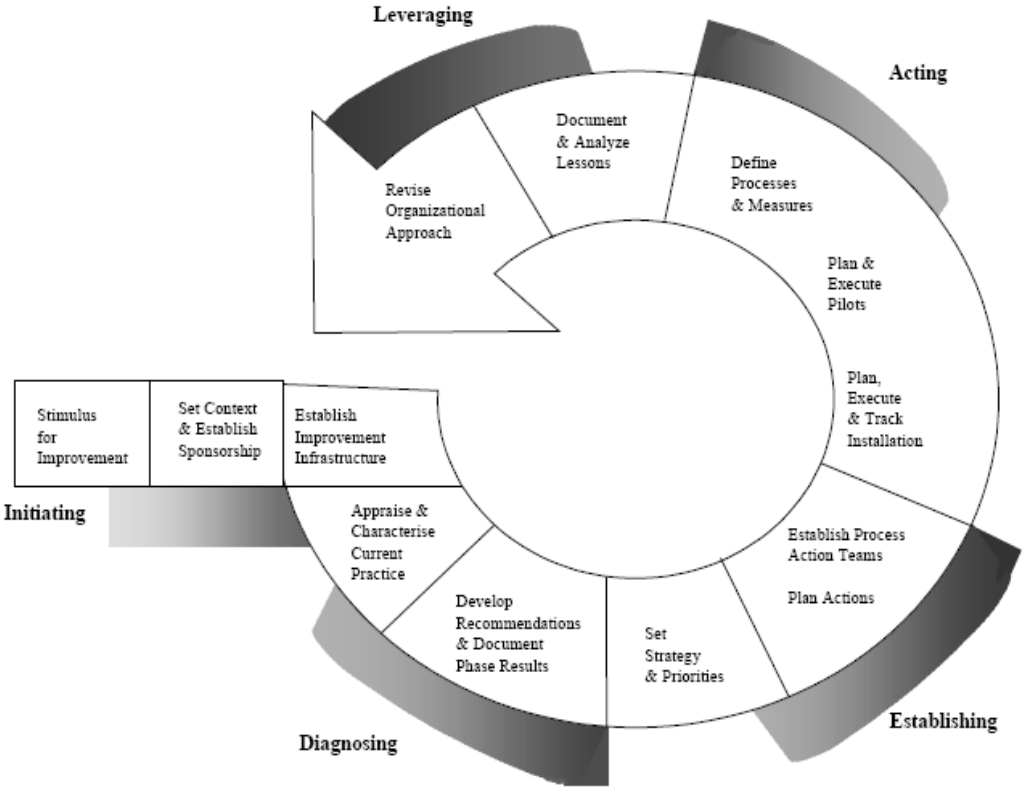


Figure 1. The IDEAL Model [Kht00]

Although the IDEAL model is described graphically as a step by step execution, the boundaries between phases are not that clear, which allows the model to be applied in parallel.

### 3.3.2 Adjustment of the IDEAL Model

It is highly recommended by the IDEAL model guide [Mcf96] to tailor the model according to the small software companies' characteristics, such as resources, visions and business objectives [Kht00].

In actual implementation of the improvement proposals in small companies, resources are more critical than imagined. Hence, we have to plan to control risks caused by staff shortage in advance. Some possible actions can be taken in resource assignment can be hiring specialists from outside the company, letting all developers participate into each part of the project, etc.

The cultural and organizational facts have great influences on shape the software companies. There are two kinds of issues faced in the acting phase, namely, problem-centred and process-centred. The problem-centred issues are easily identifiable, fast fixable and quickly effective, while the process-centred issues are related to key process and effect in long term [Kht00]. The IDEAL model tends to have an adjustment in defining issues between problem-centred and process-centred in the

acting phase as early as possible. Successively, adjustment is required in the establishing phase by using different strategies respectively. For instance, if in the acting phase, the process-centred issues are realized, the establishing phase, together with the acting phase will be synchronizing phases. On the contrary, if in the acting phase, the problem-centred issues are realized, the establishing phase is no longer needed to be executed. Furthermore, some unnecessary review tasks can be omitted between the initiating phase and the establishing phase because of the shortness in time between these two phases.

There are some other factors, which are not least important, rather than the adjustment of general models:

- Management support and commitment
- Project planning and organization
- Education and training
- Assessment
- Monitoring and evaluation
- Staff involvement
- Support and knowledge transfer by external consultants
- Usability and validity of the introduced changes and cultural feasibility

# 4 Summary

This paper described three of the challenges met by small software companies and their corresponding solutions. As mentioned in the previous sections, the challenges in small software companies tied with their business characteristics, like small-scale, limited budgets, lack of resources, especially the experienced staffs, etc. To some extent, the small companies enjoy the advantages in immediate decisions addressing, flat communication and flexibility in marketing. However, they also suffer from the pains in software products development, and more significant, in improve software process.

Fortunately, with the prosper of open source projects and products available online, the lack of funds, which are spent on purchasing process management tools, will no longer be the challenge in small software companies. However, the other software process improvement challenges in software engineering aspect still exist. Obviously, further study is needed on the role of software process improvement approaches, especially metrics, in enhancing development practice and product quality in small software companies. Moreover, we still need to continue further studies concentrating on how to apply suitable process model with necessary adjustment in real-life projects. As the small software companies begin to gain more and more attention all over the world, they are sure to find a suitable way for them to survive in the global market.

# References


Etw95 H. Etzkowitz and A. Webster, *Science as Intellectual Property*. Handbook of Science and Technology Studies, Society for Social Studies of Science.

Gre97 Jeremy Greenwood, *The Third Industrial Revolution: Technology, Productivity and Income Inequality*. AEI Studies on Understanding Inequality, Washington, DC. The AEI Press.

Gue04 F. Guerrero and Y. Eterovic, Adopting the SW-CMM in a Small IT Organization. IEEE Software 21, 4, 2004, 29-35.

Kht00 K. Kautz, H. W. Hansen, and K. Thaysen, *Applying and Adjusting a Software Process Improvement Model in Practice: the Use of the IDEAL Model in a Small Software Enterprise*. Proc. 22nd Int'l Conf. Software Eng., IEEE CS Press, 2000, 626-633.

Kau99 K. Kautz, Making Sense of Measurement for Small Organizations. IEEE Software 16, 2, 1999, 14-20.

Lum07 L. Lumsden, *Business Goals Count, Not Organization Size*. IEEE Software 24, 1, 2007, 54/56.

Mah07 Ken Martin and Bill Hoffman, *An Open Source Approach to Developing Software in a Small Organization*. IEEE Software 24, 1, 2007, 46-53.

Mcf96 B. McFeeley, *IDEAL℠: A User's Guide for Software Process Improvement*. Handbook CMU?SEI 96-HB-001. Software engineering Institute, Carnegie Mellon University, Pittsburgh, PE, USA.

Nuf99 Nukari, J and Forsell, M, *The Growth Strategy and Challenges of the Finnish Software Industry*. TEKES, teknologiakatsaus 67/99, Helsinki.

Riw07 I. Richardson and C. G. Wangenheim, *Why Are Small Software Organizations Different*. IEEE Software 24, 1, 2007, 18-22.

Sal01 Sari Sallinen, *Social Embeddedness of Dynamic Capabilities: The Case of Product Development of Small Finnish Software Supplier Companies*. University of Oulu, Finland, 2001.

Sma08 Small business http://en.wikipedia.org/wiki/Small_business [30.03.2008]

Sof06 *Software Industry Statistics for 1991-2005*. Enterprise Ireland, 2006. www.nsd.ie/htm/ssii/stat.htm

Sto82 D. J. Storey, *Entrepreneurship and the New Firm*, Croom Helm, 1982.

Tra07 Trac
http://trac.edgewall.org/
[17.08.2007]
Wik08 Wiki
http://en.wikipedia.org/wiki/Wiki
[30.03.2008]