The Software Development Process

Change Management in Software Development

CHANGE MANAGEMENT

- Change management is a process depending on the context:
  - **Configuration Management**:
    - It is the discipline to prepare, equip and support the team to receive, analyze and implement changes to the requirements
    - In this context, change management tools for version controlling
    - These tools will prevent more than one person from modifying the artifacts at the same time
    - The tools have capabilities to track whatever changes are being made, to back out changes when necessary, or to allow for multiple paths, so that different versions can be developed simultaneously
    - Tracking changes is critical to quality in software projects
    - By having change control, the team will be able to associate code fixes or changes with defects and automate builds, patches, etc.

CHANGE MANAGEMENT

- **IT Service Management (ITSM)**:
  - In this context, change management processes are highly controlled to track any changes that occur in an IT infrastructure
  - Standardized methods and procedures are used to ensure that every change made to the software is tracked and handled appropriately

WHY REQUIREMENTS CHANGE

Changes happen because:

- A requirement is missing
- A defect has been detected
- Your customer didn't understand their actual need
- Politics
- The marketplace changes (new features, new technologies, competitors)
- Legislation changes

It's common to show a stakeholder your working system to date only to have them realize that what they asked for really isn't what they want after all
This is one reason why active stakeholder participation and short iterations are important to your success.

CHANGE MANAGEMENT

The following is a Change Request process during a software development process/project:

- Change Requests are allowed only during execution phase
- Overall requested change must not exceed 20% of total planned effort
- If so, a new project shall be addressed to cope with this change
- Change request could cause a delay in the Project initial planning
- It could set a freeze period for changes, typically at the final stages in order to secure quality

The bottom line is that if you try to "freeze" the requirements early in the lifecycle you pretty much guarantee that you won't build what your customer actually needs, instead you'll build what they initially thought they wanted

CHANGE MANAGEMENT: PROCESS

Change Initiator

Change Coordinator

Understand the possible implications of implementing the change
Identify all the files and documentation
Identify the tasks and the estimation of the effort

Change request

Analysis

Impact on price?

Yes

Change request offer

Approved by customer ?

No

Discard

Yes

Change Manager

Change Priorization / Planning

No

Change Implementation Team

Implementation

Change Testing Team

Testing

Testing successful ?

No

Rollback

Yes

Update the change management system

Document the change

Change Manager

Change deployment

Change Implementation Team

TOOLS FOR CHANGE REQUEST ANALYSIS

CHECKLIST OF THE POSSIBLE IMPLICATIONS OF A PROPOSED CHANGE

- ☐ Do any existing initial requirements conflict with the proposed change?
- ☐ Do any other pending requirements conflict with the proposed change?
- ☐ What are the business consequences of not making the change?
- ☐ What are the technical consequences of not making the change?
- ☐ What are the side effects, impacts or risks of making the change?
- ☐ Will the proposed change affect performance requirements or other quality parameters?

TOOLS FOR CHANGE REQUEST ANALYSIS

CHECKLIST OF THE POSSIBLE POSSIBLE SOFTWARE COMPONENTS AFFECTED BY A PROPOSED CHANGE

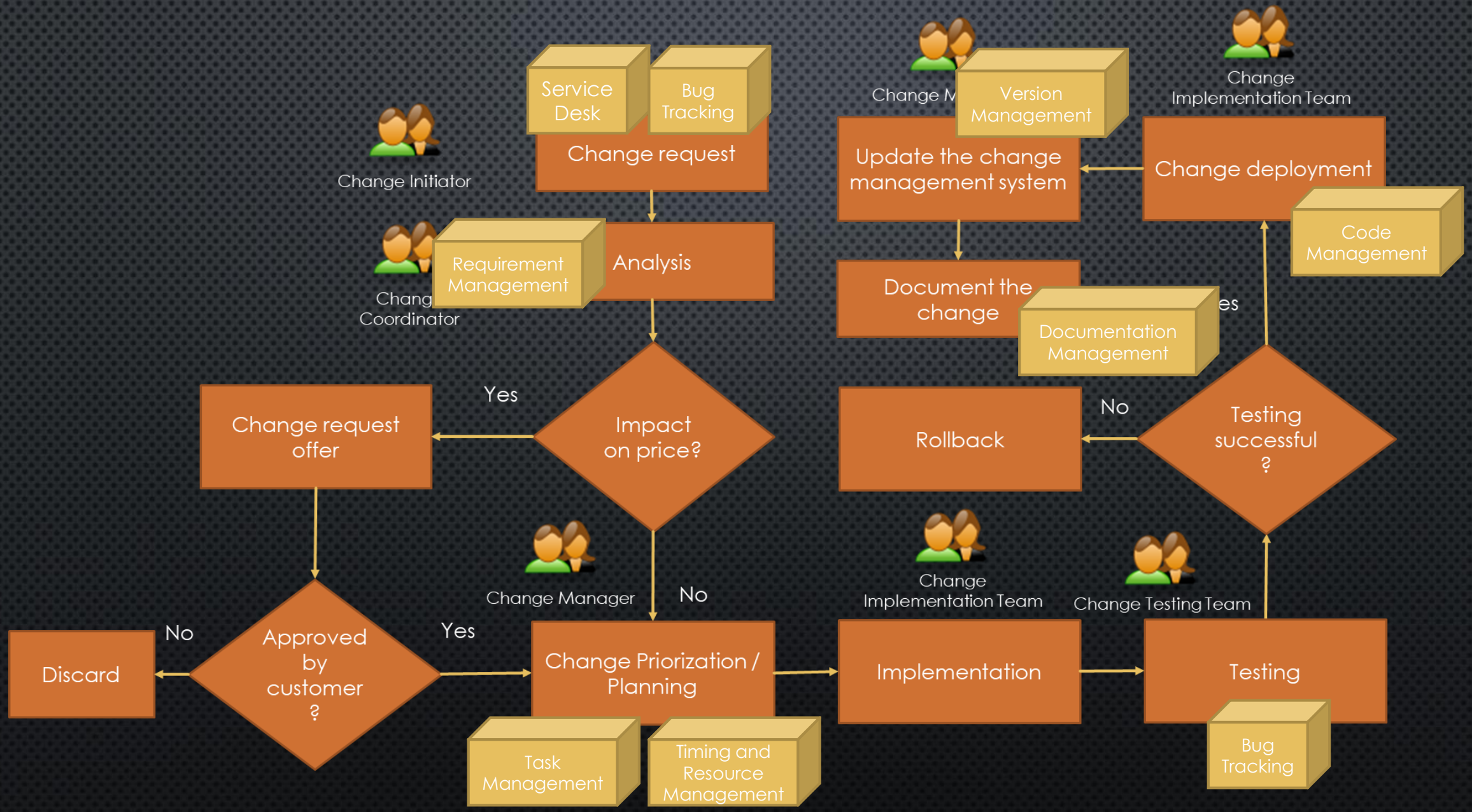
- ☐ Did you identify any user inteface changes, additions or deletions required?
- ☐ Did you identify any changes, additions or deletions required in reports, databases or files?
- ☐ Did you identify the design components that must be created, modified or deleted?
- ☐ Did you identify the cource code files that must be created, modified or deleted?
- ☐ Did you identify any changes required in build files or procedures?
- ☐ Did you identify existing unit, integration, system and acceptance test cases that must be modified or deleted?

TOOLS FOR CHANGE REQUEST ANALYSIS

FORM TO ESTIMATE THE EFFORT FOR A REQUIREMENT CHANGE

Effort [hours]	Task
	Update the requirement database
	Create new design concept
	Modify existing design components
	Develop new user interface components
	Modify existing user interface components
	Develop new user documentation and videos
	Modify existing user documentation and videos
	Develop new source code
	Modify existing source code
	Develop a new interface to third party software
	Modify existing interface to third party software
	Modify build files
	Create new test cases
	Modify existing test cases
	Develop new unit and integration tests
	Perform unit and integration tests

CHANGE MANAGEMENT: TOOLS



CHANGE MANAGEMENT: BEST PRACTICES

- Avoid changes as much as possible
- Include changes in posterior phases
- Involve in early phases to key users

AGILE REQUIREMENTS CHANGE MANAGEMENT

- Following an Agile approach we are already aware that requirements evolve throughout a project
- Because requirements evolve over time detailed documentation at the beginning does not make sense
- The focus is on the identification of the initial requirements and developing a high-level schedule and estimate
- During development each requirement is investigated in a just-in-time manner in the necessary detail

AGILE REQUIREMENTS CHANGE MANAGEMENT

- Because requirements change frequently in an Agile approach you need to implement a requirements change management process throughout your project
- This process has to follow a streamlined, flexible approach
- The goal in an Agile project is to develop software which is both high-quality and high-value
- the easiest way to do this is to create the list of requirements with their priority and implement the highest priority requirements first
- In this approach your challenges are in managing changes and not in preventing them

- The stakeholders are responsible for prioritizing the requirements
- The developers are responsible for estimating
- The priorities of non-requirement work items (*take training, go on vacation, review products of others or documentation, work on defects, doing documentation*) are defined and agreed by the team with the stakeholders

Stakeholders

Developers

High Priority

Requirement #1

Requirement #2

Requirement #3

Requirement #4

Requirement #5

Low Priority

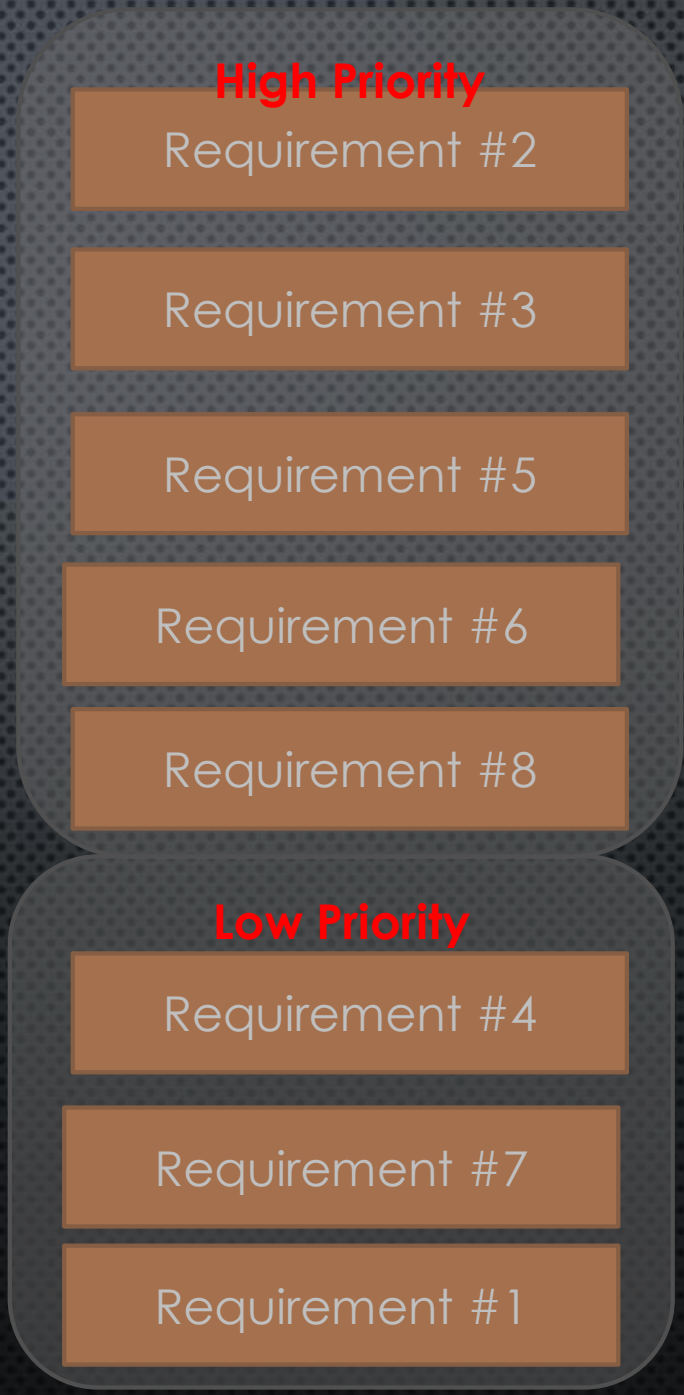
Requirement #6

Requirement #7

Requirement #8

- When a new requirement is specified you need to review the priority and maybe you change your list

Requirement #9



- When a new requirement is specified you need to review the priority and maybe you change your list

Requirement #9

- Scrum suggests that you freeze the requirements for the current iteration to provide a level of stability for the developers
- If you do this then any change to a requirement you're currently implementing should be treated as just another new requirement