Chapter

5

Project Management

CHAPTER AUTHORS

Chen Minchao Daniel

Mohd Shahab

Nguyen Viet Thinh

CONTENTS

1 INTRODUCTION

Many aspects goes into a software development ranging from planning, development to testing. Managing a software development requires knowledge, techniques and tools for a successful development. In this chapter, we will be looking at the various tools used to in managing a software development.

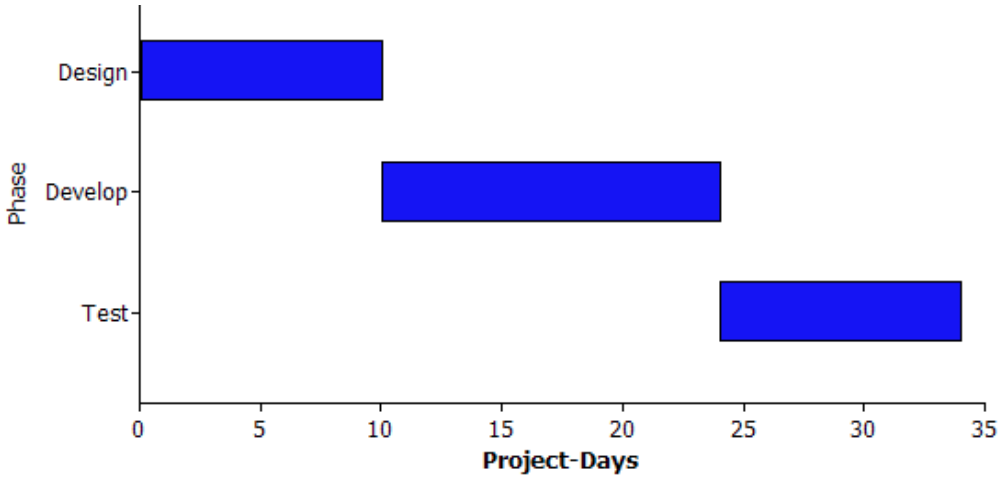
2 TOOLS FOR GENERAL SOFTWARE DEVELOPMENT LIFE CYCLE

Software development life cycle is a structure employed for developing a software product and tools can assist managing these various aspects.

2.1 Tools for Planning

Project planning is a necessary part of project management where the scope of the project is defined and the means of completing the project is determined. These include estimating the resources, time and costs required of the project. Using a tool would help you to define and manage the various aspects in the planning stage of your project.

Timelines are commonly used to define and manage the project schedule and are commonly represented as Gantt charts. The key due dates of milestones are set, deadlines of tasks are outlined and task dependencies are easily identified and resolved through the timeline. This can be seen from the following figure where the task on week 5 is dependent on the tasks before and cannot be started until the dependent tasks are completed.



Example of a project schedule Gantt chart
(Source: http://tinyurl.com/gantt-chart-example**)**

Having defined the project schedule and tasks, the human resource plan is formed next. The plan defines the members of the team, their roles and assigning tasks to individual members on the team. Planning tools such as Target Process and Accunote allows easy assignment and management of tasks to team members.

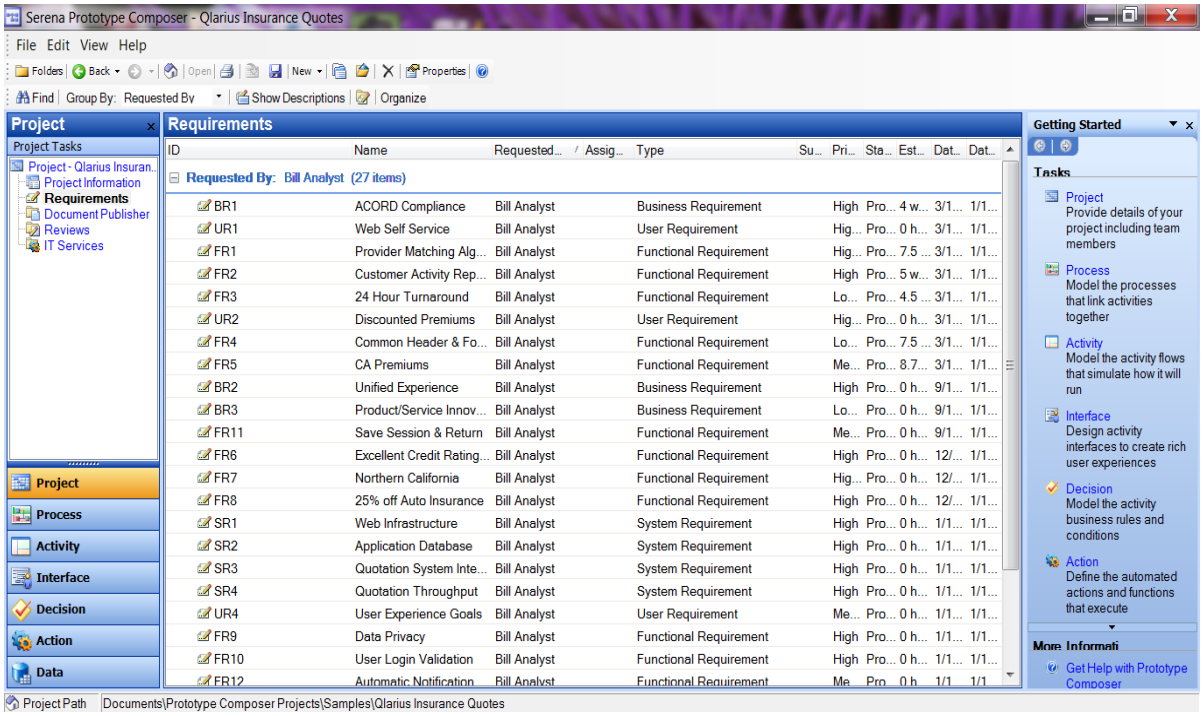
2.2 Tools for Requirements

Requirements gathering is an important part of the Software Development Lifecycle (SDLC). Requirements indicate what a software application needs to do for satisfying the users' needs and wants. Hence, well-defined requirements are crucial to the outcome of a project. Majority of the software failures and defects are blamed on bad requirements. Requirements may be high-level or low-level (very specific). It is a good practice to clarify what level of detail is needed or expected before collecting and recording the requirements.

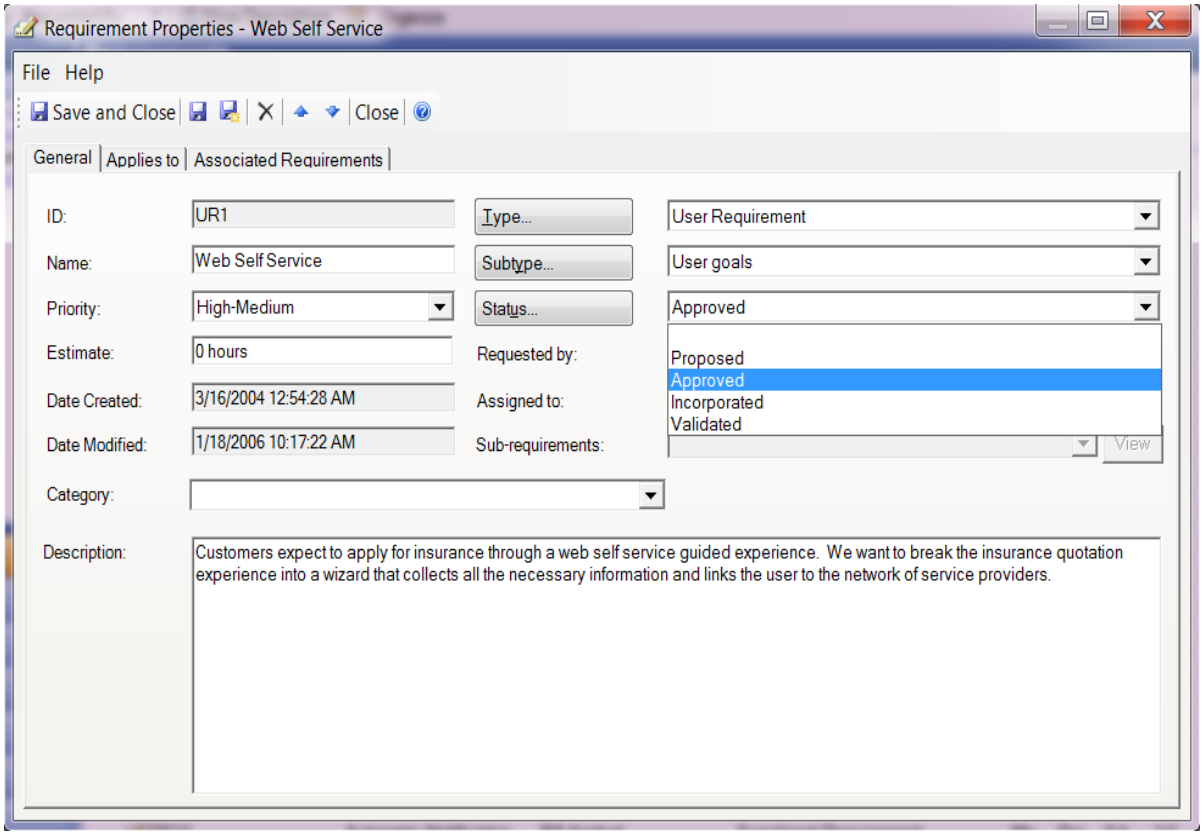
It is important to know that the requirements not only record what features or functionality need to be present in a software application but they can also describe *non-functional requirements* or system characteristics such as:

1. **Performance** – How fast does the system need to be?
2. **Capacity** – How much traffic should the system be able to handle?
3. **Availability** – Should the system be 24/7 or weekdays only etc.?
4. **Portability** – Can the system easily be moved to other platforms?
5. **Data Currency** – Does the system need to be real-time?

There are various Project Management (PM) tools in the market that not only help record and manage the requirements but also model the system and run simulations to find out if there are any flaws which need to be addressed before starting the actual development. Serena (figure below) is one such tool. Serena, and other similar tools, provide functionality to manage requirements and re-use them. When you modify a requirement anywhere in the project, Serena notifies all those who might be affected by the change.



A screenshot from Serena sample project highlighting its requirements capabilities. Anyone can record the requirements and there can be many different kinds of requirements.



Serena, like other similar tools, provides the functionality to record requirements in a categorized manner, making the entire process more efficient

Using these or similar tools for requirements gathering will enable an organization to collect, segregate, prioritize and analyze all the relevant needs for the application to be developed. This understanding of the requirements will result in a more robust, usable and scalable software application.

2.3 Tools for Development

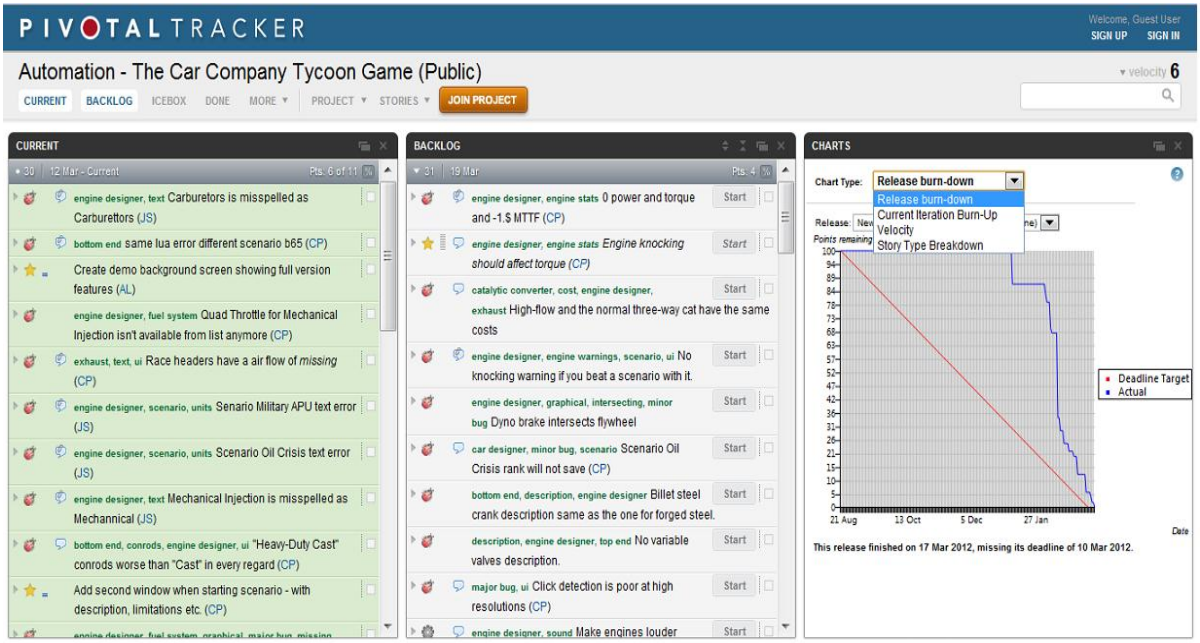
In the Development(Implementation) stage, developers will bring about the product specified by producing the codes based on the requirements gathered at the previous stages. Development stage is the longest stage of the life cycle, and often is the main focus of developers. It may overlap with other stages such as Requirements and Testing stages.

A project manager will need to monitor and control the progress of the development cycle based on the planning he had prepared, directing the evolution of the product. The development team will engage in tasks such as revision control, code review and documentation during the software development.

We will be looking at tools that can provide at least one of the following functionality :

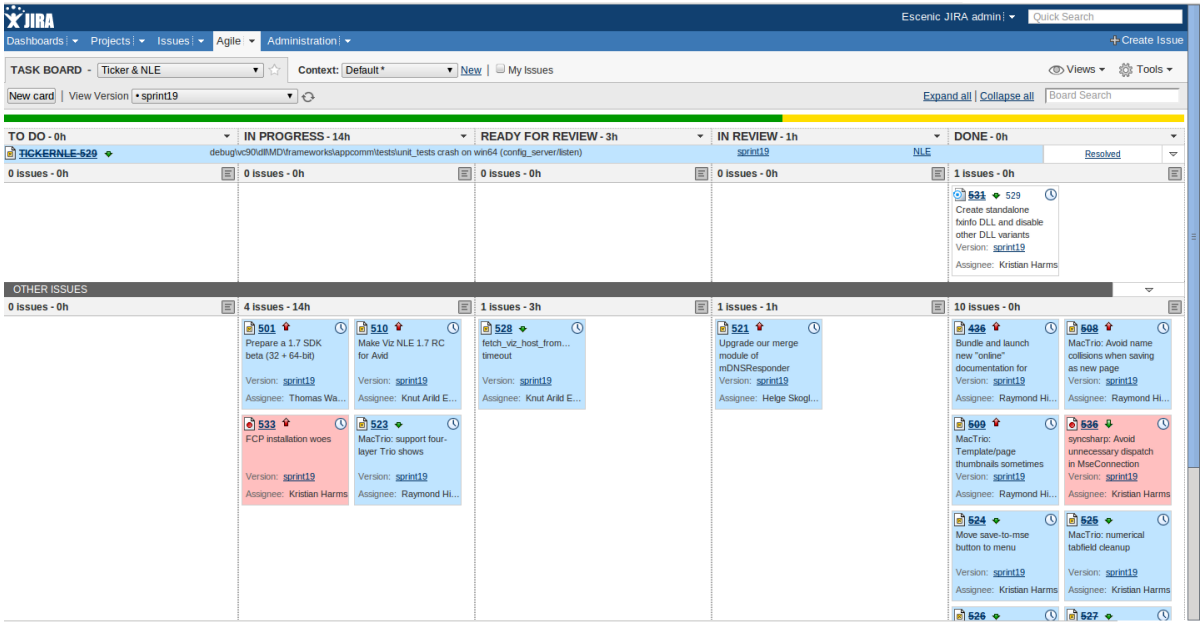
- **Tracking/Monitoring** : (allows project manager) assign tasks to developers, tracking the progress made on the projects. Provides (project manager) various kind of reports on demand.
- **Collaboration** : enhance team-works by offering fast and effective communication medium
- **Documentation** : store,retrieve project related documentations in a easy-to-access and secure repository.

- **Revision Control** : control, audit, organized product artifacts into version-ed components.
- **Code Review** : allows developers to perform various kind of software technical review such as code review, code inspection, etc.



A screenshot of Pivotal Tracker ,showing Current Log(current feature being developed), Backlog (requirement) and A Release Burndown Chart

(Source: https://www.pivotaltracker.com/projects/352253/stories on 18/3/2012)



A screenshot of Jira (a PM tools with multiple functions) with Greenhopper plugin , showing its Task Board, clearly depicting Tasks in various stage (Done-In Progress-To Do), with each Task has information such as user stories(features) associated, priority, effort required, the developer in charge etc.

(Source: https://jira.atlassian.com/browse/GHS-2212 26/3/2012)

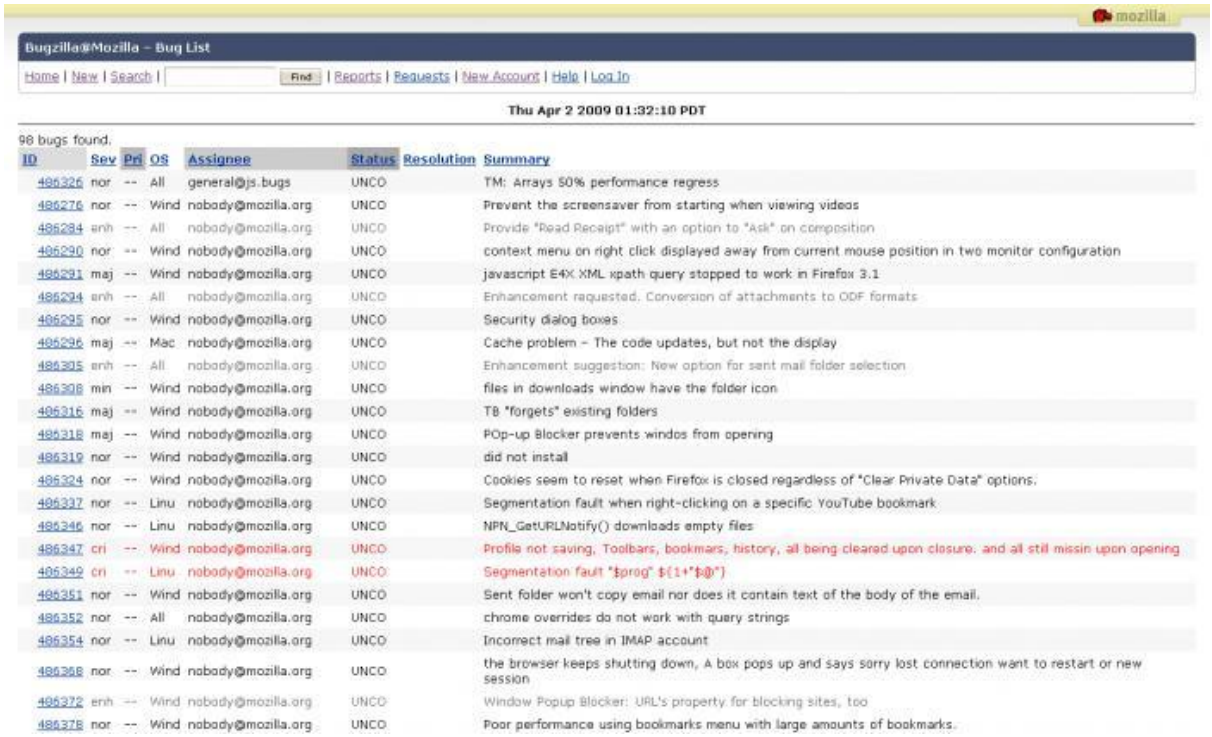
Tool	Tracking and Monitoring	Revision Control	Technical Review	Documentation	Collaboration
Pivotal Tracker	✔	✘	✘	✘	✔
TortoiseSVN	✘	✔	✔	✘	✘
Acunote	✔	✘	✘	✘	✔
GoogleCode	✔	✔	✘	✔	✔

2.4 Tools for Testing

The testing process can begin as early as the requirements gathering phase of SDLC. Developing a test plan, coming up with use test cases and scenarios, executing these test cases, tracking the bugs and resolving these bugs are some of the activities carried out during testing.

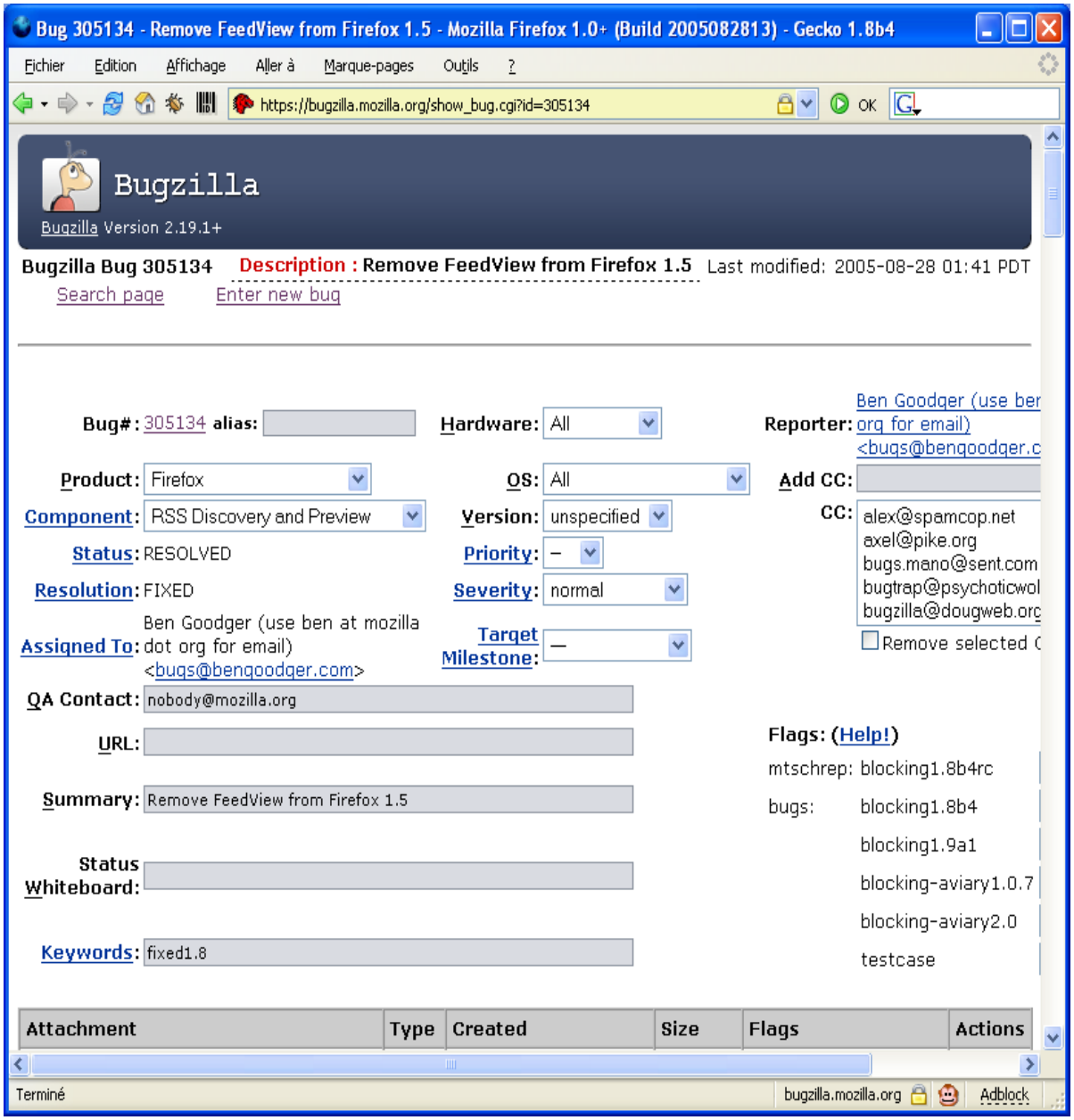
Testing is an extremely important SDLC phase and is supported by a many different PM tools in various ways. One such tool is Jira, which is an easy to use tool for bug tracking and has a powerful advanced search feature. Jira provides its users fast bug tracking, using numerous keyboard shortcuts, and the ability to link issues to the related code hence saving time.

Another good and free tool is Bugzilla (figure below)



Bugzilla offers bug tracking features and also tracks code changes

(Source: http://screenshots.en.sftcdn.net/en/scrn/80000/80773/bugzilla-2.jpg)



Bugzilla provides numerous fields and options for use when recording any bugs/defects. This allows developers to zoom in on the exact issue hence increasing the efficiency.

(Source: http://www.download.ba/bugzilla/)

Using these or similar PM tools for the different activities during the testing phase can help increase productivity, reduce downtime, improve the communication, reduce costs and raise customer satisfaction.

3 A CASE STUDY ON SCRUM

Scrum Process

Scrum is a flavour of agile software development process. It is an iterative and incremental methodology.

Scrum main ideology is : "Doing the job efficiently". Compare to other agile development process (like XP), Scrum focus more on the Project Management aspect.

It is described as "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value " (The Scrum Guide, Ken Schwaber and Jeff Sutherland).

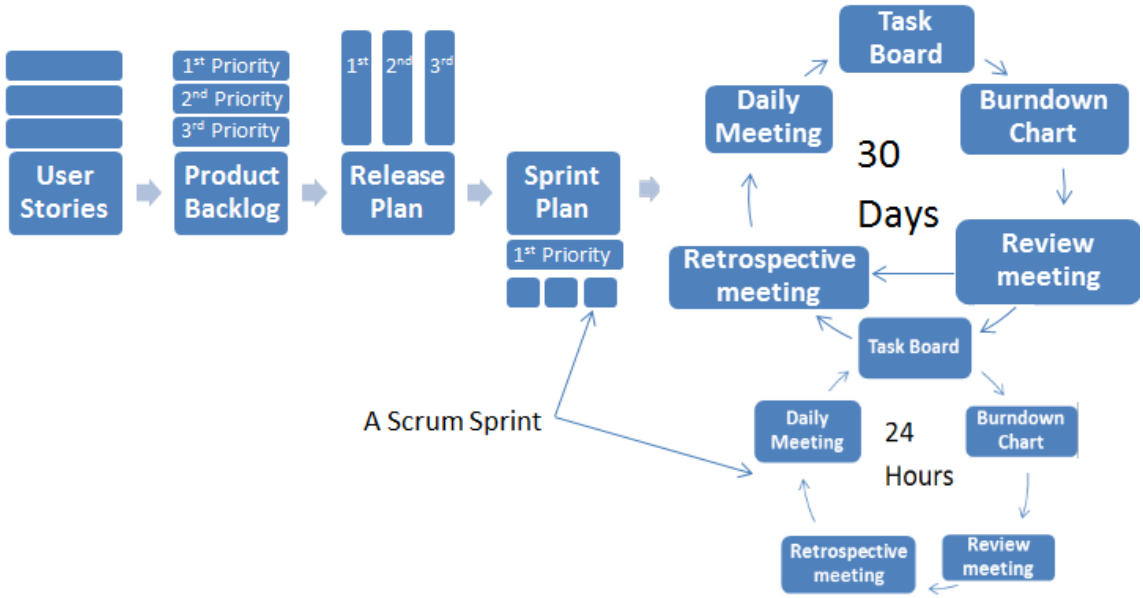
Scrum is one of the most popular processes in the IT industry today and has been successfully adopted by many organizations. The firms and their specific Scrum process used can be found at http://tinyurl.com/firms-using-scrum, which is created and maintained by http://scrumcommunity.pbworks.com.

Roles

The process is governed by 3 main roles :

Product Owner :	The customers or the users of the product.
Scrum Master :	The team leader who facilitates and help team members to use the Scrum framework as intended.
Scrum Team Members :	The developers and the testers

Process Breakdown



User stories are first collected in the scrum process. It is a short description of about the target user, what functions does the user would perform using the mobile application and why the user would use the application. User stories are collected by the product owner and team through inquiry techniques from the target users such as interviews and survey forms. A generic way of describing a user story as outlined by Mike Cohn (2004) is the "As a [role] I can

[function] so that [rationale].” form. The following user story contains specific information that helps the team in designing the product:

“As an administrator, I can assign access rights to users to control information access.”

The product owner and the team identifies, defines and prioritizes the key features and requirements of the mobile application for development from the user stories. These features constitutes the product backlog.

The release plan is formed by grouping items of the product backlog into software releases. Each release is a fully functional application and is worked on in sprints. The developer and scrum master decides what items of the release backlog to work on and how long the sprint will be.

Once the sprint plan is formed, the development starts.The main components of a Sprint is : Daily meetings, Burn Down chart, Task Board, Sprint Review meeting, Retrospective Meeting.

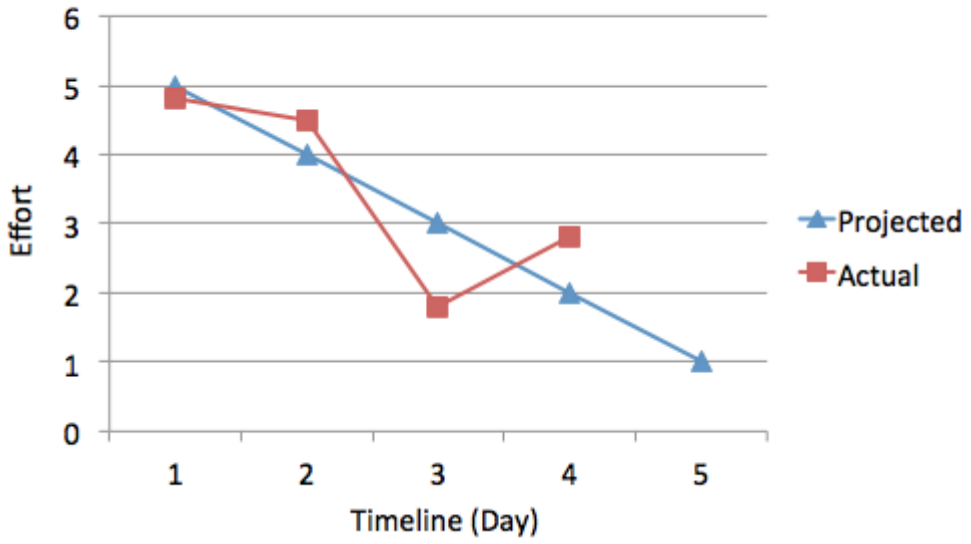
The Daily Meeting : Before each day of a sprint, a team meeting is conducted which is attended by all the people involved in the project including the product owner, the scrum master and team members. The scrum master facilitates the daily meeting. The meeting should ideally be no longer than 15 minutes. During the meeting, team members share what they worked on yesterday, what they will work on today, and what are the current impediments. The daily meetings help synchronize the whole team, keep everyone up to date and identify any impediments as soon as possible. Based on the information gathered from this meeting, the future sprints may need to be updated. Some basic impediments may be taken care of by the scrum master such as “a premium software license is needed”.

The Task Board : Keeping track of To-Do, Done, To-be-Verified(test) backlog item

The Burn Down Chart : a chart showing remaining work effort that has to be completed in the sprint backlog vs the time remaining in the sprint. The y-axis can be the number of stories, number of feature remain. The x-axis can be the sprint number or the day.

The following diagram is an example of a burndown chart. The projected effort line is generated based on the sprint plan and the actual effort line tracks the progress of the team. When the team is going off track, we observe that the actual effort line is above the projected effort line. And if the the teams is ahead of schedule, the actual effort line falls below the project effort line. If the velocity of the actual remain effort follows that of the projected remaining effort, the sprint is on track.

The velocity is a measurement of how much the development team get done during 1 sprint.



Example of a burndown chart

The Sprint Review Meeting: At the end of every sprint, another meeting is called. The purpose is to showcase what had been done in the sprint to the Product Owner and the Scrum Master. Evaluate whether the team met the goal of the sprint.

The Retrospective Meeting: Usually organized right after Review Meeting. This is the opportunity to identify the problem and discuss on how to improve for next sprint. Several metrics can be used to access the team performance here like Burn Down Chart, Velocity, Number of Defects, Final Product etc.

4 CONCLUSION

In this chapter we have looked at how different project management tools can help at different stages of the Software Development Lifecycle. Since some PM tools can be used during multiple stages of SDLC, we have categorized some of the more popular tools which will hopefully help you in deciding which tool is the best for your personal use.

There are some popular processes in use in the industry today, such as Scrum. The case study shows how scrum-specific tools can be used to manage agile processes in an efficient manner.

5 APPENDIX

5.1 General Project Management Tools

Tools Name	Planning	Requirement	Development	Testing
MantisBT	✗	✗	✗	✓
Google Code	✗	✗	✓	✗

Basecamp	✓	✗	Partial	✗
Codendi	✓	✗	Partial	✗
Bugzilla	✗	✗	Partial	✓
Asana	✓	✗	Partial	✗
Subversion	✗	✗	Partial	✗
Veracity	✓	✗	Partial	✓
Pivotal Tracker	✓	✗	Partial	✗
Trac	✓	✗	Partial	✗
Rational RequisitePro	✗	✓	Partial	✗
Acunote	✓	✗	✓	✓
BrightGreen	✓	✓	Partial	✗
Code Collaborator	✗	✗	Partial	✗
Serena Requirements Manager	✓	✓	✗	✗

5.2 Scrum Tools

Tool Name	Website	Free	Scrum Support	Revision Control	Bug Tracking	Others
On-Time	http://www.axosoft.com/ontime	2	yes	Svn, github	Yes(in built)	tortoisesvn plugin, mobile client
Acunote	http://www.acunote.com	5	yes	svn, perforce	code inspection, bugzilla,mantis, trac	Free for open source
Aglio for Trac	http://www.agilofortrac.com	3	yes	svn, git, mercurial, etc	Yes. (Trac)	Plugin for Trac
Agile Bench	http://agilebench.com/	3	yes		inbuilt (as stories)	

Planbox	http://www.planbox.com	5	yes	git, github, svn	MyBugDigger - commercial - limited free features	
Redmine	http://www.redmine.org/		yes	SVN, CVS, Git, Mercurial, Bazaar and Darcs	built-in	Open source (web based app)
VersionOne	http://www.versionone.com	10	yes	no	built-in	acceptance testing
Kunagi	http://kunagi.org		yes	no	built-in	open source (web-based)
Leankit Kanban	http://www.leankitkanban.com/	5	yes	no	built-in	
TinyPm	http://www.tinypm.com	5	yes	svn,git,github, mercurial	jira	wiki,email,rss, pop3,NOT HOSTED
TargetProcess	http://www.targetprocess.com	5 on-site on Window Server	yes	svn, git	bugzilla ,jira or built-in	hosted or non-hosted VS and Ellipse integration

6 REFERENCES

Cohn, M. (2004). *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional.

Kniberg, H. (2007). *Scrum and XP from the Trenches.* C4Media.

Schwaber, K., & Sutherland, J. (2011 йил 10). The Definitive Guide to Scrum: The Rules of the Game .

7 CHAPTER RESOURCES

Scrum Process - http://www.scrum.org/scrumguides/

How to use a task board - http://www.mountaingoatsoftware.com/scrum/task-boards

Applying User Stories - Cohn, M. (2004). *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional.