Change Management in Project

Changes are of 2 types

- any proposed deviation from any part of the project management plan or
- any alteration to specifications, processes, and procedures

If an adjustment or correction is needed to the scope, schedule, activities, efforts, costs, budget, quality, staffing, risk components, resources, or contracts then that change will impact the project management plan which, if unmanaged can disrupt the project plan and work being done.

Change can occur as a result in a modification to specifications of the product or a process involved in creating the deliverables if unmanaged, these changes can introduce ripple effects that go unnoticed until suddenly they result in mayhem

Change Aspects in the Project Context pictorially

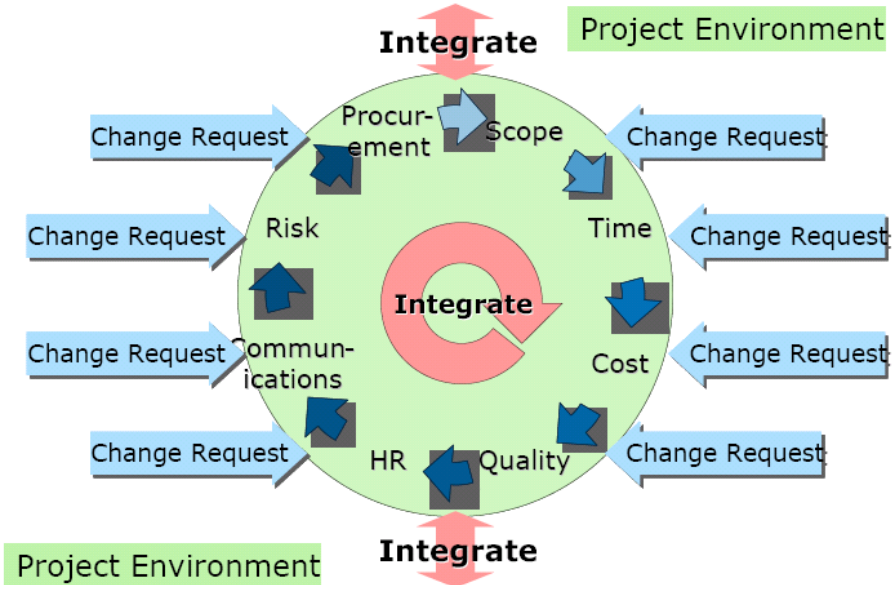

Figure 1. Change aspects in the project context

Sources of the Change

Changes are necessary in order for the project team to be responsive and adaptable to evolving customer, organizational, and project management needs. Changes are usually driven by one of the following needs:

- Value added - The change would be beneficial to the deliverable or project
- External events - The change is a reaction to an event triggered outside the project boundaries - political, legal, economical, social, technological etc changes
- Risk responses - The change is needed to take advantage of an opportunity, reduce the chance of a negative event occurring, or is in response to an unplanned event that´s currently taking place

- Errors and omissions - The change is needed because of an oversight or defect, or the change is needed because the iterative nature of the project has exposed new knowledge

Under oversight belong:

- scope creep - scope is being altered over time without proper change management applied
- hope creep - team member being behind schedule but reporting to be on schedule
- effort creep - team member working but not making progress
- feature creep - team members arbitrarily adding features and functions, no proper change management applied

Integrated Change Control

Three main objectives of change control:

- influence the factors that create changes to ensure they are beneficial;
- determine that a change has occurred;
- manage actual changes when and as they occur

Provides a mechanism to make sure that needed changes don´t overwhelm the project and that they´re properly managed by:

- identifying and logging all request for changes
- analyzing, evaluating, and documenting the impacts that the required changes will have throughout the project
- defining a method for the formal review, make-up of the change control board, and decision-making authority of changes
- ensuring that communication is thorough and complete about all changes to the stakeholders and project team

Change Control System

A formal, documented process that describes when and how official project documents and work may be hanged. Describes who is authorized to make changes and how to make them. Is connected to change control board (CCB), configuration management, and a process for communicating changes. CCB is a formal group of people responsible for approving or rejecting changes on a project. CCBs provide guidelines for preparing change requests, evaluate change requests, and manage the implementation of approved changes. Includes stakeholders from the entire organization

CCS is a component of configuration management system. CCS is focused on changes that directly impact the project management plan: scope, schedule, budget, cost, quality, risk, and procurement management plans. Pictorially:

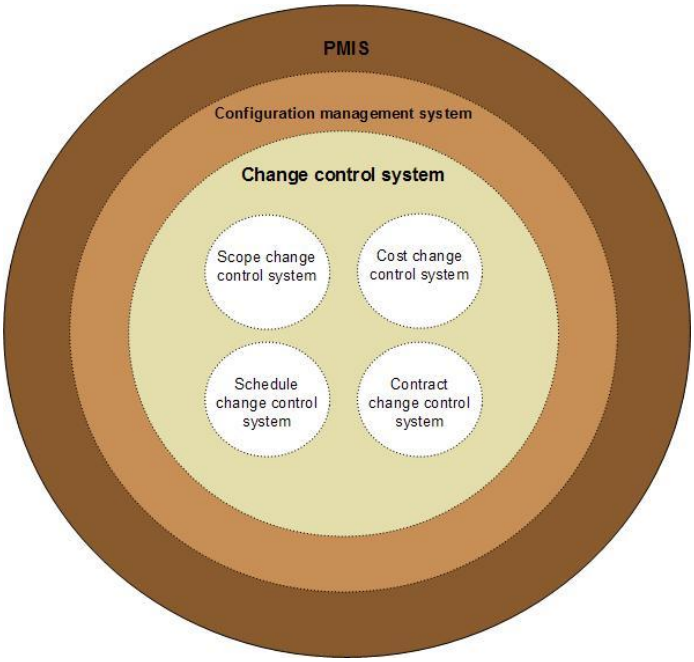

Figure 2. Relationships between configuration and hange management system

Configuration Management ensures that the products and their descriptions are correct and complete. Concentrates on the management of technology by identifying and controlling the functional and physical design characteristics of products. It can also be used to track product specifications, processes, policies, and procedures, providing an approval mechanism for changes to them. Configuration management specialists identify and document configuration requirements, control changes, record and report changes, and audit the products to verify conformance to requirements

Example of Integrated Change Control Process

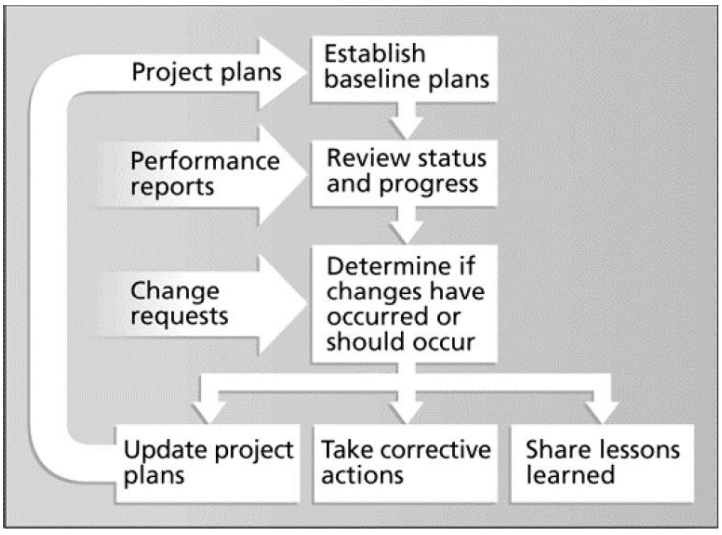

Figure 3. Example of change management process

Scope Control

Is concerned with influencing the factors that create project scope changes and controlling the impact of those changes. Assures, that all requested changes and recommended corrective actions are processed through the project Change Control System. Monitors performance to look for scope variances between what is occurring to what was planned for. Manages change requests for preventative or corrective actions to bring actual performance back in line with what was planned, or to alter the plan to reflect a changed situation

Approved scope changes have to be managed rigorously; communicated to the stakeholders and project team, and incorporated into all effected components of the project management plan. Approved scope changes will result in updates to the overall project performance baseline.

Changes to Development Artifacts in RUP

Changes to development artifacts are proposed through Change Requests (CRs) - a formally submitted artifact that is used to track all stakeholder requests. Change Requests are used to:

- document new features;
- document and track defects;
- enhancement requests and any other type of request for a change to the product along with related status information throughout the project lifecycle

The benefit of CRs is that they provide a record of decisions and, due to their assessment process, ensure that change impacts are understood across the project. Change Control Manager is responsible for Change Requests. Change request management responsibilities and artifacts pictorially:

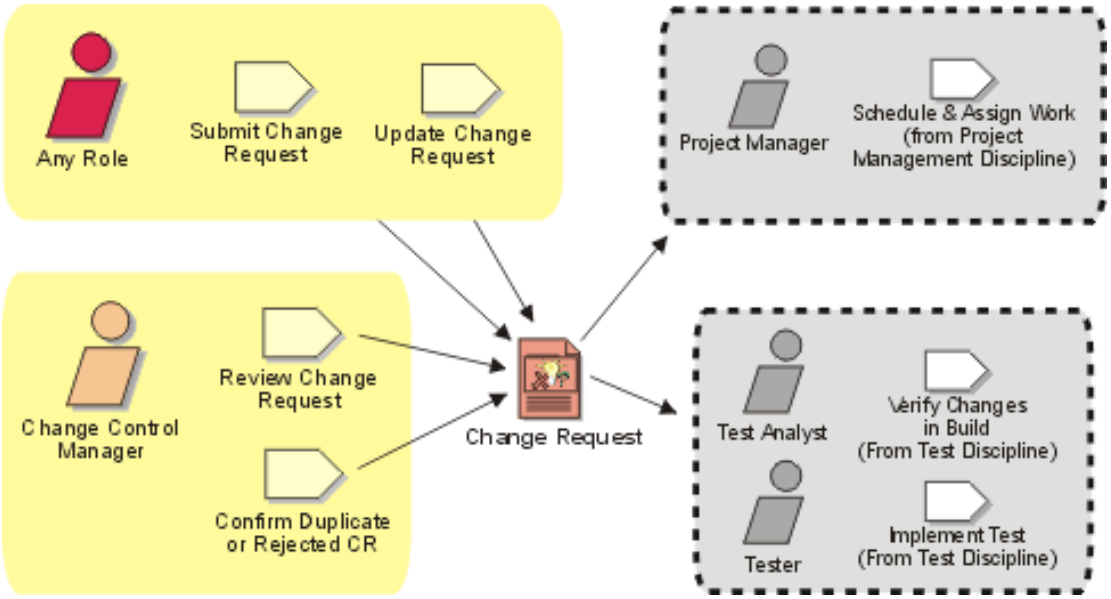

Figure 4. Change management responsibilities and artifacts in RUP

Change Request Management Process

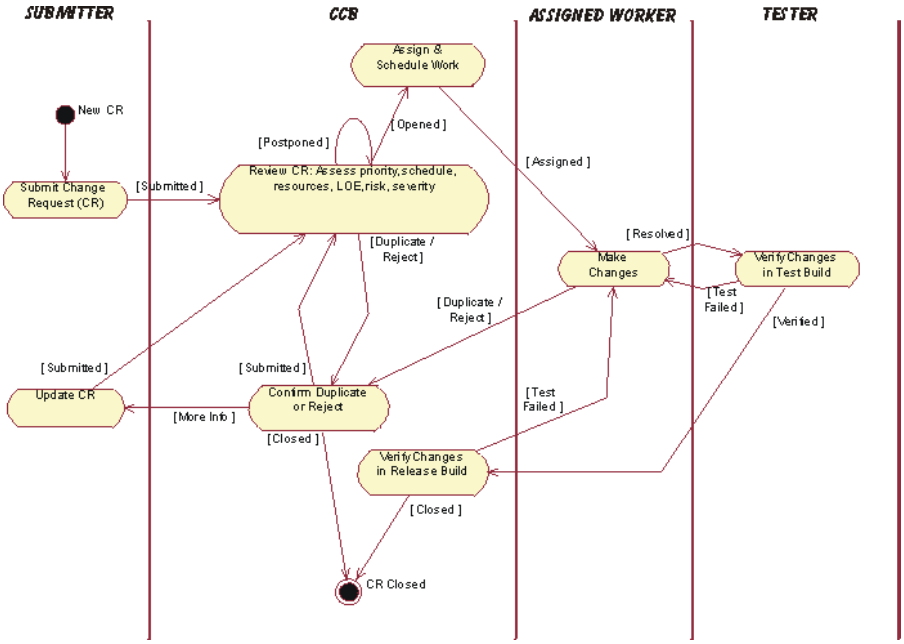

Figure 5. Change management process in RUP

Change Request States

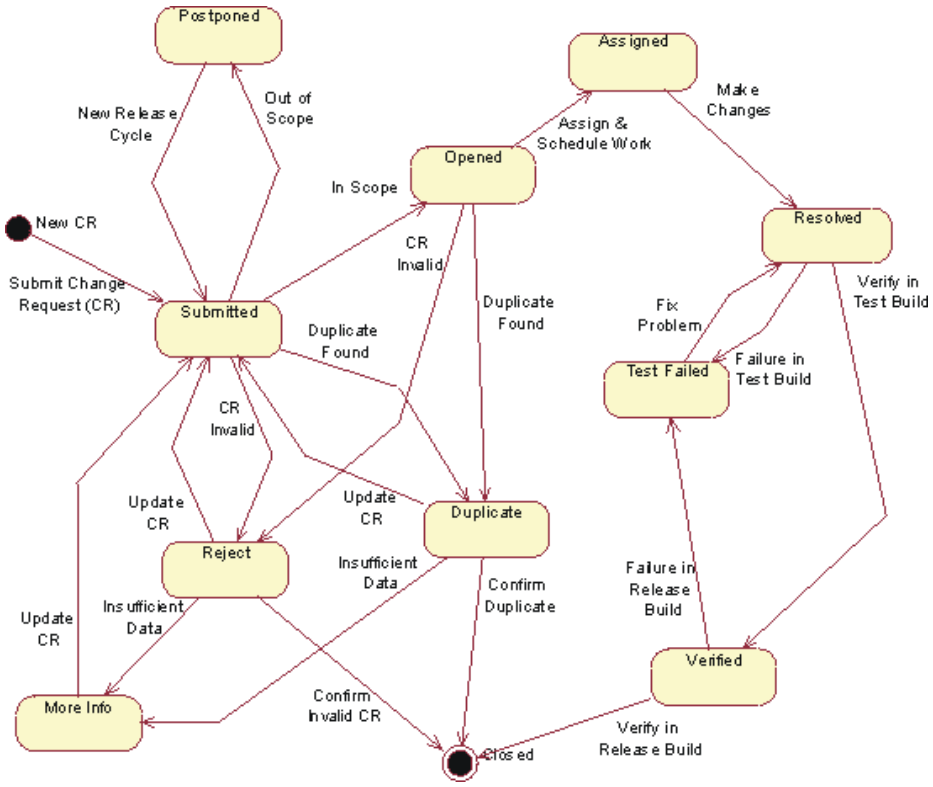

Figure 6. Change request states

By Implementing change we must consider:

effort - How much work is needed to make over again? How much additional work must be done?

complicity - Is requested change easy to implement? What is the possible impact to others system components?

severity - What kind of is the impact of not to implement? Is there any lost of work already done or some kind of data?

impact - What are consequences of implementing change? What are consequences of not implementing change?

schedule - When this change is needed? Is this temporally achievable?

costs - What is cost or saving of change? authority Is there authority of implementing change request?

Changes regarding development process

A Methodology-Growing Technique - Just-in-time (On-the-fly) methodology construction and tuning by Alistair Cockburn

What to do at 5 different times

1. right now (before project start)

2. at the start of the project
3. in the middle of the first increment
4. after each increment
5. in the middle of subsequent increments

1. Right now

Discover the strengths and weaknesses of your organization through short project interviews. During interview:

- ask to see one sample of each work product produced;
- ask for a short history of the project;
- ask what should be changed next time;
- ask what should be repeated next time;
- identify priorities;
- find any holes

2. At the start of the Project

Expect to do some tailoring to the corporate methodology standard. This will be needed whether the base methodology is ISO9001, XP, RUP, Crystal, or a local brew. Perform 2 steps:

2.1. tune the base methodology;

2.2. hold a team meeting

2.1. Tune the Base Methodology

- Determine how many people are going to be coordinated and identify their geographical distribution
- Decide what level of correctness is expected of this software and what degree of damage it could cause
- Determine and write down the priorities for the project: time to market, correctness etc
- Select the basic parameters for the methodology
    - How tight the standards need to be
    - The extent of documentation needed
    - The ceremony in the reviews
    - The increment length - The time period until running code is delivered to real, even if sample, users. If the increment length is longer than 4 months, team will have to find some way to create a tested, running

version of the system every 4 months or less, to simulate having real increments

Select a base for the methodology, one that is not too different from the way in which team would like to work. It is easier to modify an existing methodology than to invent one from the scratch. Boil the methodology down to the basic work flow involved - who hands what to whom and conventions - team thinks they should agree to

2.2. Hold a Team Meeting

To discuss the base methodology´s work flow and conventions, and adjust it to become the starter methodology. The purpose of the meeting is to catch embellishments; look for ways to streamline the process and ways to communicate with less cost; detect other issues that were not spotted in the base methodology draft.

Questions to Consider in that Meeting: How long are the iterations and increments to be (and what is the difference)? Where will people sit? What can be done to keep communication and morale high? Which work products and reviews will be needed, at what ceremony levels? Which standards for tools, drawings, tests, and code are mandatory, and which are just recommended? How will time reporting be done? Which other conventions should be set initially, and which might be evolved over time?

The meeting results will include Basic work flow; Hand-off criteria between roles, particularly including overlapped development and declaration milestones; Draft standards or conventions to be followed; Peculiarities of communication to be practiced

3. In the middle of the First Increment

Run a small interview with the team members, individually or in a group meeting, allowing 1-3 hours. Whether your increment length is 2 weeks or 3 months; At approximately the mid-point of the increment. A single question for resolution is “Are we going to make it, working the way we are working?”. In the first increment, you can´t afford to change your group´s whole way of working unless it is catastrophically broken

What you are looking for is to get safely to your first delivery. If the starter methodology will hold up that long, you will have more time, more insight, and a better moment to adjust it - **After** you have successfully made your first delivery. The purpose of this interview or meeting is to detect whether something is critically wrong and whether the first delivery will fail. If you discover that team´s way of working isn´t working First consider reducing the scope of the first delivery. **Most** teams overstate how much they can deliver in the first increment. This is simply normal and not a fault of methodology; It is a result of over-ambitious management driving the schedule unrealistically and overly optimistic developers who overlook the learning to be done, the meetings to be held, and the normal bugs they put into the code; It comes from underestimating the learning curve of new technology and new teammates. You may, however, discover that

reducing scope will not be sufficient. You may discover that the requirements are incomprehensible to the programmers or that the architects won´t get their architecture specification finished in time. If this is the case, then you need to react quickly and find new way of working. This, combined with drastically reduced functional scope, will allow you to meet that first delivery deadline. You may introduce overlapped development or put people physically closer together, cut down the ambition level for the initial architecture or make greater use of informal communication channels You may have to make emergency staff changes or introduce emergency training, consulting, or experienced contractors. Your goal is to deliver something: some small, running, tested code in the first increment This is a critical success factor on a project. After you deliver the first release, you will have time to pause and consider what is happening

4. After Each Increment

Hold a team reflection workshop. *Bothering to reflect* is a critical success factor in evolving a successful methodology; Just as incremental development is a critical success factor in delivering software. The dominant reason for delaying this workshop until after the first increment is that you can only properly evaluate the effects of each element in your methodology after you have delivered running, tested software to a user. Only then can you see what was overdone and what was underdone. This meeting provides a chance to breathe and reflect. Done regularly, it becomes part of the project rhythm. After each increment, the team members benefit from a short shifting of mental and social gears. 2 questions to address: “what did we learn?” “what can we do better?”. Very often, teams tighten standards after the first increment, get more training, streamline the work flow, increase testing, and reorganize the teaming structure

5. In the Middle of the Subsequent Increments

After the first increment, the team has established one (barely) successful way of working – this a methodology design to fall back on, if needed. Having that as a fallback plan, you can be much more adventuresome in suggesting changes in the mid-increment meetings you hold in the second and later increments. In those mid-increment meetings, and particularly after the second successful delivery look to invent new and better ways of delivering. See if you can do any of the following:

- Cut out entire sections of the methodology;
- Do more concurrent development;
- Use informal communications more to bind the project information;
- Introduce new and better testing frameworks;
- Introduce new and better test-writing habits;

- Get closer collaboration between the key groups in the project: between domain and usage experts, programmers, testers, training people, the customer care center, and the people doing field repair

In inventing new ways of working in these later increments, you create the opportunity to significantly improve your methodology. This is an opportunity not to be missed

Agile Principles in Change Management

- Assume Simplicity - as the project evolves it should be assumed that the simplest solution is the best solution
- Embrace Change - since requirements evolve over time
- Incremental Change - Instead of futilely trying to develop an all encompassing project plan from the start, put a stake in the ground by developing a small portion of the system, or even a high–level model of a larger portion of the system, and evolve this portion over time
- Rapid Feedback - the time between an action and the feedback on that action must be minimized

Summary

Uncontrolled changes make confusion, what in turn decrease commitment to project – the result is project failure. Changes under control give confidence against unpleasant surprises for all stakeholders. Change management creates transparency and accountability in project. Balance in implementing changes indicates good project management practice. Change is not to be afraid of, but we must be able to deal with it

Used Literature

- Alex Sherrer, Project Management Road Trip, Chapter 4: Project Integration Management, Section 4.5: Perform Integrated Change Control, , http://www.pmroadtrip.com/pmpv4_04e.html
- Project Integration Management, http://www.cs.su.ac.th/course/517537/slide/lecture3.pdf
- Herbert Gonder “Change Management - PMBOK® says...”, http://www.pmi-muc.de/Vortraege/20071203/CM-PMBOK-20.pdf
- Change Management Metamodel, http://en.wikipedia.org/wiki/File:Metamodel_change_management.png
- Configuration Management, http://en.wikipedia.org/wiki/Image:ConfiurationActivityModel.png
- Change Control Process Template, http://www.processimpact.com/process_assets/change_control_process.doc

- Simon Wallace, The ePMbook, www.epmbook.com
- Michael D. Taylor “How To Control Changes to The Project”, http://www.pmhut.com/how-to-control-changes-to-the-project
- William Ibbs, Clarence K. Wong,Young Hoon Kwak “Project Change Management System”, http://home.gwu.edu/~kwak/PCMS.pdf
- RUP, http://www.cin.ufpe.br/~if682/RUP/process/activity/ac_epcmp.htm
- Alistair Cockburn “Just-in-time methodology construction” http://alistair.cockburn.us/Just-in-time+methodology+construction
- Glen B. Alleman “Agile Project Management Methods for IT Projects” http://www.niwotridge.com/PDFs/PM%20Chapter%20(short%20no%20email)%20Update%202.pdf
- Project Management Partner LLP, The Project Management Knowledge Areas, Chapter 4: Project Integration Management, (PMBOK® Guide pg 77-102), http://www.pmp.sg/erc/Day%202/4.%20Project%20Integration%20Management_ERCv1.pdf