

Test Management is :

- **a cooperation process of expertise areas, tools, metrics, and hopefully, well applied good sense (not common sense).**
- **in place to help the project to fulfil customers expectations (QA). Prepare and execute Test Cases are only part of the work.**
- **to guide people to understand the process, and then, use it.**
- **an information-providing service**

The intention of this presentation is to show how a combination of tools (mostly open source) can be used to implement a Test Management.

This work I have been developing for some time and have the chance to learn better, improve and implement at my last assignment.

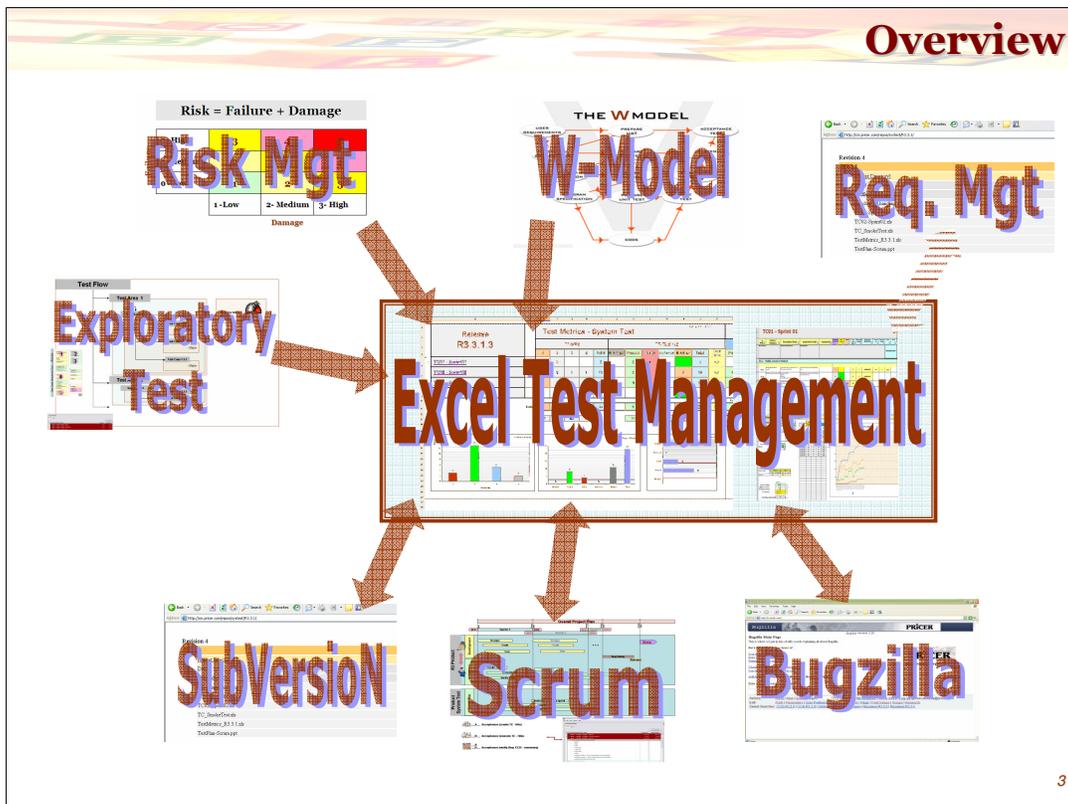
The ideas here presented are also possible to be transferred to many other types of Test processes.

I do not believe to choose Agile or a more structured process (like RUP, Props, etc) are the main problem for a proper SW development.

To me, the main issues are how the plan, execution, control, and most of all, Team cooperation is done.

I hope this presentation can have an impact in the way we see and perform our work.

We must enable our creativity and not be slaves of processes and tools.



The idea is to control Test using Excel templates and supply the other needed parts via additional tools.

In my implementation I have used the tools and processes show in the slide, but of course we can change and adapt according to needs.

This is not an automatic process and need constant intervention, but is also functional, low cost and 'reliable'.

The main point is this implementation is to make sure that all participants in the activities have a good sense of the process and how they are expected to perform and act.

The Seven Basic Principles of the Context-Driven School

1. The value of any practice depends on its context.
2. There are good practices in context, but there are no best practices.
3. People, working together, are the most important part of any project's context.
4. Projects unfold over time in ways that are often not predictable.
5. The product is a solution. If the problem isn't solved, the product doesn't work.
6. Good practice (ex: software testing) is a challenging intellectual process.
7. Only through judgment and skill, exercised cooperatively throughout the entire project, are we able to do the right things at the right times to effectively test our products.

The Seven Basic Principles of the Context-Driven School: <http://www.context-driven-testing.com/>

Context Driven Testing <http://c2.com/cgi/wiki?ContextDrivenTesting>

4

Some ideas

I prefer to have point 3 as my first. Everything else is dependent of it.

The school has a focus on testing, but all the assertions can be used for any development (SW,HW,FW)

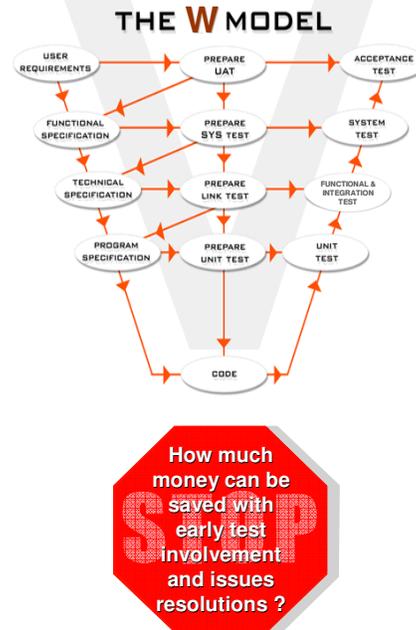
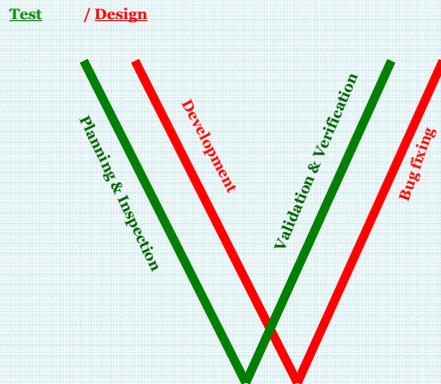
In this case I would change point 6 to: Good **practice (ex: software testing)** is a challenging intellectual process.

More Principles

- * It is not a testers' job to demand documentation. Good testers work with the information they have, use unofficial documents, and are specific when requesting additional information.
- * Testing is an information-providing service, not a "quality assurance" function.
- * The value of testing is determined by whether it provides useful and timely information (about the quality of the software).
- * There are no guarantees. We don't hide behind the test plan. We learn how to improve testing as we test.
- * A tester is a customer advocate. Testers try to understand the customer position and make the best case when they feel it isn't being address.

The W model

The W-Model demonstrates the complexity of relationships between each stage of the development life cycle and acknowledges that for every stage within the development there is an associated stage of testing.



Strengthening the Bond Between Development and Test http://www.stickyminds.com/s.asp?F=S3572_ART_2 5

Test should not start after development delivers the code, it must start as soon the project is decided.

During the development phase, test activities will concentrate to make sure that the product can be testable and does have a solid base for delivery (internally and to customers)

The W-Model demonstrates the complexity of relationships between each stage of the development life cycle and acknowledges that for every stage within the development there is an associated stage of testing.

It shows that testing does not have to occur once the 'code' has been delivered which is what you need to have to begin test execution. The testing can start early with analysing the requirements and creating test criteria of 'What' you need to test.

Faults found earliest in the process are least costly to correct, generally under 20% of the cost of correcting the same error post implementation. Therefore there is significant financial benefit from monitoring and managing testing to identify and perform corrections at the least costly opportunity.

The objective is to ensure every element of the system is validated at the earliest possible stage, to the quality criteria set out by the business managers, providing a comprehensible and manageable audit trail of the systems actual capabilities.

Wmodel article

<http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectID=3572>

Risk Assessment and Calculation

RISK is defined as the chance of a failure occurring related to the damage expected when this failure occurs.

$$\text{Risk} = \text{Failure} + \text{Damage}$$

Failure	0- High	3	2	1
	1- Medium	4	3	2
	2 -Low	5	4	3
		3 -Low	2- Medium	1- High
		Damage		

Definitions

Error	a mistake, something done or gone wrong
Fault or Defect	The result of an error
Failure	when action is executed, produces an error, visible as defect.
Damage	How much it will cost (in money, time, resources, reputation, etc)

Based on 'SW testing A guide to the TMAP approach' pag135

6

Risk is a question of Money (or: Take Risks < Take Money)

Basically , how much money, in sales, time, marked share and others we are willing to risk to loose or how much we are trying to gain.

This table gives a simple number to access the risk we are willing to take. The values will decide what we do first or set our efforts to mitigate.

The numbers and formula are my interpretation, you can use yours, but keep in mind that higher risks must be mitigated first.

Note that we do not evaluate only risks related to development or test, but also conditions that can affect the whole project, like: resources, stolen goods, power breakdowns, resignations, learning and competence, etc.

I know that in some companies they do not care about money (or risks) , but I still never worked for one of those.

Exploratory Test Process

Discover the Requirements

- Gather data:
 - Read documentation (SRS, SDS, Marketing material, etc)
- Examine the product Architecture
- Ask Project Manager, Developers
- Run old product/releases
- Check fixed bugs for the previous release
- Check open bugs for the next release

Create TEST FLOW divided by AREAS

- Divide the project by areas of functionality

Create TEST SUB-AREAS and TEST CASES

- Write a title for each *Test Sub-Area*
- Transform Requirements found in *Test Cases*
- Distribute the *Test Cases* by the *Test Sub-Areas*
(set the *Test Cases* in a chronological order for testing)

Define what quality means to the project

- This will help to establish how hard you have to test
 - Minimize time to market
 - Maximize customer satisfaction
 - Minimize the number of defects
- Set priorities and criteria for the tests

The plainest definition of exploratory testing is test design and test execution at the same time.

This is the opposite of scripted testing (predefined test procedures, whether manual or automated).

Exploratory tests, unlike scripted tests, are not defined in advance and carried out precisely according to plan.

But even a free-form exploratory test session will involve tacit constraints or mandates about what parts of the product to test, or what strategies to use.

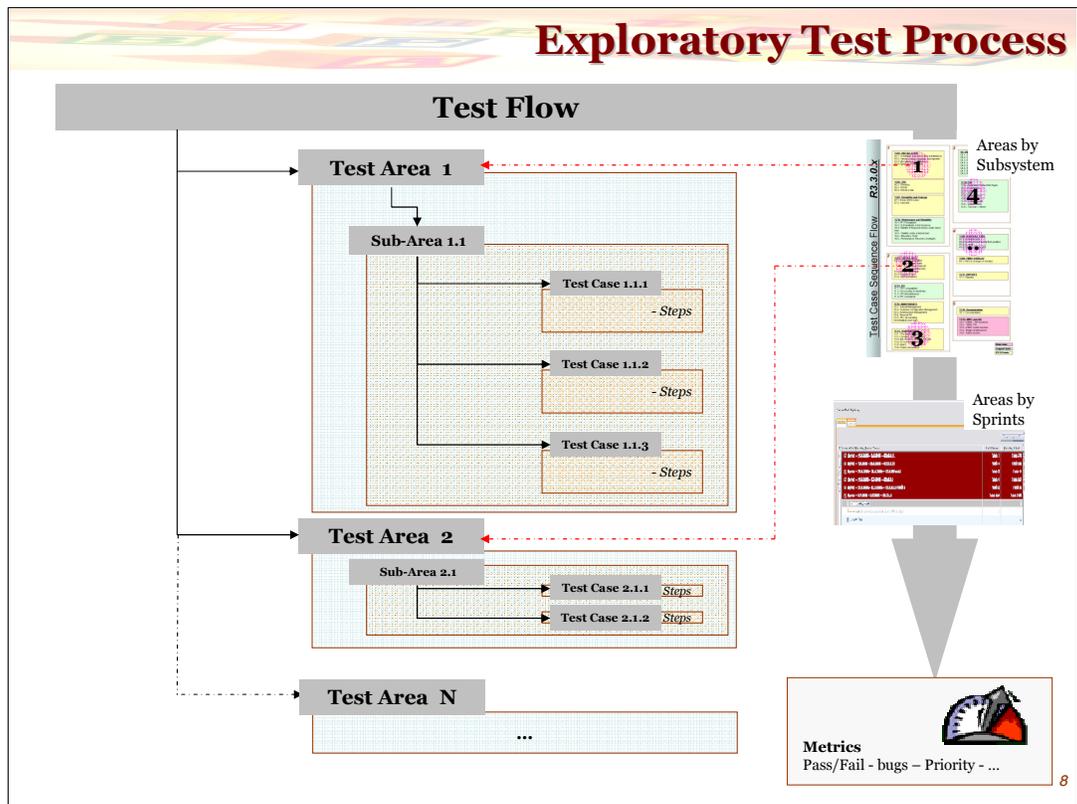
See more:
<http://www.satisfice.com/articles.shtml>

7

This is the first “tool”.

The test team must ‘think out of the box’, but make sure we have a structure to collect these thoughts.

Exploration is very sensitive to competence, knowledge, time, etc – make sure to have well trained and motivated testers.



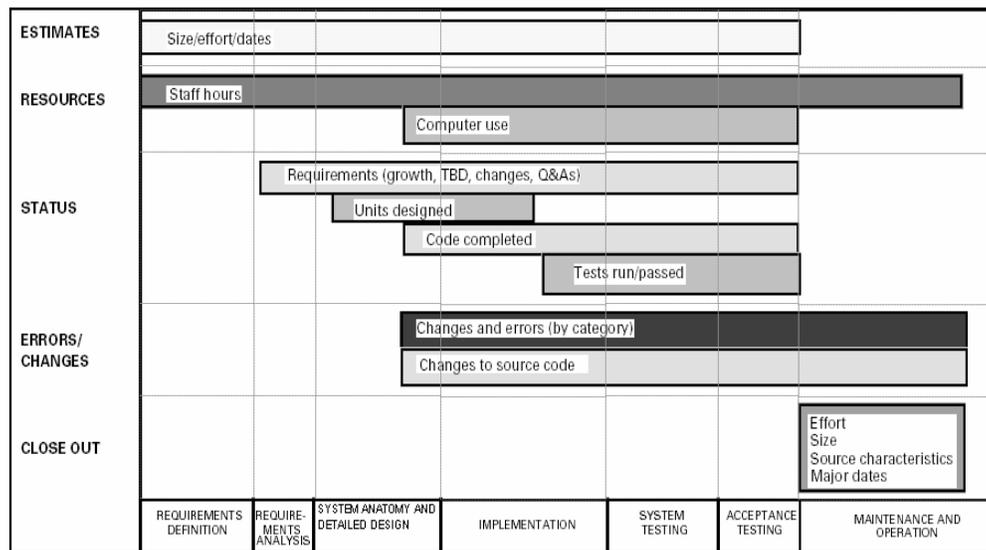
Exploratory structure overview

The Areas can be structured using many approaches, in this example I show by Subsystem and by Sprints

I prefer the approach via areas of development. Pick yours based on best placement for test cases and traceability.

There is no point in testing if we cannot prove (show) what was tested and covered, thus Metrics are a very important part.

Metrics in Lifecycles



Types of metrics that are possible to provide versus time of availability

9

Metrics are only valuable if they are useful , be careful about it.

Only choose and use the ones that are going to bring value to the project, and skip the ones that are 'only nice' or time consuming.

Configuration Management

The screenshot shows the Subversion project website on the left and the TortoiseSVN interface on the right. The website header features the Subversion logo and the text: "The goal of the Subversion project is to build a **version control system** that is a compelling replacement for CVS in the open source community. The software is released under an [Apache/BSD-style](#) open source license." Below this, a "Baseline" label points to "Revision 4 /R3.3.1" in a browser window showing a file list. The TortoiseSVN interface shows a file explorer with a context menu open over a folder named "R3.3.0". The menu includes options like "Open", "Update", "SVN Commit...", "TortoiseSVN", "Add to archive...", "Add to 'R3.3.1.rar'", "Compress and email...", "Compress to 'R3.3.1.rar' and email", "Scan for Viruses...", "Send To", "Cut", "Copy", "Create Shortcut", "Delete", "Rename", and "Properties".

Subversion: <http://subversion.tigris.org/> 10

If there is no CM, there is no control of the deliverables and soon the whole project will be lost.
Emails are not a good place to store development info
Have a plan for Baselines and how to use it.

Subversion is a good tool to use, but any other can do the trick (VSS, Clearcase, CVS etc)

Some good practices with SVN:

- Encrypt the c:/work dir, as SVN copies all the files to local disk.
- Only create local directories that you need
- **Update SVN always before start to work**
- Always remember to upload files, locally stored are not visible to others and are not under CM
- Set properties to read only (need get key) for files that are important and can not be changed without authorization

Create a bat to automatically update the SVN every day and time needed:

```
rem UPDATED          REV: 061016
echo off
at /d /yes
at 09:00 /every:1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 "C:\Documents and Settings\gerson. Almeida\My
Documents\ToolBox_Test\SVN_upd.bat"
REM ---- Change address to your SVN work area -----
set SVN_ROOT=C:\Work
echo Will update SVN to Latest Revision
cd C:\Program Files\TortoiseSVN\bin\
echo CHECK-OUT
mkdir "%SVN_ROOT%\systest"
IF %ERRORLEVEL% EQU 1 goto onlyupd
start /wait TortoiseProc.exe /command:checkout /path:"%SVN_ROOT%\systest" /url:"http://cmsvn.com/repos/systest" /notempfile /closeonend:1
:onlyupd
echo UPDATES
start /wait TortoiseProc.exe /command:update /path:"%SVN_ROOT%\Departments\R&D\SystemTest" /notempfile /closeonend:1
```

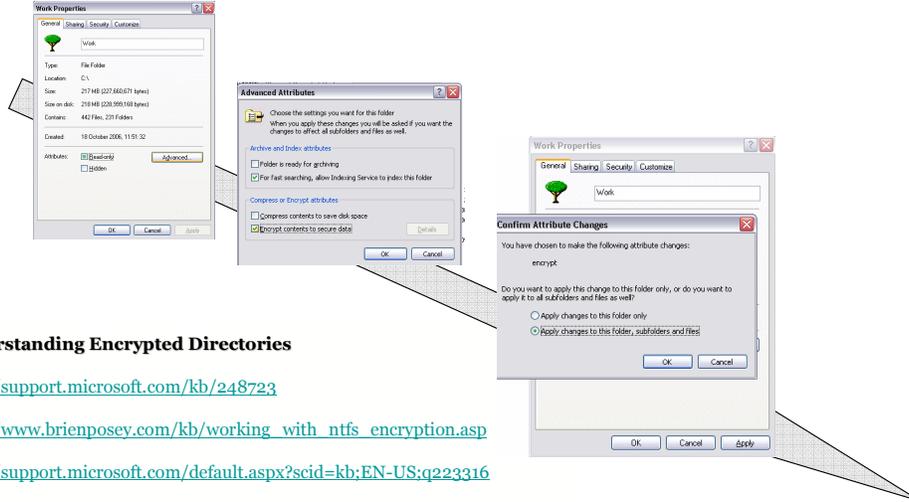
Configuration Management

Tip

SVN repository in the user's PC can be a security risk as it contains all the information and is accessible. If the PC is stolen, all the files are open. One way to mitigate the risk is to make the repository encrypted.



Select encrypt contents to secure data



Understanding Encrypted Directories

<http://support.microsoft.com/kb/248723>

http://www.brienposey.com/kb/working_with_ntfs_encryption.asp

<http://support.microsoft.com/default.aspx?scid=kb;EN-US;q223316>

Trouble Report Handling

Bugzilla @

Introduction

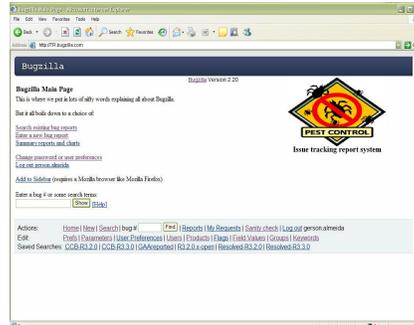
- ❖ Bugzilla is an issue reporting system (open source)
- ❖ It is just a database with a web administration interface.
- ❖ Notifications are sent from the system by email.

Life cycle of an issue

- ❖ Anyone may report an issue.
- ❖ The CCB will analyze the issue and decide on the course of action.
- ❖ The one who changes state is strongly encouraged to write a short comment in the comment field.
- ❖ Bugzilla has a short guideline on how to write a bug report

Where

Bugzilla can be found at:
<http://www.bugzilla.org/>



12

A bug reporting system is mandatory, otherwise we can not trace issues in the development

Note that not always an issue is a bug, sometimes a good suggestion is done and we should trace it until it becomes part of the requirement and/or project.

We use to call it enhancements

It could be better to call 'Issue Reporting system', to help people that get confused when and enhancement is treated as a bug.

Test Management

Spreadsheet

The Test Management metrics are divided in 3 areas

- 1- Test Cases
- 2- Priorities
- 3- Iterations

Every Test Area has its own Test sheet

1

Release R3.3.1.3		Test Metrics - System Test											
		Priority				TC Statistic							
		1	2	3	4	Total	Not Appl.	Passed	Failed	Deferred	NotDone	Total	Exec. Plan
TC01 - Sprint01		3				3		2	1			3	6.3
TC02 - Sprint02		2	4	3	1	10		2	1		7	10	6.3
TC03 - Sprint03		1	6	2	1	10		4	2		4	10	6.5
Total		3	13	5	2	23		6	4		11	23	2.2

2

R3.3.1.3		Priority Status															
		Prio 1				Prio 2				Prio 3				Prio 4			
		Pass	Fail	Not Done	Done	Pass	Fail	Not Done	Done	Pass	Fail	Not Done	Done	Pass	Fail	Not Done	Done
TC01					OK	2	1			47	7						OK
TC02		1	1	50	2	1	3	25	7	1	2					1	7
TC03		1		100	OK	3	1	2	56	7						1	7
Total		2	1	3	3	6	2	5	15	2	3	2	2	1	1	2	2

3

R3.3.1.3		Iteration Metrics									
		Iterations									
Regression		1	2	3	4	5	6	7	8	9	Total
TC01		3									3
TC02		3									3
TC03		3									3
Total		9									9

14

Here is the Excel structure for controlling the test

3 areas are to note : the overall status (1) the status by priority (2) and the TCs by iterations.

These are the needs I had during my projects, but it is easy to create new metrics.

The main point is to be clear of what info are really necessary and do not oversize it with not very important info.

Test Management

Test Case - Template

TC01 - Sprint 01

Area →

Sub-Area →

Test Case →

Priority value

1	2	3	4	5
Must	Must	Should	Could	Wish

Test Status Rules

If the test is **not** performed yet, set to **. (dot)**, this is the default value

If the test **Passed**, set to **P**

If the test is **Faulty**, you have found a **BUG**, set to **F**, write a bug report, and add the bug number.

or

If the test is not possible to be performed because a Bug, set the box to **F** and add the bug number.

If a test is **not applicable**, set the priority to **. (dot)**, the pass/fail box to **NA** and write the reason in the **comments**

If the Test case is **Deferred** (not for the project, postponed, etc), set the priority to **. (dot)**, set the pass/fail box to **D**, and write the reason in the **comments**.

Bug number and link

Remember to link to Bugzilla (ex: bug 123)
http://tr.bugzilla/show_bug.cgi?id=123
 macro: **Ctrl+Q** to link to Bugzilla

When a bug is **RESOLVED** and **VERIFIED** change the bug number to : **V123** and set the TC to **P**

In the Bug Report (<http://tr.bugzilla/>)
 Write:
 - Test Case number : Area.subarea (ex: **TC05.3**)
 - Test Environment: OS/ HW/DB
 (ex. **WinXP/TE11c/Mysql**)

Exec. Time:
 Write the expected amount of time to perform the test (in minutes)

Requir:
 Write the Requirement number.
 Make sure to have a link to the SRS
 macro: **Ctrl+W** to link to SVN

Iteration:
 Write the Iteration number were the test can be executed first time

Updated by:
 Include your Initial and date when you write/update the Test Case

Tested by
 Tester's id (AAA)

Tested in Build
 Build number used for testing = Iteration (ex: 2)

Doc1.2
 make links to external documents (reference to scripts, docs, automated tests, etc).

6

This is how we fill the cells

One important point to be very clear with the testers is to follow this guide, as there is no build in check for mistakes.

We have only used Prio 1-4, as we thought that it was enough.

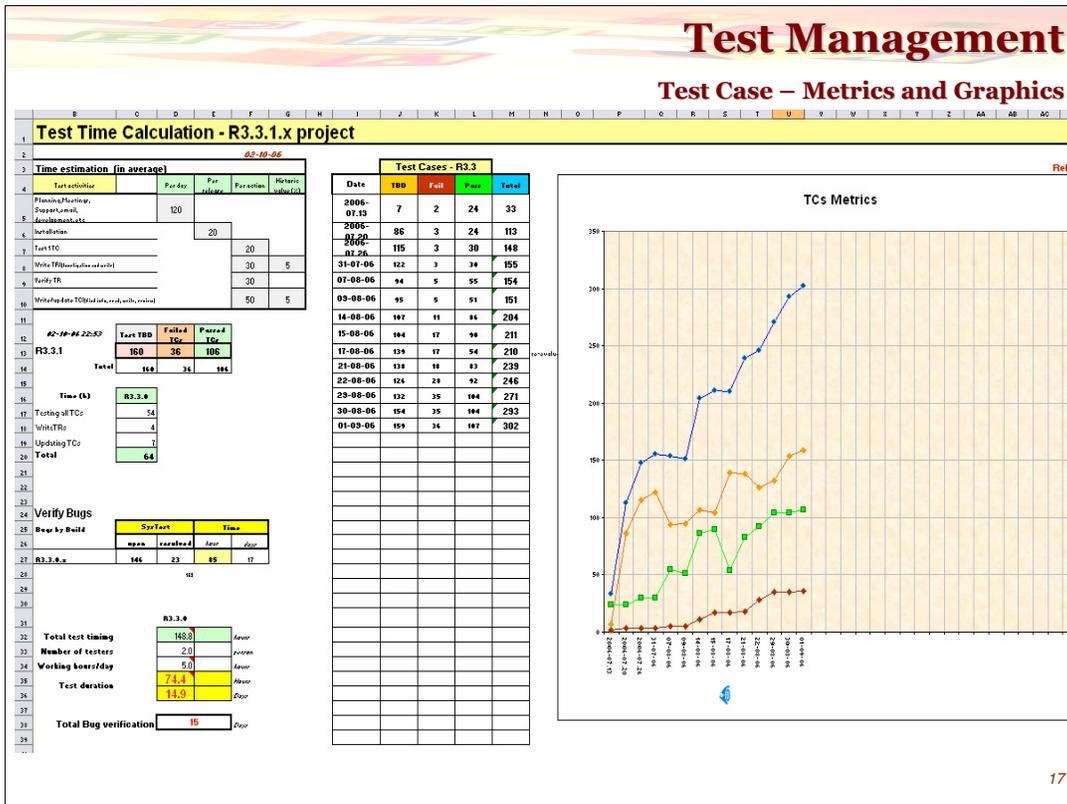
- Prio 1 was regarded as Smoke Test Cases = need to be run every new delivery
- Prio 2 to be run once during the project and repeated every major release or before Beta testing.
- Prio 3,4 were to be run once during the project

A Macro was created to automatically make the link to Bugzilla

Docs are external documents, scripts, tables, etc.

Test Management

Test Case – Metrics and Graphics



Metrics for the daily progress are manually collected and stored in the table

The calculation for the remaining time was very empirical and followed the gut feeling of the test team.

This should be re-evaluated on every project.

Some Tricks and Tips for Excel

- Calculations
- Cell's Colour
- Link & Updates
- Macros to Bugs (in Bugzilla)
- Time estimations

This need to show using excel template

Tasks and Issues:

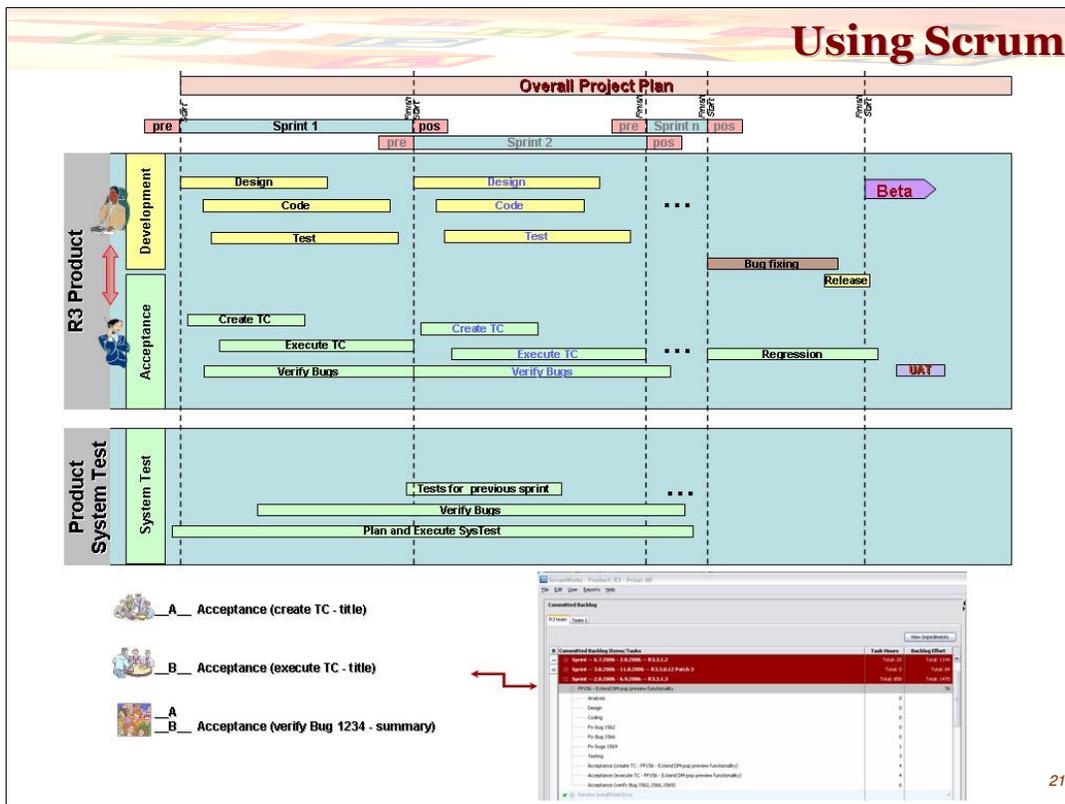
- Set priority and remove impediments
- Update tasks continuously
- Divide tasks into manageable size that will allow accurate time planning
- Define all supporting tasks in the beginning of the iteration
- Requirements handling must be done properly

20

What to do :

- Daily meetings – developers and testers
- Tests in the Sprint (and for the sprint)
- Added/removed tasks
- Changed priority
- tests that are not related to the sprint
- Tests not possible to perform or finish before the end of the sprint (?)
- Bugs found during the sprint
- Outside Bugs and needed fix introduced in the sprint
- Minimal information and CM
- Outsourcing
- Handle teams spread in different locations

Using Scrum



Only the item described in the sprint are to be tested at acceptance and should be done as soon they are released (Task hours in development = 0)

Tests are to be performed on every selected item of the sprint, and must be done during the sprint time.

Tests have 2 parts: Create TC & Execute TC

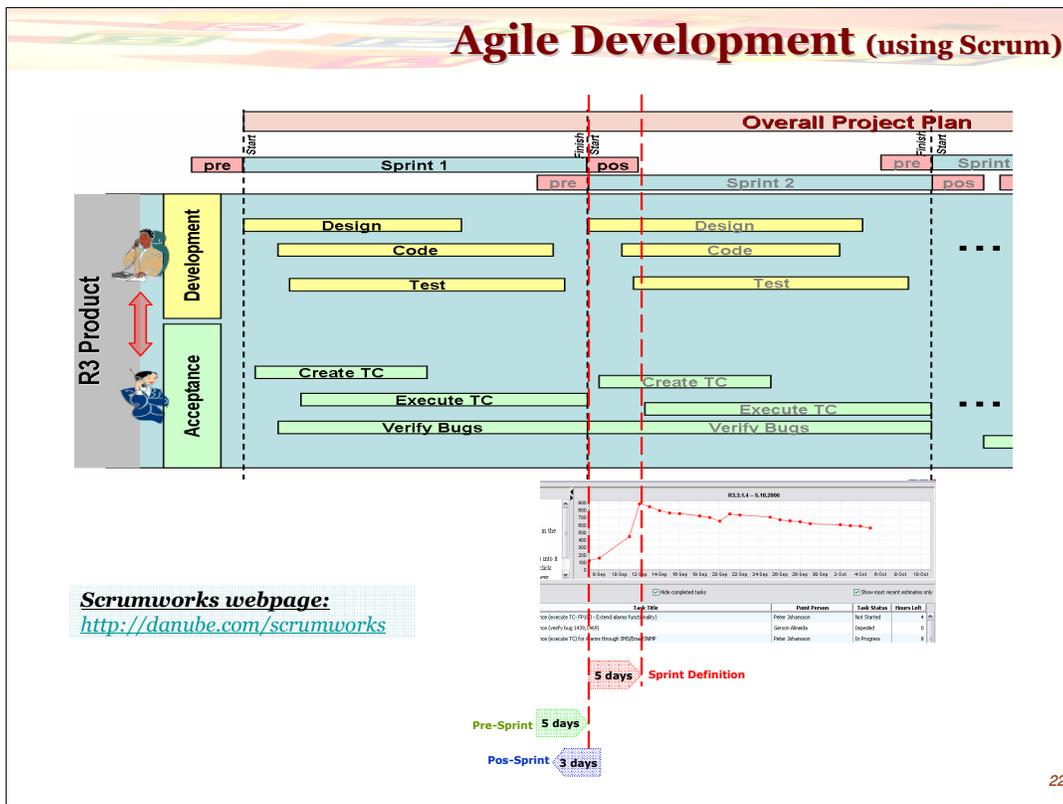
Bugs are to be verified as soon they are resolved (in Bugzilla)

SysTest sprint are tests to be done outside the development sprint. They can be:

- TCs that could not be done during the sprint
- Regression tests
- Performance/Load/Stress Tests
- Environment Tests and preparation
- Bug verification

The person appointed for a sprint should work full time on that. Communication between the person in the R3 product and System Test product is vital to synchronize both works.

Agile Development (using Scrum)



Consider :

- Since at the beginning of a process everyone is involved, but at the end there are people who are finished work and are free to start on the next sprint, it seems to me a waste of time to follow the standard approach.
- Team is concentrated in the backlog in hand and has no time or process to focus in the overall picture. a holistic approach needs to be in place to avoid the creation of a spaghetti system.

Proposal :

- Sprint 1 is to define (or evaluate) the Overall Project plan – all the implementations, needs, fixes, etc are to be defined; of course the plan is a live creature and will *change during the process and subsequent Sprints need to handle this fact.*
- The backlog for sprint 2 is created and plan/definitions of CM, Test, Req, etc are done.
- Priority is set
- Sprint 2 starts as normal
- No new task is allowed during the Sprint, *If an insertion is to be done then something needs to be removed (otherwise time is compromised)*
- Close to the end of sprint 2 , some people are assigned to investigate and prepare the backlog for Sprint 3 (pre-sprint), the backlog is proposed.
- Reviewed and accepted of next backlog done on the first hours of Next Sprint.
- The post-Sprint is necessary to clean up leftovers and propose solutions for those.
- There should be 1 or 2 Sprints in the end to focus only on bug fixing

How to handle CCB in Scrum ?

Questions to answer:

- Should we have CCB at all?
- Time / Periodicity
- Separate CCB or inside Daily Scrum?
- Participants
- Priority & Severity
- Bugs found during Sprint are to be fixed when?
- Need a formal reporting?
- If no formal reporting is done, need traceability?
- Relevance and connection to Requirements

23

Issues during Sprint

During the sprint, some issues can be raised that does not need to become a bug, but must be fixed before the end of the sprint.

The solution can be to include the issue in the backlog "Issues during Sprint"

- The task should contain the necessary information about the problem and, if possible, suggestions.
- If the task is discovered to belong to an already existing backlog, it will be moved by the management.
- Management will set the priority for the tasks and appoint people, but we all are responsible to at least read and see that the issue is not related to your current work.
- The task hours should be decided ASAP.
- Everybody that find an issue should report it in this backlog (if applicable), for analyses, at the same time an email should be send to the person responsible (cc: to R3DEV)
- If the issue is not resolved or should not be resolved during the sprint (for any reason) a Bug must be entered in Bugzilla

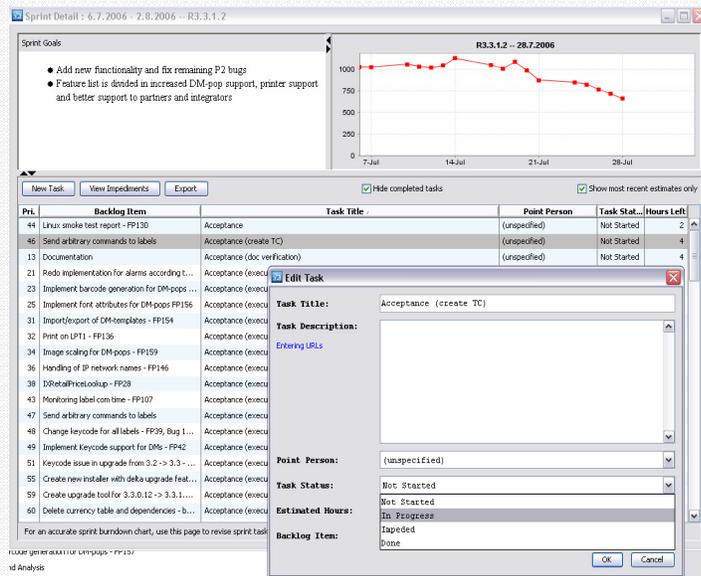
Suggestion on how to handle issues during implementation (sprint):

- Write an email and add screen shots, commentaries, etc about the issue (ex: "[Preview functionality issues](#)")
- Send to the task responsible in Scrum: Sprint ; CC: to scrum members
- Write test (bug or issue) as a TC (example TC03.2)
- Update the task in the Sprint with the reference to the email and TCs
- If the faulty is from a closed sprint or the issue is to be postponed , submit it as a bug

What Why Where How

Connection via:

- Excel
- SVN
- Scrumworks
- Bugzilla



24

Start ScrumWorks

- Choose the top task to perform from the **Sprint Detail**

Look for Tasks = **not started**

- If the task is = **In progress**, is because someone has reserved it and most likely is doing some work just now

- Reserve the task (**task status = In progress**)

- Add your name (**point person**)

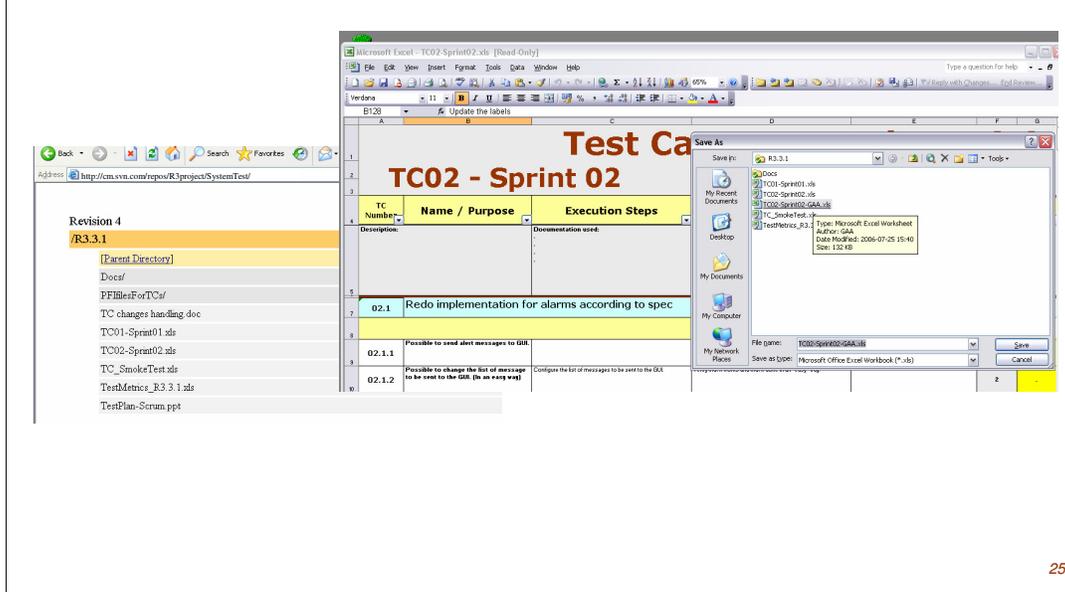
- Write comments, if needed (at any time during the work)

This is a good practice, to keep some history about the work add you userid and date in the comment

- Verify if necessary info is available In the SDS documentation

Testing in Scrum

Save TC_master to yours own private copy



TC_Master is locked, while the file is locked, nobody can update it or get the lock. If the lock is not obtained you are not allowed to save the file, most likely because someone is updating the file and has locked it.

Make a copy of the TC_Master (to a private copy)

When changes are done, copy-and-past the changes from your copy to TC_Master (remember to get the lock first)

Save TC_master

Commit the changes in SVN

Update ScrumWorks tasks

When committing files, the locked files will be released and will be available again for the next user

Some helpful information

Seek information and ideas at forums (example)

Stickminds (<http://www.stickyminds.com/index.asp>)

SQAforums (<http://www.qaforums.com/>)

Read and evaluate Test using TPI

http://www.sogeti.nl/Home/Expertise/Testen/tpi_downloads_uk.jsp

Get an overview of what is RUP and how to apply it.

http://en.wikipedia.org/wiki/Rational_Unified_Process

Read the ISTQB syllabus (and take the certification)

<http://www.istqb.org/index.php?id=8>
[Syllabus Foundation Level](#)
[Syllabus Advanced Level](#)

Keep surfing the WEB !!