A Study of Software Change Management Problem

S. M. Ghosh[1], H. R. Sharma[1], V. Mohabay[2]

[1]*Chhatrapati Shivaji Institute of Technology, Durg (CG) – INDIA*

[2]*Department of Electronics and IT, Kalyan Mahavidalaya Bhilai (CG) - INDIA*
samghosh06@rediff.com, hrsharma44@gmail.com, mohabay@gmail.com

Abstract

Software change management is an essential discipline for enterprise IT organizations. In modern enterprises, software automates a wide variety of business processes. Changes made to software are, in effect, changes made to the business processes themselves. While the majority of enterprises have implemented basic software change management processes, these processes fall down in the face of the increased complexity of today's business and IT environments. Truly effective and efficient software change management will require better coordination of change across geographic, organizational, functional, and technological arena of IT implemented organization. This paper presents a classification of the change management problems derived from the literature study done. This study explores software change management — the definition and implementation of defined processes to control changes made to software. It also addresses the specific problems in the change management of application software development.

Keywords*: Enterprise Resource Planning, change management , software Configuration Management*

1. Introduction

To assess the state of software change management practices in enterprise IT organizations, research activity is conducted using semi focused interviews and literature study was done among three ERP implemented manufacture industry with annual turnover of $300 million in annual revenues. This survey was conducted as the part of research activity for PhD programme. The respondents were decision-makers or influencers who are familiar with their IT organization's change management processes and who play the role of decision-maker or influencer when it comes to defining and implementing those processes. Sample respondent titles include director/VP of application development/systems development and director/VP of the project management office. The goal of the research is to identify the problems of change management processes and its categorization. The studies also incorporate the improvement requirements in change management processes.

2. Classification of Problems and Improvement Requirements in Change Management Processes

The literature study and the case studies revealed problem areas and improvement requirements in the change management practices. The following six main problem groups can be derived from the problems identified in the three case studies:

- Analysis and identify problems
- Communication problems
- Decision-making problems
- Effectiveness problems
- Traceability problems
- Tool-related problems

The following paragraphs explain each problem category in detail and give examples of the problems related to each problem class. The problems related to the special features of developing enterprise software are examined separately for each class.

The purpose of the problem classification is to provide a framework for analysing the problems related to change management [8]. The framework supports the change management process analysis task by providing a structure which can be used for identifying the change management problems in the process under analysis. The classification can be used in for analysing and structuring the problems found in change management, and later to evaluate how the new change management procedures introduced improved the change management processes.

Analysis and identification problems

One of the most time-consuming and error prone phases of the change process is problem understanding in the context of understanding the system and analyzing which parts of the system will be affected by the change [1]. This was also clearly stated by the software engineers in the organisations analysed in the case studies. Repeating the problem in an in deterministic environment , locating the problem in a complex multi-technology environment , analysing the ripple effects of the various solution proposals and achieving an understanding of the complex system were among the most problematic change tasks identified. This category includes problems related to the analysis of the change and change request, and relating the change to the system to be modified. The special features of enterprise software systems are related to the analysis and identified problems as presented in the Table 1.

Table 1. Analysis and identification problems related to enterprise software

Special feature of enterprise systems	Manifestation of analysis and identify problems
Concurrent system engineering	• When incomplete, not properly tested SW components are integrated, it is extremely difficult to find out the actual root causes for errors.
Code optimization	• Optimised source code is difficult to understand. • Special technical features, such as object oriented technology and non deterministic behavior have to be dealt with.

Sharing software components	• The module can be used in several different HW and SW environments, and the effects of the modification has to be analysed in all of them
High reliability demands	• Unwanted impacts of the modification have to be predicted before release, because corrective actions may be impossible or very expensive.

Locating the problems encountered in late testing phases is extremely difficult, because the complete testing environment with all hardware and software components is not available until rather late. The integrated system is very complex. Locating problems and identifying change impacts on highly optimised and source code is challenging [5].

Enterprise systems often have special technical characteristics, which cause problems in analysis and identifying of changes [6]. Examples of such characteristics are in deterministic behavior, which complicates the task of repeating the error situation and hinders the use of automated test result analyzers, and multitasking architectures, which make analysis of the impacts of changes more complex.

Sharing software parts in several product environments complicates analyzing the impacts of the change. The impact analysis of Enterprise software products must also be efficient, since unwanted impacts may create problems which could result in expensive or irremediable error situations afterwards.

Communication problems

This category includes the problems of informing all necessary parties about the change requests, changes and related items.

Communication problems are typical in situations where the software modules are shared by several development projects, and the products or product versions share parts of software or other software-related items, such as test cases or user manual parts. Communication problems also arise between technology groups when the change affects several technological parts of the product.

Communication problems usually have three components: “Who?”, “How?” and “What?”. The “Who?” problems dealt with the problem of identifying who should be informed about the change. Since the amounts of change requests handled in the time critical phases of the development cycles, such as late testing phases, are huge, restricting the distribution list only to the relevant persons is necessary to avoid overloading people. The “How?” problems concerned the processes related to how to distribute information about changes and change needs The processes were very informal in the organizations studied, i.e. the change requests were received in personal conversations by phone or E-mail, and other involved people or projects were informed in regular meetings or in person-to-person communication. The “What?” problem was usually faced when common solutions for the “How?” problem had been defined, for example when common form templates for reporting errors were defined, and the relevant fields had to be identified.

The special features of enterprise software system affect the communication requirements as presented in Table 2.

Table 2. Communication Problems

Special feature of enterprise systems	Effect on communication
Concurrent system engineering	• Close interaction needed between technology groups throughout the development time. • Communication mechanisms have to be usable and comprehensible by all technology groups, not only SW people.
Sharing software parts	• Distributing change information to all parties using modified components. • Distributed decision making when modifications affect modules used by several parties.

Concurrent system engineering results in changes, which have effects on several product technology parts. Therefore, close interaction and communication of changes is required between the technology groups throughout the development time. The development groups usually operate as separate development groups or projects, who work in their own environments. In case one, the development groups were different subcontractors working with separate devices, which would be integrated together in the final product. In case three, the development groups were different technological groups within the company, some working with a specific software component, and others with hardware units. The working practices, tools, and vocabularies between the groups are different.

When several products share common software components, communicating changes to all affected parties becomes crucial. The impacts of the changes on all possible product environments have to be evaluated and considered in order to generate implementation alternatives and to provide adequate and precise information for implementation decisions. The changes have to be distributed and validated in different environments.

Decision-making problems [7]

The decision-making problems arise from the difficulty of defining how the decision-making responsibilities should be assigned to keep the change processes simple enough but still under control. They also include the problem of how to present all relevant information to decision-makers and how to pass the decision and the reasons behind the decision to the implementation and later phases of the development cycle. The problems are closely related to the communication problems.

The most typical decision-making problems were encountered in situations where the responsibilities had not been clearly defined. For example, the customers directly called the software designer to request new features to be implemented, and the feature was implemented by the decision of the designer. This may lead to inconsistencies in project plans, software components, etc. Decision-making problems were also encountered when the responsibilities for implementation decisions depended on the criticality of the change request, i.e. the implementation of a minor request could be decided by the designer, but the major ones had to be handled by the project manager. In these cases the difficulty was in defining the criticality of the

change request.

The decision making problems are very general in nature. The only special feature related to systems is the added complexity of decision-making introduced by the involvement of the hardware development. When the change affects several technology components of the product, the decision-making process has to take into account the hardware development groups, as well. Also, sometimes the change request can be implemented by means of software or hardware, and the decision making process has to evaluate which one is the most beneficial solution option from the product viewpoint.

Effectiveness problems

This category includes the problems related to the effectiveness of identifying needs for change and the effectiveness of the change activities. Effectiveness is defined here as an ability to achieve the desired results, i.e. to perform change management activities so that all relevant changes and change needs will be processed within constraints which are practical and reasonable from the project point of view.

Identifying changes effectively means that all relevant changes and change needs are found and managed as early as possible. The problem covers not only the problem of managing all relevant change requests, but also the problem of avoiding unnecessary change requests, which do not require software changes, to decrease the analysis work needed to process the identified change requests. These change requests include misunderstandings, duplicate change requests, requests requiring user guidance instead of software changes, etc. [2].

The most typical problems and improvement requirements related to the effectiveness of identifying the change needs in time were related to the effectiveness of reviews, inspections and testing. For example, the interviewees in case three felt that reviews were not effective because not enough time was used by the participants to prepare for the review, the work load of the people who possessed the best knowledge about the subject reviewed was already excessive, and poor review or inspection methods were used. The problems in testing effectiveness usually arose from inadequate planning and failure to achieve a sufficient enough test coverage, lack of support for dynamic testing (i.e. repeating test cases, managing test cases and test results, etc.) and difficulty of integration testing before the hardware environment was ready.

The special features of the enterprise software system presented in Table 3.

Table 3. Effectiveness Problems

Special Feature of Software	**Manifestation of effectiveness program**
Concurrent System Engineering	• Early integration testing is difficult, because hardware environment is not ready in early phases. • Hardware requirements are incomplete at the beginning of the project, which leads to requirement changes on software created by hardware.
Code optimisation	• Source code is hard to understand, which leads to difficulties in reviewing source code. • Not much tool support available for testing.
High reliability demands	High costs and practical limitations of after-release corrections require high effectiveness of pre-release change identification.

Concurrent system engineering has two major effects on the effectiveness of change management. Firstly, when the software parts are developed concurrently, the integration of software and cannot be done in the early phases of the development project . This causes delays in finding errors and incompatibilities. Early testing phases have to be performed without the complete environment. This creates delays in detecting inconsistencies or incompatibilities between the software components, and delays the testing of the requirements which are affected by software components, such as power consumption and execution times. Late error detection leads to a far greater amount of correction work than what would be needed if the error is identified and managed at an earlier stage [3][4]. Secondly, changes in one component may lead to changes in the requirements for another component. This can create one additional source for software requirement changes.

Since the reliability demands for the delivered enterprise products are usually high, the need for changing the software after release has to be minimised. The effectiveness of pre-release change identification has to be high in order to avoid after-release change pressure.

Traceability problems

Traceability problems are related to establishing and maintaining traceability links between software artifacts and related items. The traceability links are used for maintaining the consistency of the different abstraction levels of the system, and supporting program understanding by providing information about the related items. Support for establishing and maintaining traceability links between abstraction levels and within abstraction levels, and aid for keeping the system consistent by using traceability information was sought by all case organisations. Manual creation and especially manual maintenance of the traceability links was found to be difficult and error-prone. Also, traceability from software items to design decisions and reasons behind design decisions and other process related information, such as modification records, was regarded as valuable.

The traceability issues are general in nature; they are not directly related to any specific feature of embedded software systems. Development environments for software seldom provide any support for managing traceability is sues, but neither do the general purpose software development environments. The only special feature of systems that has to be dealt with is the traceability between the different technology components of a product.

Tool-related problems

Tool-related problems are problems related to the tools used in change management activities in the organisation. These problems can be formulated into requirements for improved tool support, either for new tools to be implemented or for new, improved revisions of the old tools. Tool-related problems are very company-specific, depending on the specific tools used in the company. Therefore, it is not reasonable to list the typical problems related to companies developing software. Instead, the tool-related problems and requirements can be classified according to the tool functions.

In general, enterprise software development projects seem to require support from process-related tools especially for the communication and information distribution activities between technology groups. The requirements for tools supporting individual tasks, such as configuration management or testing are usually very simple requests for basic support, since the development environments available for enterprise software systems are usually rather primitive.

The requirements for each functionality are briefly discussed in the following subsections.

Process support

Deficiencies in formalising and modeling the change processes were identified in all three case studies. Since the change processes were not formalised, no tool support was available at the time of the studies.

Therefore, the requirements for tool support focused on the basic process sup port functions, such as process monitoring, tracking and process flow control.

Application understanding

All case studies revealed problems in understanding the target system several reasons contributing to the difficulty of understanding the systems built in the case organisations were identified:

- In deterministic nature of the system. The behaviour of the system cannot be predicted from the input values. This causes problems in analysing root causes and repeating cases.
- Understandability of the software is not always one of the most important quality criteria, and it has to be sacrificed because of optimization issues.
- Modification may have ripple effects on several products through common software components. This complicates the analysis, since the software engineer should be able to track several product combinations and analyse the effect of the change in all of those.
- Locating the problem in a complex device with several hardware and soft ware components is often troublesome.

The requirements for supporting application understanding dealt with providing visualisations of the source code and providing information about relationships within the system. The requirements for application understanding tools deal with providing support for understanding source code, since it often is the only reliable material available to help the understanding of an application. The documentation may be out-of-date, or there may be no documentation available. Also, tool support for updating the documentation when the source code is modified was requested.

Reverse engineering

The requirements for reverse engineering tools, i.e. tools which would extract higher level descriptions from lower level artifacts, were very much directed towards the purpose of supporting application understanding tasks and keeping the documentation synchronous with the source code. In addition, the third case study identified the problem of code and system decay, which could be addressed by means of reverse engineering techniques.

Modification request management

Modification requests were collected in all the organisations studied. Three types of requirements for improved modification request management arose:

1. Not all types of modification requests were managed equally well.
2. The procedures for managing modification requests were not followed for some reason.
3. Modification request management should be supported by adequate tools and databases.

Impact analysis

Impact analysis support was requested in the context of analysing the ripple effects of the modification in order to ensure the consistency of the system after the modification. The ripple effect analysis can be automated only if the links between the system items are somehow known. For that reason, the traceability links between the system components have to be managed somehow. Manual creation and maintenance of traceability links was regarded as unreliable.

Regression testing

Change management related requirements for testing support arose from two directions:

1. Testing activities should identify errors (i.e. inputs to the change management process) effectively and efficiently.
2. Regression testing of a change should be supported.

The special problem in testing support within the context of embedded software is that the development environments seldom provide advanced tool support for testing activities [9] and the testing environments with simulators and emulators are complicated to build and maintain.

Software configuration management

Cases one and two had simple software configuration and version management systems in use, but no defined or established practices for using them. Both organisations were in the process of establishing consistent practices, and the change management related configuration management requirements were very much related to this improvement initiative.

The case organisation had configuration management procedures in place, and its requirements for software configuration management tools related to change management were focused on the issue of integration of configuration management and change management tools.

Case organization also had a problem of managing the versions of the development tools, such as compilers and debuggers. As the development tools evolve, the software parts developed using the older versions of the tools may become inconsistent with the new versions of the development tools. This creates problems when modifications to these software parts have to be made.

3. Summary

This paper presented the classification and summary of change management related problems and requirements derived from the literature study and case studies presented in the previous chapters. The problem classes were characterised, and the special problems identified in the change management of enterprise software that were identified through the case studies were summarised.

References

[1] Barros, S., Bodhuin, T, Escudie, A, Queille, J. & Voidrot, J. 1995. Supporting impact analysis: a semi automated technique and associated tool. IEEE Software. IEEE. Pp. 42 – 51.

[2] Stark. G., Kern, L. & Vowell, C. 1994. A software metric set for program maintenance management. Journal of systems and software. Elsevier Science. (Vol. 24). Pp. 239 – 249.

[3] Humphrey, W. 1989. Managing the software process. Addison Wesley Publishing Company, Inc. 494 p. ISBN 0-201-18095-2 IEEE 828. 1990. IEEE standard for software configuration management plans (ANSI). IEEE. 16 p.

[4] Ince, D. 1994. Introduction to software quality assurance and its implementation. McGraw-Hill. 202 p. ISBN 0-07-707924-8

[5] Jones C. 1996. Strategies for managing requirements creep. Computer. (June 1996 Pp 92 – 94).

[6] Harjani, D.-R. & Queille, J.-P. "A process model for the maintenance of large space systems software", Proceedings of Conference on Software Maintenance. Los Alamitos: IEEE Computer press. Pp. 127 – 136. ISBN 0-8186-2980-0, 1992.

[7] B.Boehm, "A spiral model of software development and enhancement", Computer. (Vol. 21.) pp. 61 – 72. ISSN 0018-9162, 1988.

[8] S.M.Ghosh , H.R.Sharma, V.Mohabay, "Analysis and Modeling of Change Management Process Model" IJSEIA – Vol-5 No-2,ISSN : 1738-9984, April 2011.

[9] Vierimaa, M., Kaikkonen, T., Oivo, M. & Moberg, M. 1998. Experiences of practical process improvement. Embedded Systems Programming Europe. (Vol. 2, No. 13). Pp. 10 – 20.

Authors

S.M.Ghosh

Director, Prism School of Business and Entrepreneurship

He received his Master Degree in Computer Application and authored 3 books i.e. Software Engineering , Introduction to Computing and Management Information System. Published research papers in Software Change management related topics in National and International Journals. Designed Enterprise Resource Planning Software for Job based fabrication and Casting Steel Industries. Teaching QMS , Software Engineering and MIS to the PG Students from last 12 years.

Dr. H.R. Sharma

Director, Department of Computer Science, Chhatrapati Shivaji Institute of Technology

He received his M.Tech degree from Delhi University in Computer Science and has completed his Ph.D from IIT, Delhi. He has also published more than 380 research papers at National and International level. He has supervised 9 research projects sponsored by UGC & AICTE. He has more than 45 years of teaching experience at Graduate and Post Graduate level. He is presently working as the Director of Chhatrapati Shivaji Institute of Technology, Durg.

Dr. V.K.Mohabey

Head of Department, Depatment of Electronics and Information Technology, Kalyan Post Graduate College

He has published more than 356 research papers in which 68 are published at National level and 12 at International level. He has supervised 7 research projects sponsored by UGC, DST, MAPCOST, etc. He is presently working as a Professor and Head of Department of Electronics and Information

Technology in Kalyan PG College, Bhilai. He has over 35 years of teaching experience at Graduate and Post Graduate level.