Com S 417
Software Testing

Fall 2017 – Week 7, Lecture 11

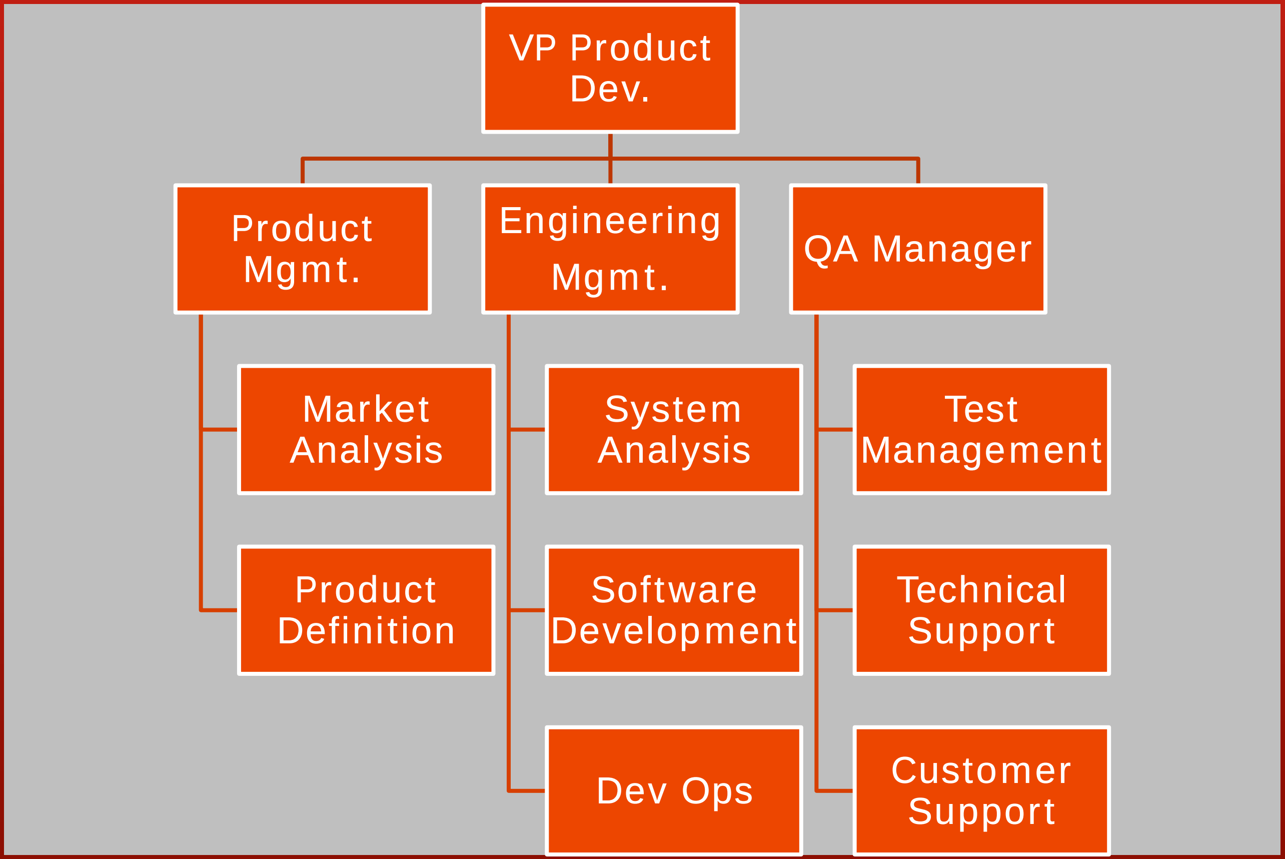
Announcements

- Chapter 4 O&A is available on digital reserve.
- We will probably only have 4 projects (not 5).

Topics

- Traditional Test Management
- Continuous Integration
- DevOps
- TDD (Test Driven Development)

Typical Dev Org. Structure

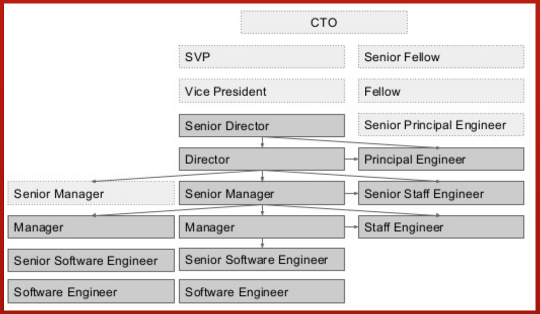


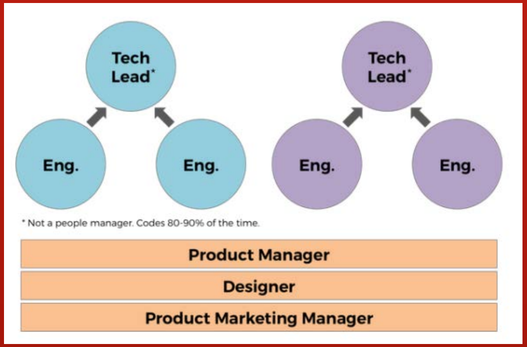
Traditional
Roles and Responsibilities

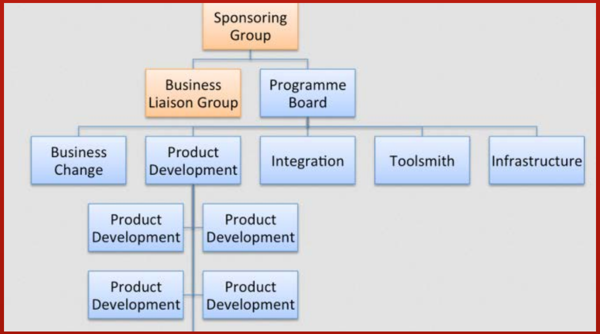
- Product management
  - Identify product opportunities
  - Articulate Product Vision
  - Define market strategy
  - Facilitate “early adopter” input
- Engineering Management
  - Negotiate Resources and Schedule
  - Elicit Agreed to Requirements
  - Solution Design
  - Monitor project progress
  - Manage DevOps Services
- QA Management
  - Manage Test Program
    - Set Test Strategy
    - Create and execute Test Plans
    - Report test and other quality metrics
    - Suggest process improvements
  - Manage Customer Service
    - End User Support
    - End User Documentation Services
  - Manage Technical Support
    - Custom integration, installation, configuration, and other technical services
    - Technical Documentation

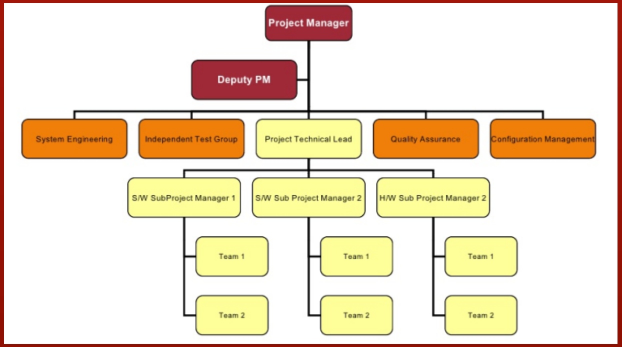
‘Typical Structure’
Representive, not Universal

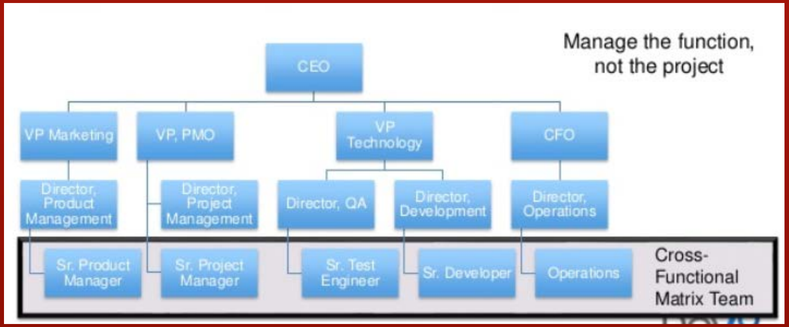
- A look at google images offers *lots* of alternatives:




Tech Lead*
Tech Lead*
Eng.
Eng.
Eng.
Eng.
* Not a people manager. Codes 80-90% of the time.
Product Manager
Designer
Product Marketing Manager



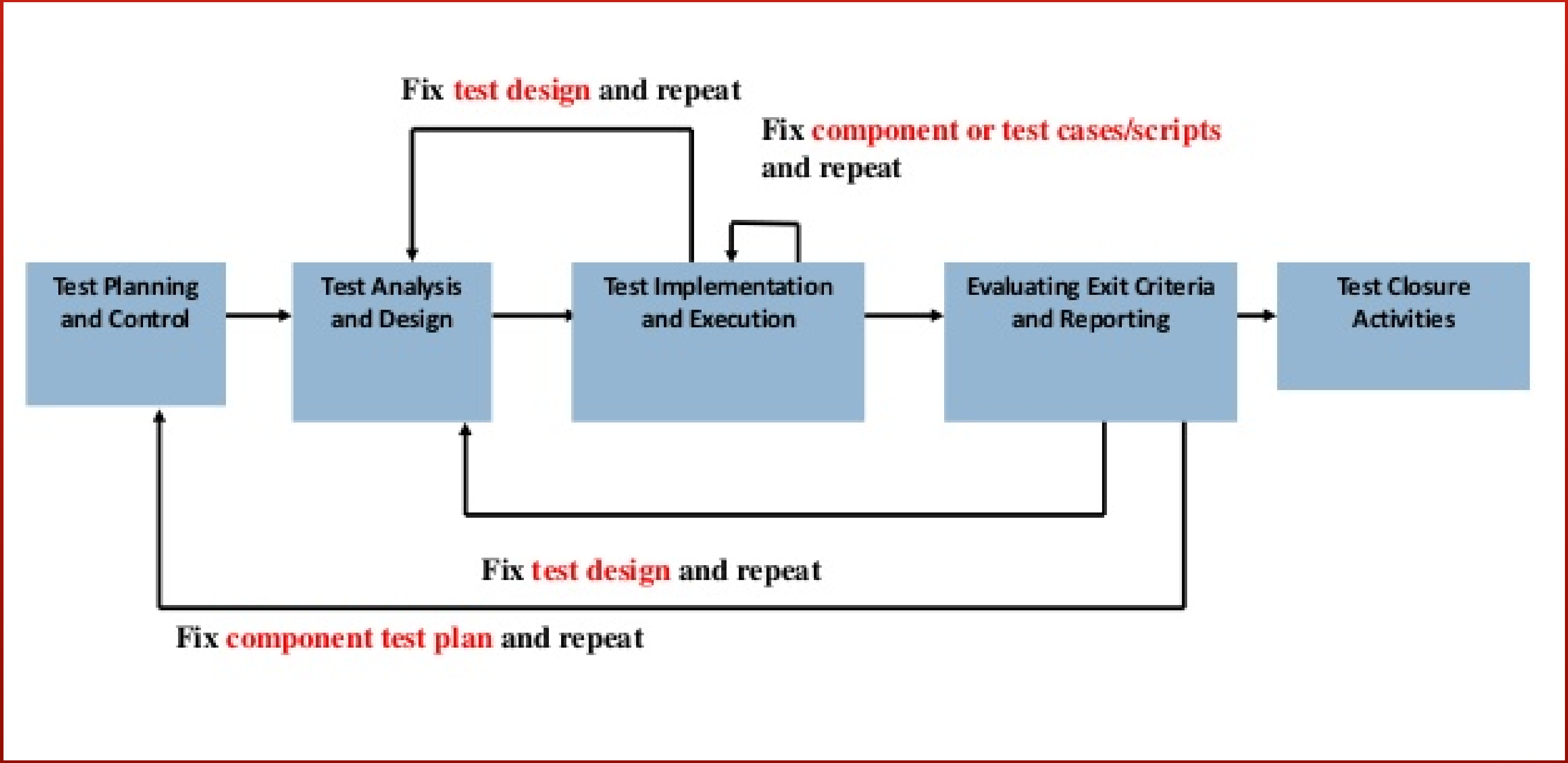




Test Stakeholders

- Developers
- Engineering management
- QA Management
- Customer Support
- Technical Support
- Customers
- Marketing and Product Management
- Executive Management

The ‘Top-Level’ Test Process


Fix test design and repeat
Fix component or test cases/scripts and repeat
Test Planning and Control
Test Analysis and Design
Test Implementation and Execution
Evaluating Exit Criteria and Reporting
Test Closure Activities
Fix test design and repeat
Fix component test plan and repeat

from: https://www.slideshare.net/joshisagarr/fundamentals-of-software-testing-53487079

Fundamental Test Cycle

- Planning
  - Construction of test plan(s)
- Specification
  - Selection, description, and implementation of test cases
- Execution
  - Execute the test cases.
- Recording
  - Complete test logs and defect reports.
- Check for Completion
  - Check Records against Completion Criteria
- Test Closure activities
  - Handovers, Retrospective, Archive Artifacts

Communication Among Silos

Key Artifacts (often dedicated databases of some kind)

- Requirements Repository
- Project Plan
- Progress Reports
- Defect Reports and Defect tracking
- Test Case/Script library
- Integrated traceability support

Test Plan

ANSI/IEEE Standard 829-1983:

- “A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies
  - test items,
  - the features to be tested,
  - the testing tasks,
  - who will do each task,
  - and any risks requiring contingency planning. “

Replaced by 829-2008:

Example detailed template:

www.ecs.csun.edu/~rlingard/comp480/TestPlanTemplate.pdf

Test Plan Purpose (2008 version)

The purpose of this standard is to:

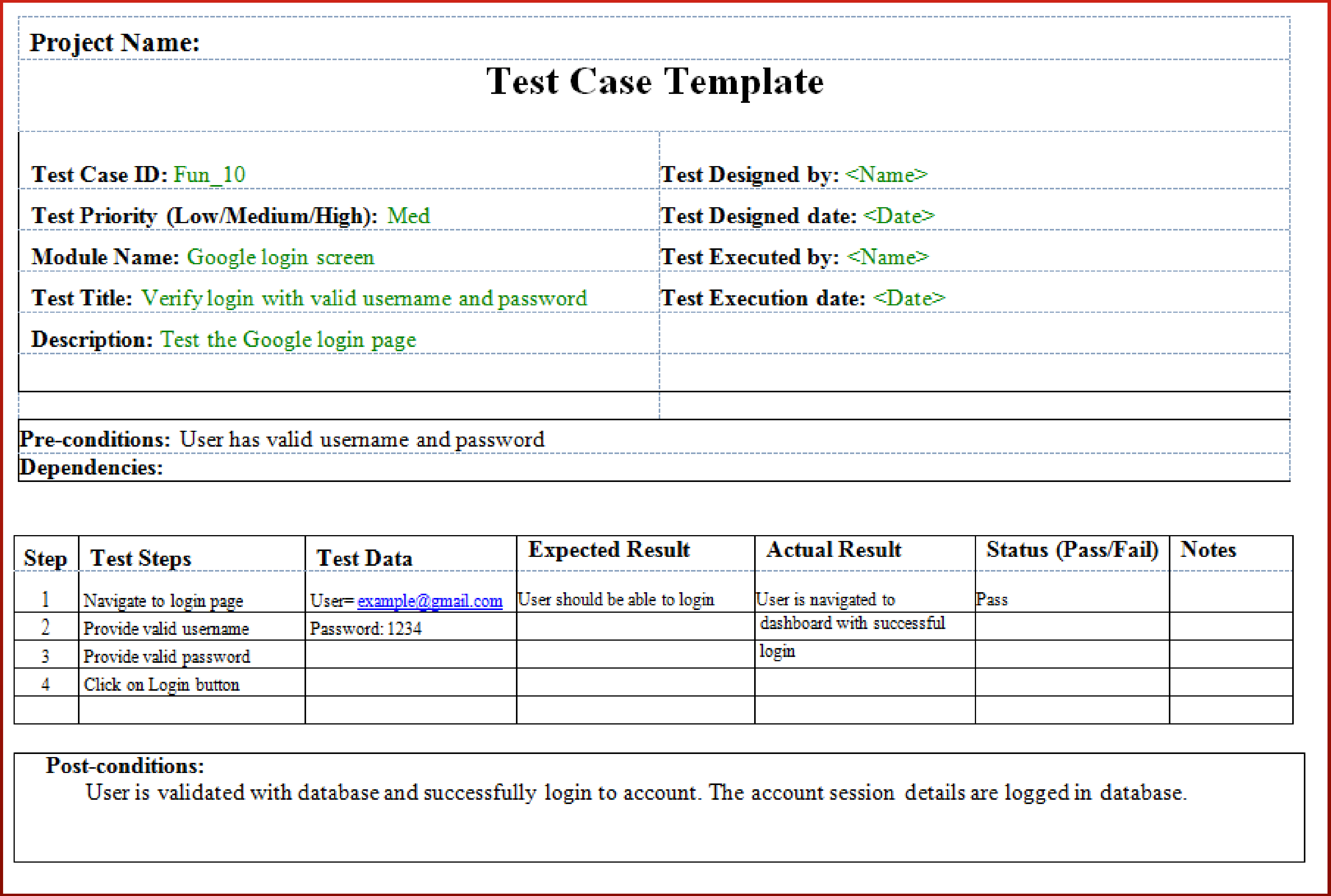
- Establish a common framework for test processes, activities, and tasks in support of all software life cycle processes, including acquisition, supply, development, operation, and maintenance processes
- Define the test tasks, required inputs, and required outputs
- Identify the recommended minimum test tasks corresponding to integrity levels for a four-level integrity scheme (see the used example in 4.1)
- Define the use and contents of the Master Test Plan and the Level Test Plan(s) (e.g., for component, integration, system, and acceptance test)
- Define the use and contents of related test documentation (Test Design, Test Case, Test Procedure, Anomaly Report, Test Log, Level Test Report, Interim Test Report, and Master Test Report)

Test Plan – brief outline

1. Purpose
2. Target Audience and Application
3. Deliverables
4. Information Included
   - Introduction
   - Test Items
   - Features Tested
   - Features not Tested
   - Test Criteria, Strategy and Approach
   - Pass/fail Standards
   - Criteria for beginning Testing
   - Criteria for suspending test
   - Test Deliverables
   - Hardware and Software Requirements
   - Responsibilities for determining problem severity and correcting problems.
   - staffing and training needs
   - Test schedules
   - Risks and contingencies
   - Approvals

From Ammann and Offutt p. 226-227

The Test Script (Test Case)



Project Name:

Test Case Template

Test Case ID: Fun_10	**Test Designed by:** <Name>
Test Priority (Low/Medium/High): Med	**Test Designed date:** <Date>
Module Name: Google login screen	**Test Executed by:** <Name>
Test Title: Verify login with valid username and password	**Test Execution date:** <Date>
Description: Test the Google login page	

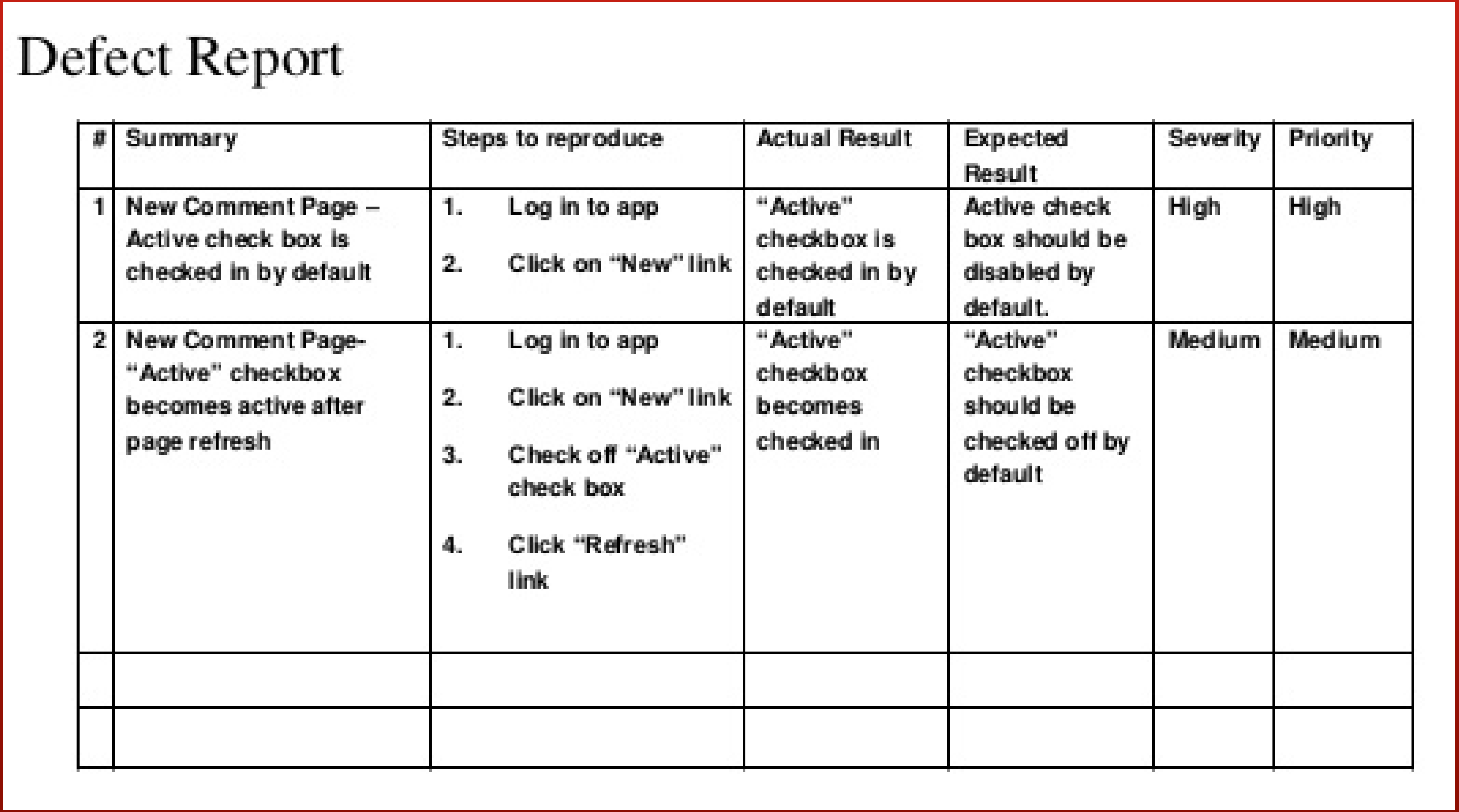
Pre-conditions: User has valid username and password
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User= example@gmail.com	User should be able to login	User is navigated to	Pass	
2	Provide valid username	Password: 1234		dashboard with successful		
3	Provide valid password			login		
4	Click on Login button					

Post-conditions:
User is validated with database and successfully login to account. The account session details are logged in database.

from: http://www.softwaretestinghelp.com/test-case-template-examples/

Anomaly (defect) report



Defect Report

#	Summary	Steps to reproduce	Actual Result	Expected Result	Severity	Priority
1	New Comment Page – Active check box is checked in by default	1. Log in to app 2. Click on "New" link	"Active" checkbox is checked in by default	Active check box should be disabled by default.	High	High
2	New Comment Page- "Active" checkbox becomes active after page refresh	1. Log in to app 2. Click on "New" link 3. Check off "Active" check box 4. Click "Refresh" link	"Active" checkbox becomes checked in	"Active" checkbox should be checked off by default	Medium	Medium

from: https://www.slideshare.net/mulyavkav/bug-reporting-and-tracking

IEEE Standard Anomaly Classification (1044-2009)

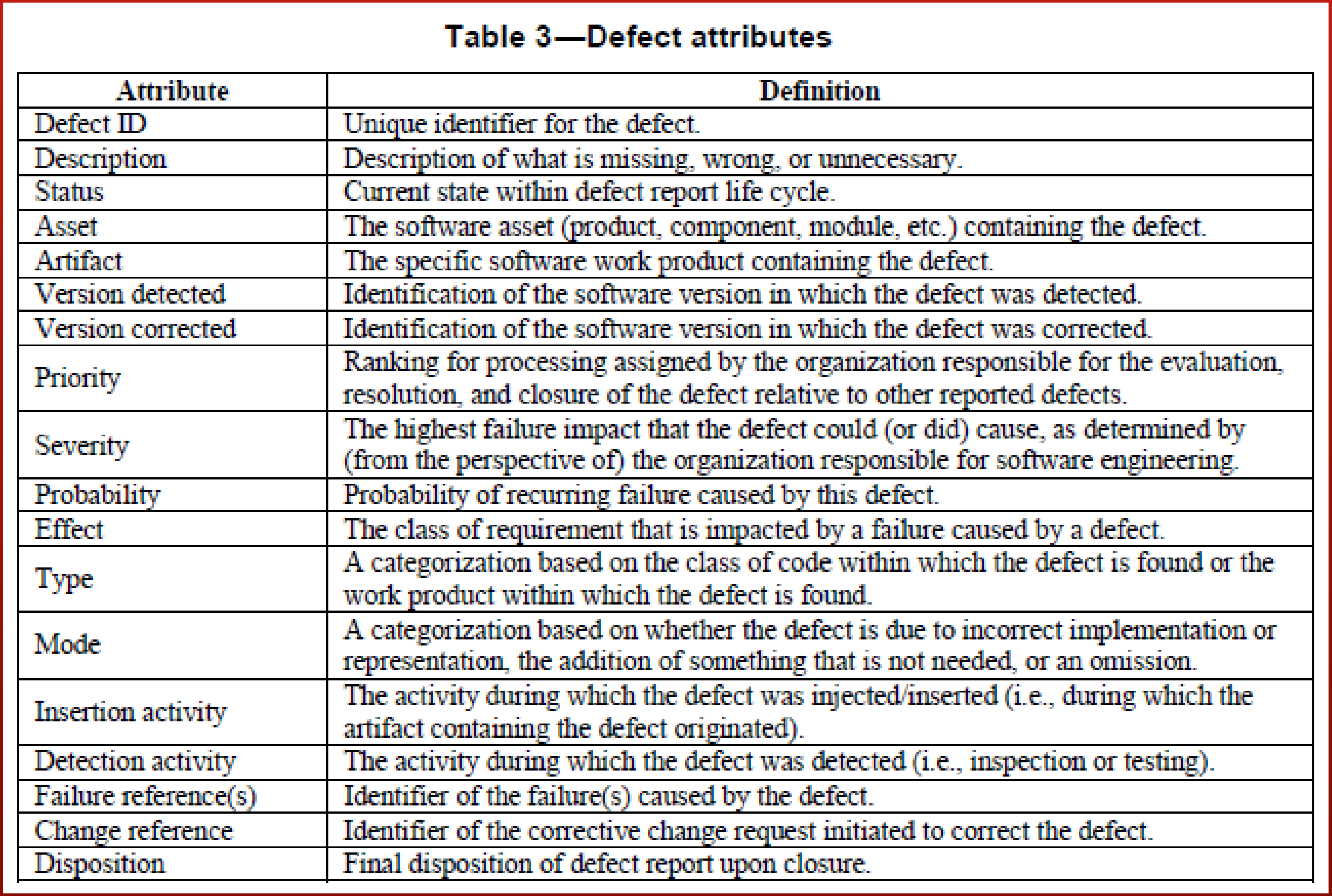
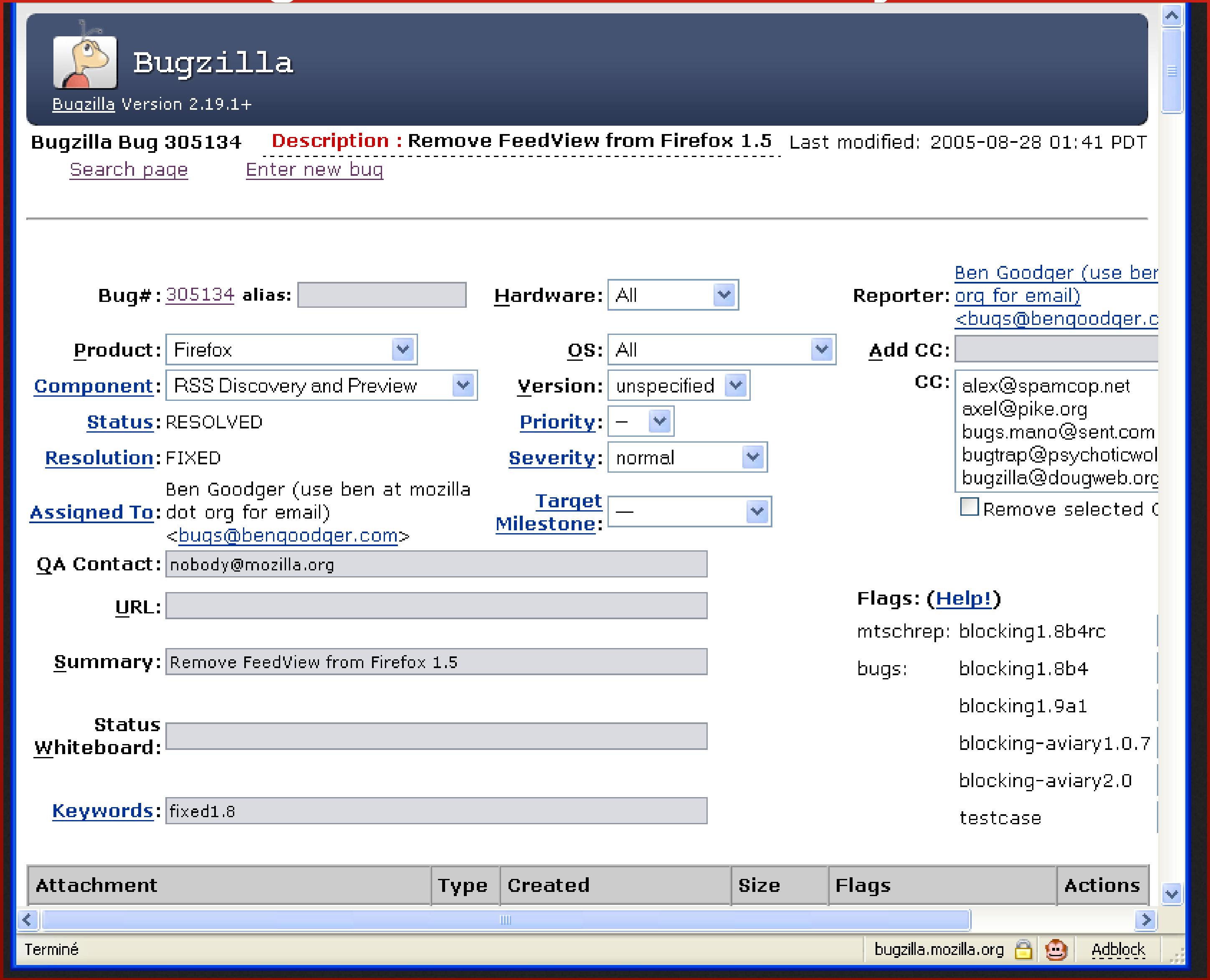


Table 3—Defect attributes

Attribute	Definition
Defect ID	Unique identifier for the defect.
Description	Description of what is missing, wrong, or unnecessary.
Status	Current state within defect report life cycle.
Asset	The software asset (product, component, module, etc.) containing the defect.
Artifact	The specific software work product containing the defect.
Version detected	Identification of the software version in which the defect was detected.
Version corrected	Identification of the software version in which the defect was corrected.
Priority	Ranking for processing assigned by the organization responsible for the evaluation, resolution, and closure of the defect relative to other reported defects.
Severity	The highest failure impact that the defect could (or did) cause, as determined by (from the perspective of) the organization responsible for software engineering.
Probability	Probability of recurring failure caused by this defect.
Effect	The class of requirement that is impacted by a failure caused by a defect.
Type	A categorization based on the class of code within which the defect is found or the work product within which the defect is found.
Mode	A categorization based on whether the defect is due to incorrect implementation or representation, the addition of something that is not needed, or an omission.
Insertion activity	The activity during which the defect was injected/inserted (i.e., during which the artifact containing the defect originated).
Detection activity	The activity during which the defect was detected (i.e., inspection or testing).
Failure reference(s)	Identifier of the failure(s) caused by the defect.
Change reference	Identifier of the corrective change request initiated to correct the defect.
Disposition	Final disposition of defect report upon closure.

Bug Tracking Systems

Bugzilla – data entry



Bugzilla

Bugzilla Version 2.19.1+

Bugzilla Bug 305134 **Description : Remove FeedView from Firefox 1.5** Last modified: 2005-08-28 01:41 PDT

Search page Enter new bug

Bug#:	305134 **alias:**	**Hardware:**	All	**Reporter:**	Ben Goodger (use ben org for email) <bugs@bengoodger.c
Product:	Firefox	**OS:**	All	**Add CC:**	
Component:	RSS Discovery and Preview	**Version:**	unspecified	**CC:**	alex@spamcop.net axel@pike.org bugs.mano@sent.com bugtrap@psychoticwol bugzilla@dougweb.orc
Status:	RESOLVED	**Priority:**	--		
Resolution:	FIXED	**Severity:**	normal		
Assigned To:	Ben Goodger (use ben at mozilla dot org for email) <bugs@bengoodger.com>	**Target Milestone:**	---		Remove selected C
QA Contact:	nobody@mozilla.org				
URL:					
Summary:	Remove FeedView from Firefox 1.5				
Status Whiteboard:					
Keywords:	fixed1.8				

Flags: (Help!)

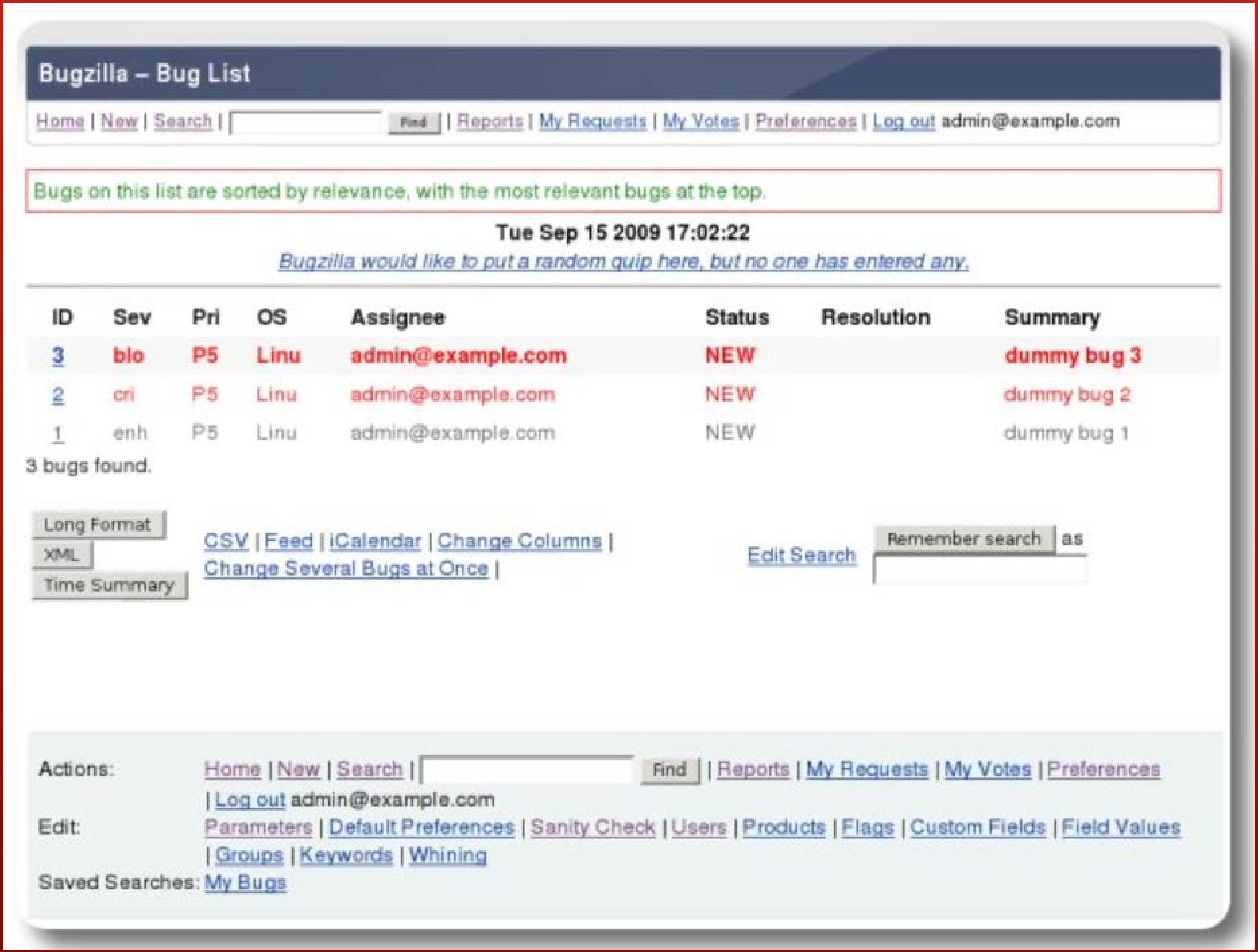
mtschrep:	blocking1.8b4rc
bugs:	blocking1.8b4
	blocking1.9a1
	blocking-aviary1.0.7
	blocking-aviary2.0
	testcase

Attachment	Type	Created	Size	Flags	Actions

Terminé bugzilla.mozilla.org Adblock

Bug Tracking Systems

Bugzilla – bug list



Bugzilla – Bug List

Home | New | Search | Find | Reports | My Requests | My Votes | Preferences | Log out admin@example.com

Bugs on this list are sorted by relevance, with the most relevant bugs at the top.

Tue Sep 15 2009 17:02:22

Bugzilla would like to put a random quip here, but no one has entered any.

ID	Sev	Pri	OS	Assignee	Status	Resolution	Summary
3	blo	P5	Linu	admin@example.com	NEW		dummy bug 3
2	cri	P5	Linu	admin@example.com	NEW		dummy bug 2
1	enh	P5	Linu	admin@example.com	NEW		dummy bug 1

3 bugs found.

Long Format
XML
Time Summary
CSV | Feed | iCalendar | Change Columns |
Change Several Bugs at Once |
Edit Search
Remember search as

Actions: Home | New | Search | Find | Reports | My Requests | My Votes | Preferences | Log out admin@example.com

Edit: Parameters | Default Preferences | Sanity Check | Users | Products | Flags | Custom Fields | Field Values | Groups | Keywords | Whining

Saved Searches: My Bugs

Bug Tracking Systems

Jira - Atlassian

- Commercial
- Core application is an 80’s style screen and report generator.
- Many plugins, integrations, and add-on’s available.

You either love it or hate it.

Integrated
Tracking and Traceability

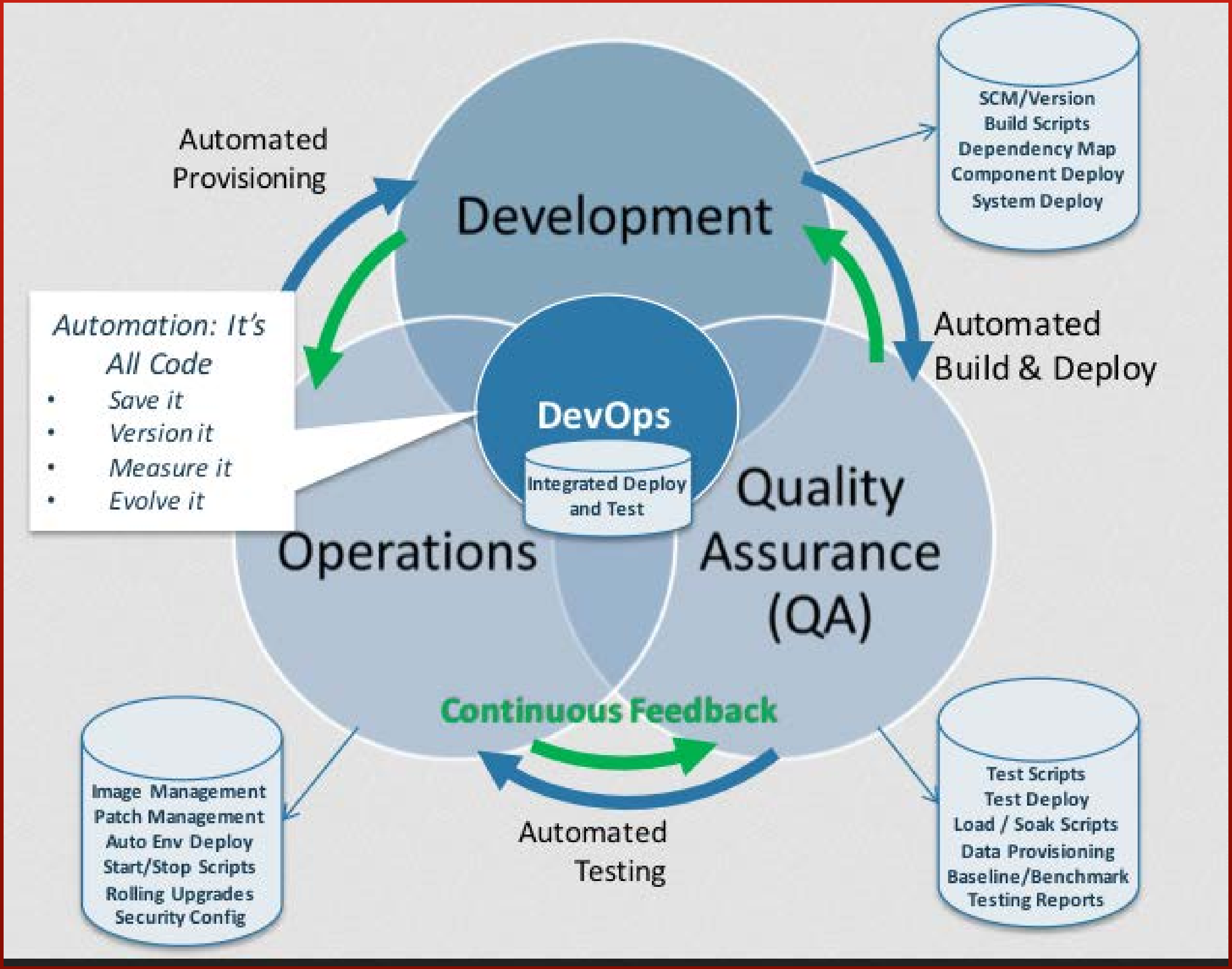
Modern, large scale operations tend to use commercial tools that link related requirements, code, test cases, and defects.

https://support.smartbear.com/screencasts/qacomplete/requirements-defects-traceability/

IBM Doors and RTC

also https://www.slideshare.net/Intland/from-requirements-management-to-release-with-git-for-android-system

What is DevOps?



from: https://techbeacon.com/7-steps-choosing-right-devops-tools

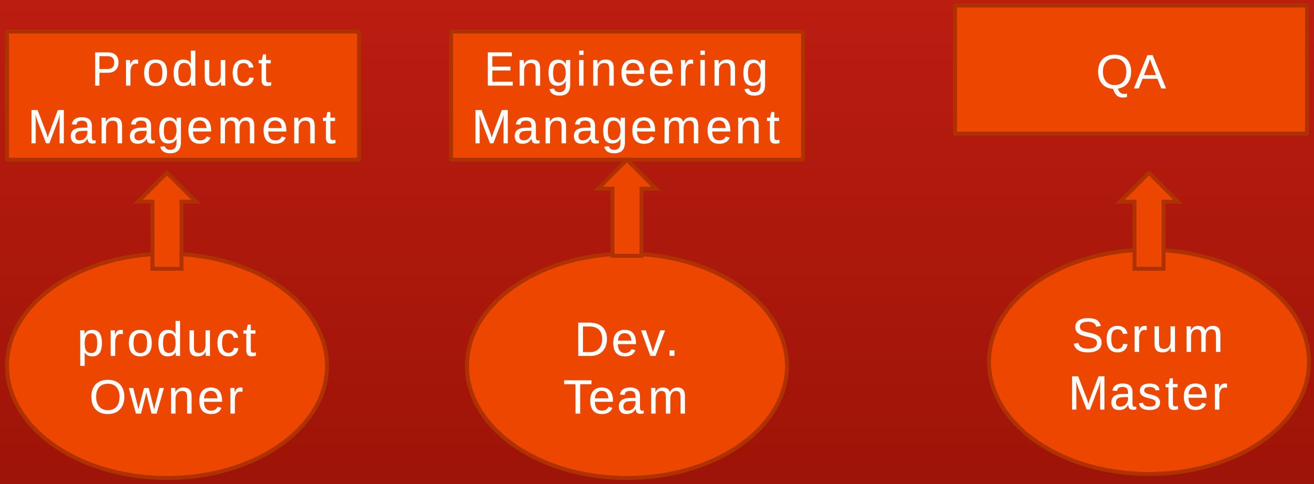
Typical DevOps Services

- Requirements manager
- Source code control/repository
- Integrated/automated build, configure, deploy, test, report
- Test management services
- Integration between test and requirements management systems to support traceability.
- Integration between auto test execution, test management, and requirements management to support traceability and impact analysis.
- Project dashboards/information ‘radiators’

How is DevOps different than Ops?

- Open, collaborative approach built around CI goals.
- Focus on self-service access to highly integrated and automated services.
- System administration tasks become ‘invisible’, even to devoloper-users.

Scrum and Balance of Power



TDD: The Discipline

The Three Rules of TDD:

- You are not allowed to write any production code unless it is to make a failing unit test pass.
- You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.
- You are not allowed to write any more production code than is sufficient to pass the one failing unit test

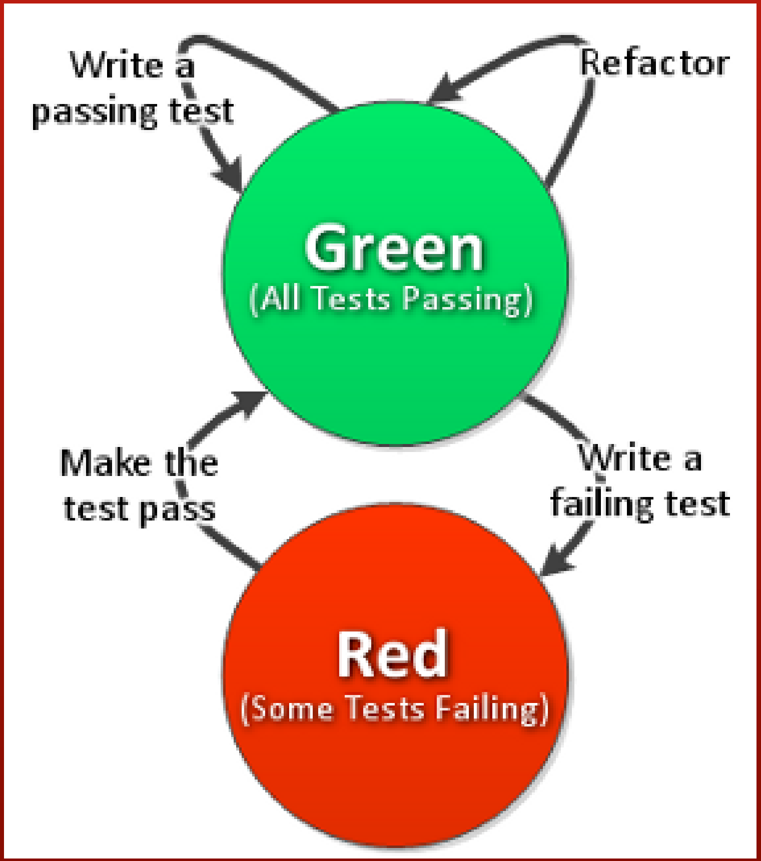


Figure from: http://www.peterprovost.org/blog/2012/05/02/kata-the-only-way-to-learn-tdd/
Rules from http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd

The FizzBuzz Code Kata

https://www.youtube.com/watch?v=JyRouDwzCoo

- Code Kata
  - Code Katas are to programming as Required Figures are to figure skating.
  - A Code Kata is an exercise designed to build and demonstrate technical precision, accuracy, and skill.

Note: watch the lower left corner of the screen for the test result when the camera zooms out.

Writing code == writing tests

No handoffs. No deferred activities. Seamless integration.

Reading Assignment

Fundamental ISTQB Test Process

- https://www.testingexcellence.com/fundamental-test-process-software-testing/

Suggested Optional Resources:

- Links to detailed descriptions of various aspects of test process
  - http://www.softwaretestingmentor.com/istqb-advanced-certification/testing-process/
- More CodeKatas
  - CodeKata.com