PRACTICAL EXPERIENCE WITH THE CHANGE MANAGEMENT PROCESS IN SOFTWARE DEVELOPMENT

Martin Štufi[1], Dejan Rančić[2], Milica Ćirić[2], Aleksandar Stanimirović[2], Leonid Stoimenov[2]

[1]*Solutia, s.r.o., Prague, Czech Republic*

[2]*Faculty of Electronic Engineering, University of Niš, Serbia*

Abstract - *Complexity of the software development process has greatly increased compared to early days of software development. Considering that new software development methodologies allow changes in each phase of the process the need for using a change management system is obvious. These systems enable the software development team to respond to customer's change requirements and changes to the business process and business model while maximizing value and minimizing incidents, disruption and reworking. In this paper we describe the role of change management systems and our practical experience with the change management process in the Solutia s.r.o. company.*

1. INTRODUCTION

Software development has changed dramatically over the last decade. It is moving towards a more agile and more flexible approach which corresponds to an environment where technical, financial and strategic constraints are changing constantly [1].

Software development is a managed process organized to provide a software product built in line with the customer requirements [2]. It usually comprises of research, development, prototyping, modification, reuse, re-engineering, maintenance and other activities. During any of these activities a need for change may arise.

With the current economic climate, competitive pressures and technology change rate businesses have to adapt their processes and models constantly [3]. The purpose of the change management process in software development process is to manage the lifecycle of all changes and enable the implementation of beneficial changes with minimal disruption to IT services.

Change management process enables an enterprise to change its business process and model and address the technology changes that emerge as a consequence. In this way change management helps in improving availability, performance and throughput and reducing errors, planned and unplanned downtime.

Change management systems must version control all types of objects. Version control system is a software system for managing multiple revisions of the same unit of information. One of the key factors for managing an IT project in engineering and software development is the managing of ongoing evolution of e-files (documents) such as application source code [4]. Changes to these documents are identified by software that takes care of versioning control through "revision number", "revision level", or simply "revision" and associates them with the person making the change.

In this paper we will first describe the need for change management that exists in software development process. We will then give an overview of change management systems, focusing on change requests. Next, we will specify the conditions a change management system must fulfill in order to represent an overall solution for software development teams. The final chapter of this paper contains information about our practical experience with the change management process in Solutia s.r.o. company from Prague.

2. SOFTWARE DEVELOPMENT AND THE NEED FOR CHANGE MANAGEMENT

The term software development may be used to describe programming activity that occurs in order to develop source code which has to represent a software product built in line with the customer requirements. But software development usually includes other activities that combined should result with a software project. It is a narrowing of the focus of software engineering to just that part concerned with the creation of the actual software. And it's a broadening of the focus of programming to include analysis, design and release issues [5].

A software development methodology is a structure imposed on the development of a software product. It

includes procedures, techniques, tools and documentation aids which help system developers in their task of implementing a new system [6]. The intent of a methodology is to formalize what is being done, making it more repeatable.

In the earliest days of software development, code was written and then debugged (Code-and-Fix Model) without any formal design or analysis [6][7]. The approach for developing complex hardware systems was used as a model for developing software. The idea was to approach system development with a standard and well defined process using a requirements/design/build paradigm. The resulting methodologies are now known as Heavy Methodologies or Plan Driven methodologies [6][7]. They implied a disciplined process and created the need for complete requirements prior to start. However, that resulted in slow respond to change.

Newer methodologies then started making an appearance in software projects. They are considered more agile and more adaptable to change. Rather than focusing on a long development cycle, they consist of short iterations, lightweight processes and rely heavily on close customer involvement. These types of methodologies have come to be known as Light or Agile Methodologies [6][7] and later on Advanced Software Development Methodologies [6].

Not very long ago, a software development team would usually work on a single release of software, ship that release, and then move into a maintenance phase where the team would issue updates, changes etc. However, today a software development team often has complex relations and requirements for development and software delivery which is part of almost every IT project [8].

In a typical case there are multiple teams working on the same code base, working simultaneously on releases, patches, maintenances releases etc. and delivering multiple variants of the code for different platforms, operating systems, databases. This kind of parallel development is fundamental for a modern Software Configuration Management (SCM) system. On the other hand, size and complexity of applications continue to increase rapidly. There are constantly new demands for functionality which typically results in larger and more complex applications. One of the most influencing factors in determining the need for software configuration management is the size of the team. As the team grows in size, the problems of communication, coordination, integration, conflicts, and parallel development can quickly become overwhelming.

3. CHANGE MANAGEMENT SYSTEMS

We can distinguish two key factors which that lead to creation of change management systems. First, larger teams developed larger and more complex application, which combined with more and more legacy systems created a critical need for tracking changes to software. The initial attempts at tracking changes were typically disconnected from the systems that managed the source code. There were usually two systems - one to manage the source code and the other to manage the changes to the source code. This kind of systems had only limited success.

Second, a close relationship between SCM and process was acknowledged by software development teams in many companies. They needed to write their own layers of process and workflow on top of the SCM system since they were missing a major component in their environment.

The goal of the change management process is to respond to the customer's change requirements and demands of business and IT change while maximizing value and reducing incidents, disruption and reworking. All of the changes must be recorded and evaluated, planned, tested, implemented, documented and reviewed appropriately. The overall business risk should be optimized, not always minimized, because sometimes it is appropriate to consciously accept the risk because of the potential profits.

Change in software development can be a change in specifications, user requirements, design change, code change or so on. Changes can be requested from various sources including customers, end users, the project team or the test team. Customers and end users usually make changes in their requirements, the project team might ask for design changes and the testing team could request code changes. Changes are communicated to the Software Project Manager (SPM) using a Change Request (CR) form. The CR should contain details of the project, module and component which are likely to be affected by the CR and may include reasons for the CR.

In many organizations, the customer support organization is the interface between the development team and the customers, so change requests may be sent from customers via the technical support engineers. In others, change requests may be issued directly from customers to the development organization.

Change requests may refer to existing products, new products currently under development or in test, or new product efforts not yet started. In large software systems, managing the volume of change requests is a significant task in its own right, so tools for tracking, categorizing and prioritizing become increasingly important.

When a CR is received it is first recorded in the CR Register and then the analyzed by the SPM or a person designated by them. Sometimes, especially in large projects, there is a Change Control Board (CCB) or Change Advisory Board (CAB) that analyzes the CR and approves or rejects it. Analysis determines whether the implementation of the CR is feasible, the amount of effort and calendar time it would take to implement it and the impact of the CR on the overall project – especially in terms of effort, schedule and cost.

If the CR is rejected, the decision along with the reasons is communicated to the originator of the CR and the CR is closed in the CR Register. In the case of approval the CR is implemented in accordance with the decided strategy which is recorded in the Software Configuration Management Plan (SCMP). Some of the possible strategies for CR implementation are [9]:

- Implement CRs as and when received, immediately on receipt
- Collate all CRs and retrofit them at the end of development
- Situational implementation – i.e.:
    - If the component affected by the CR is yet to be coded or is being coded, then implement it after the component is coded.
    - If the coding of the component affected by the CR is completed, keep the CR pending and retrofit it at the end.

Change Management incorporates modern software configuration management, process management and change tracking into an overall solution for software development teams. There are five distinct but interrelated components of the Change Management solution [8]:

- Object Management
- Process Management
- Work Area Management
- Build Management
- Change Tracking

The object management system must version control all types of objects, not just source files; but also directories, libraries, executables, designs, documents, schedules, test cases, test results, help data, make files, configurations, and so on.

The process management system must support the concept of security, access control and roles, so that certain users can be restricted from certain operations when that is what the team desires. Moreover, the process management system is responsible for enforcing the overall flow of objects through the organization, from development to test to release management.

The work area management system must be able to reproduce earlier configurations on demand, including the versions of programs and libraries that were constructed from that configuration.

The build management system should be an integral part of the SCM system, able to read and set variant related attributes, create and register products, share reusable products, and understand the workflow of products.

The change tracking system needs to provide full traceability between the change request, the tasks to implement the request, and the physical files that were changed in performing the tasks. Moreover, the change tracking system should provide the user with ability to work at the level of logical changes rather than individual files.

Some of the benefits of proper change management process are [3]:

- Better alignment of IT services to business requirements.
- Increased visibility and communication of changes to both business and service-support staff.
- Improved risk assessment.
- Better assessment of the cost of proposed changes before they are incurred.
- Fewer changes that have to be backed-out, along with an increased ability to do this more easily when necessary.
- Improved problem and availability management through the use of valuable management information relating to changes accumulated through the change management process.
- Increased productivity of key personnel through less need for diversion from planned duties to implement urgent changes or back-out erroneous changes.
- Greater ability to absorb a large volume of changes.

4. CHANGE MANAGEMENT IN PRACTICE

The change management process in Solutia, s.r.o. company deals with managing the lifecycle of all changes and enabling the beneficial changes to be implemented while keeping the disruption of IT services to a minimum. Our company's change management process is fully

organized in line with ITIL® v3 (Infrastructure IT Library version 3) [10] and ITSM (IT Service Management).

Change management process step	**Person responsible**
Create and record	ChReq initiator (customer side)
Evaluate and rate	ChReq administrator
Approve and test change	Software Project Manager or Key Accountant Manager
Coordinate and test ChReq	Quality Assurance Manager
Approve ChReq	Software Project Manager
Coordinate deployment and test change	Quality Assurance Manager
Review and enclose ChReq	Software Project Manager

Table 1. Person responsible for change management steps in Solutia

The Table 1 shows the person responsible for each step of the change management process in Solutia, from the creation of the Change Request (marked as ChReq) to the closure of the request.

Figure 1 shows the change request form in Solutia. This form contains information about the desired change. The change's priority is defined as high, medium or low. If the change does not significantly affect the supply of the product/project, its priority is low. If the change has strong link to the product/project, its priority is medium, and if it has a key impact on the delivery of the product/project, its priority is high. The *Business requirement specification* field should contain description of the proposed changes, the reason for the proposed changes, the impact of the proposed changes and the consequences of failure, background. The field *Analysis of business requirements* should contain results of performance analysis requirement and impact of the project cost, the recommended next steps, etc.

The change management process flow is shown in Figure 2. First SPM reviews and assesses the information provided in the proposed change request, and determines whether to accept or reject the proposed change request. If the change request is accepted, the class of the change should be verified and the approval should be documented in order to implement the change. SPM then validates the

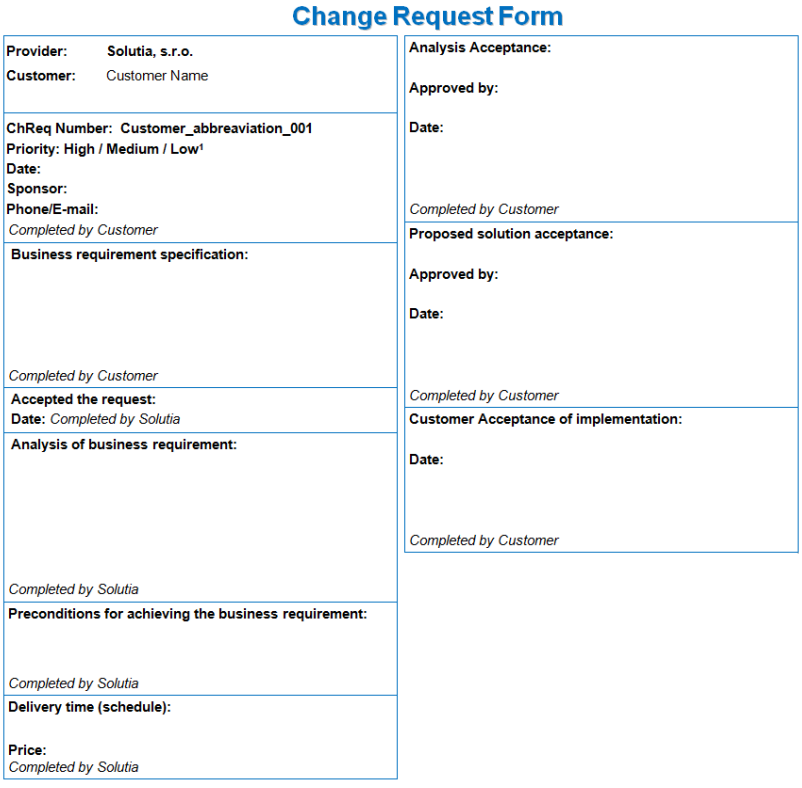

Figure 1. Change Request Form in Solutia

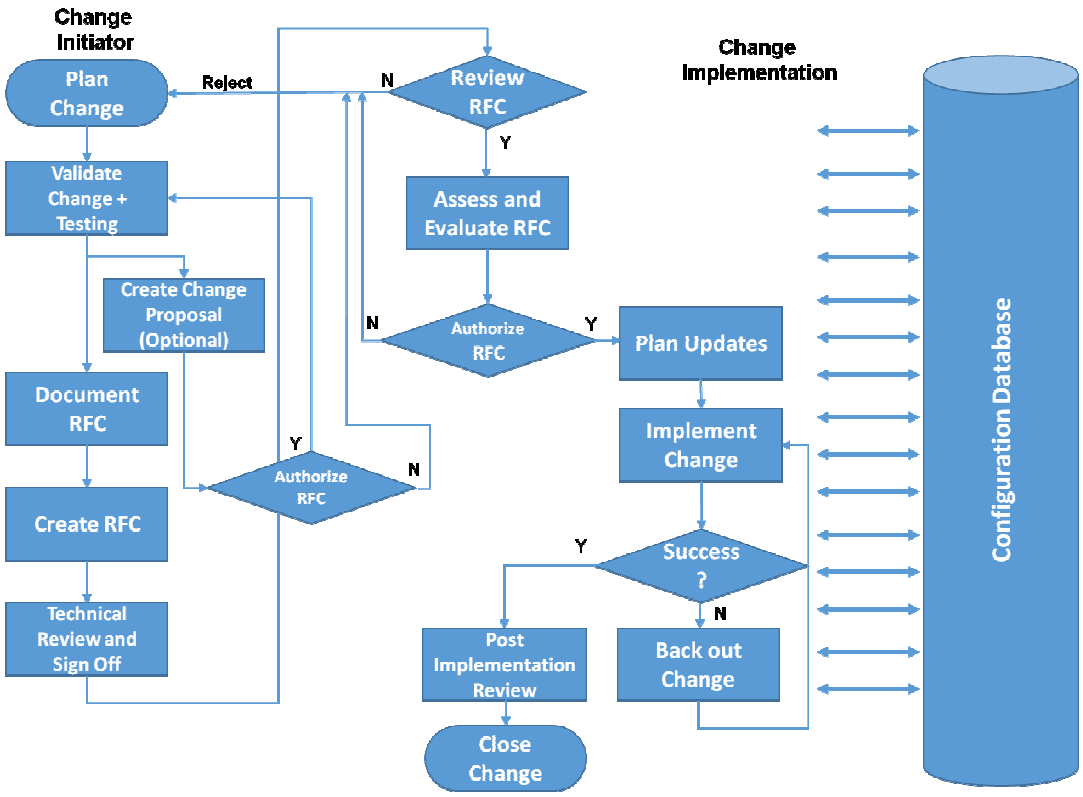

Figure 2. Change Management Process Flow in Solutia

change from impact assessment point of view and creates test scenarios to ensure successful create of change request proposal and then the proposal for change request may be created. Next, the request change should be documented using the company's standardized company template and then a new change request record is created. After creating the record a technical review follows, i.e. making that everything is done and ready for review from technical perspective of view. Reviewing the change request practice helps us to improve the change management process. Usually, there are multiple change requests, so there is a need prioritize the change requests by assessing the cost involved, benefits, perceived risks and possible impacts of the change request. Business partners are normally engaged in the process of authorizing the change request; their expertise in evaluating the impact of the changes to the whole system is very valuable. Updates must then be planned to ensure that the change implementation plan is complete and that there are no conflicts with other planned changes. Only then can the change be implemented according to the implementation plan. In the case that the change request was implemented successfully the post implementation review follows. After the change is reviewed and accepted by the user who has submitted the change request, the change request should be closed. All configuration changes should be properly and systematically documented, and all approvals, comments, testing plans, back-out plans and other workflow-related documentation is kept according to company policy. On the other hand, if the applied change didn't perform as expected or has lead the system into an unstable state the change has to be backed out. In that case we should return to the implementation of the change.

For Change Management Process evidence and software development issue capturing in Solutia we're using Change request and Issue/bug Tracking System called Eventum [11].

Eventum is a user-friendly and flexible issue tracking system that can be used by a support department to track incoming technical support requests, or by a software development team to quickly organize tasks and bugs. It is also used by the MySQL Technical Support team. Eventum captures the overall structure of the project, project specific attributes such as categories, priorities, statuses etc., users with different permission levels for different projects, change request or issue entries etc.

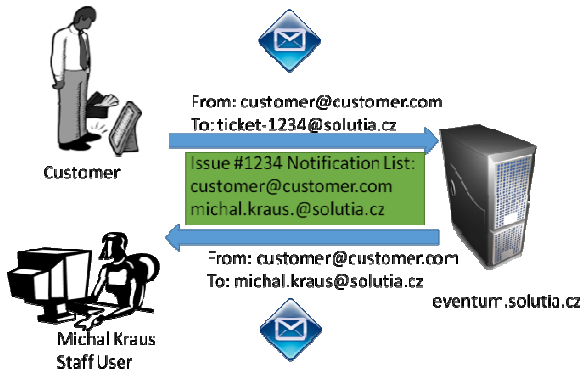

Figure 3. Principle of the notification interface with a customer

Figure 3 shows the principle of the notification interface with a customer in Eventum. Reaction to a customer's email is a creation of a new ticket. Customer sends an email to a predefined email address and Eventum responds to this event through a notification email to a staff user. That staff user is then responsible for the created ticket.

In a software tool for Change Management and issue tracking evidence such as Eventum there are two types of reporting tools:

- Statistic reports – related with the Change request or issue based on ChReq Id or Issue Id and
- Global Reporting System

Statistic reports show number of change requests or issues by status, priority, assignment to a particular user, change request link with the release, category of the change request or issue, etc.

Global Reporting System has various reporting possibilities. Issues can be reviewed displayed by a user they are assigned to, or by a user that has reported these issues. We can display all issues or only those that are still open. Weekly or monthly reports are also possible to generate. For a given time period the workload can be reviewed as a bar graphics and a table with workload by the time of day. Of course, the system can also generate custom reports with desired fields.

All of these reports can be uses to manage changes during the course of a project. You can assess the health of a project and proactively take corrective action by monitoring parameters such as newly arrived issues, resolved or closed issues and number of activities allocated per person.

5. CONCLUSION

In conclusion, it is virtually impossible to engage in software development process without a change management tool. The reasons can be found everywhere, from the complexity of software development even in small teams to the SOHO (Small Office/Home Office) trend that emerged thanks to internet penetration, clouds, project management principles, creation of virtual realization teams and so on.

One of the benefits of the change management process is the uniformity of the process regardless of the size of the company. Whether the company has 5, 20 or 100 workers the same internationally recognized processes are followed, most of which are formalized as ISO standards (ISO 10006 for project management, ISO 12207 for software development, ISO 20000 for ITI v3).

6. REFERENCES

[1] Stober, T., Hansmann, U., *Agile Software Development: Best Practices for Large Software Development Projects*, Springer, 2009.

[2] Sommerville I., *Software Engineering*, Addison-Wesley, 2010.

[3] *What is Change Management?*, available at http://www.clemson.edu/ccit/help_support/cm/what_is_cm.html

[4] Collins-Sussman, Fitzpatrick B. W., Pilato C. M., *Version Control with Subversion For Subversion 1.7,* 2011.

[5] Dooley J., *Software Development and Professional Practice*, Apress, 2011.

[6] Faridani H., *A Guide to Selecting Software Development Methodologies* , 2011.

[7] Misra S., Kumar V., Kumar U., Fantazy K., Akhter M., *Agile software development practices: evolution, principles and criticisms*, International Journal of Quality and Reliability Management, Emerald Group Publishing Limited, 2012.

[8] *Continuus/CM: Change Management for Software Development*, available at http://net1.ist.psu.edu/ist420/readings/r5.pdf

[9] *Change Management in Software Development Projects*, 2011., available at http://www.brighthubpm.com/change-management/5737-change-management-in-software-development-projects/

[10] *Information Technology Infrastructure Library* (ITIL), http://www.itil-officialsite.com/

[11] Eventum, http://eventum.wikinet.org