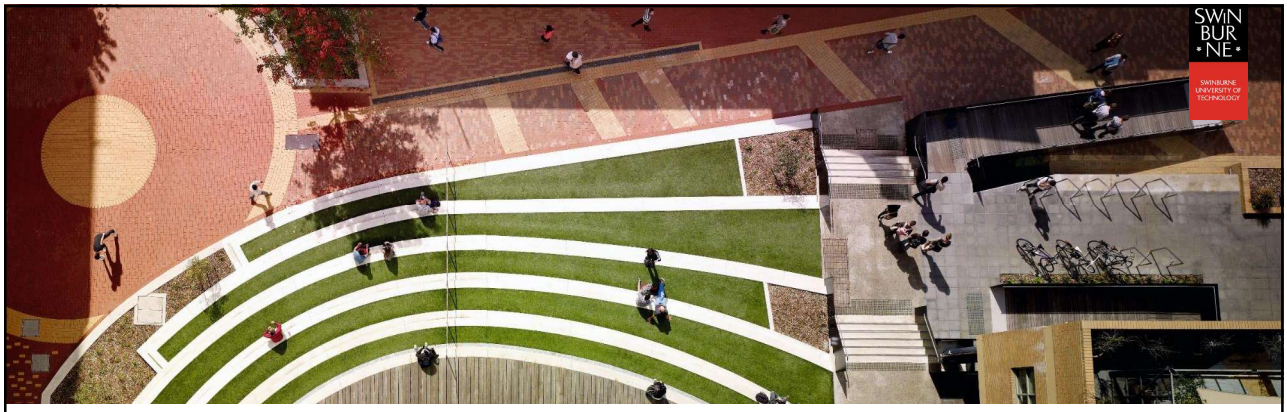


COS80022 – Software Quality and Testing

Week 12 – Agile Testing

Outline

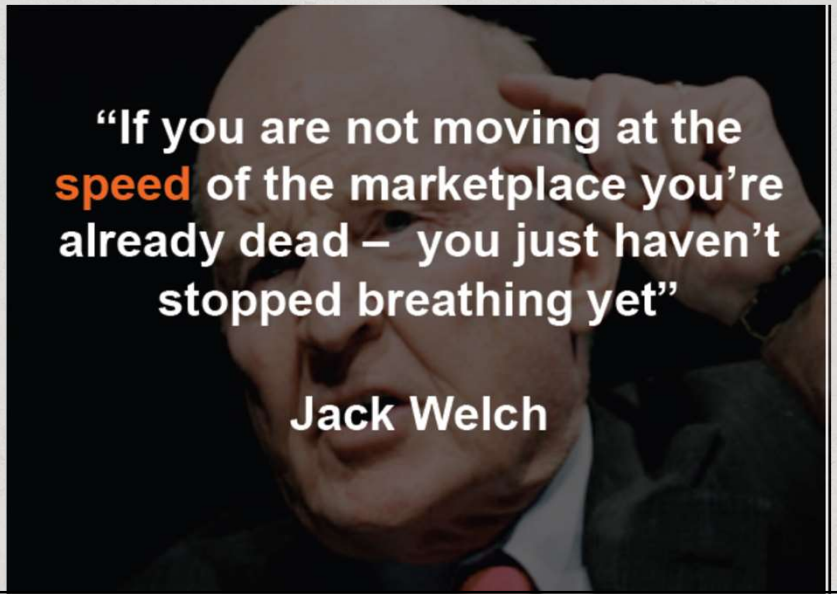
- Agile development
- Testing in agile
- Agile teams
- Tools and automation

Agile Development

Motivation

- Rapidly changing customer's requirement
  - Speed
  - Customer
  - Change

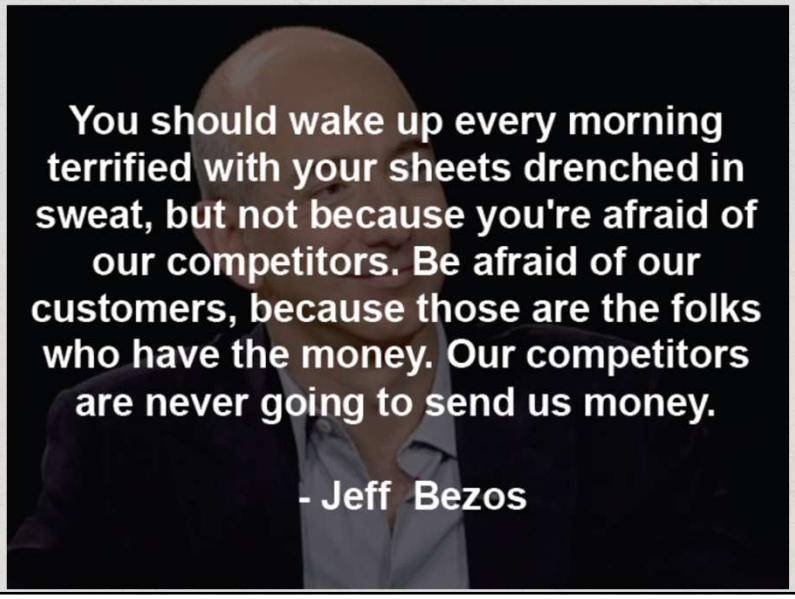
Motivation – Speed



Motivation – Speed

- R&D: 10% of revenue ($100m out of $1b)
  - Product development cycle: 12 months
- Approach A: improve efficiency by 10%
  - $10m saving in development cost
- Approach B: reduce cycle by 10%
  - Product sold 1.2 months earlier
    → $100m more revenue

Motivation – Customer

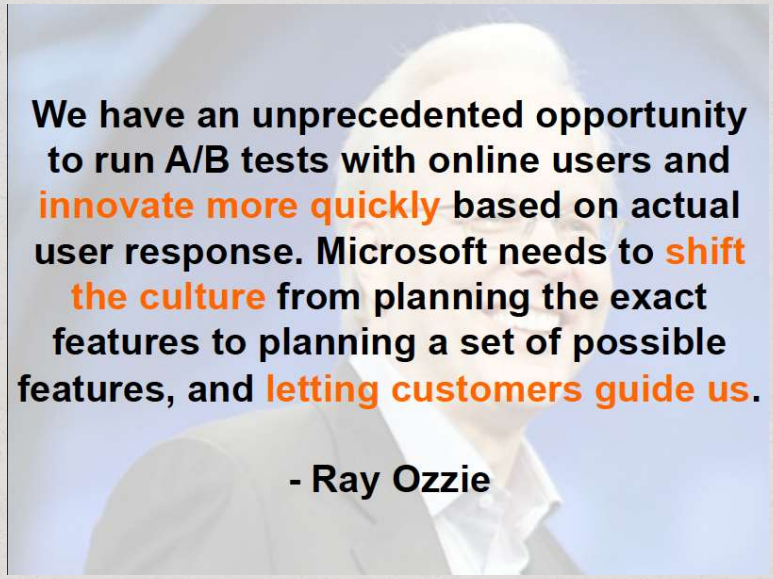


7

Motivation – Customer

8

Motivation – Change



9

Motivation – Change

The Myth	The Reality
Inspired innovation	Create and winnow 10 pixel-perfect prototypes
Inspired design	Build a better backstory (intricate layers of business design behind the products)
Brilliantly inspired marketing	Engineer the perfect customer experience to create customer experience and buzz

10

Motivation

- Can we survive in the current business environment?

  Yes We Can!

- One solution: agile development

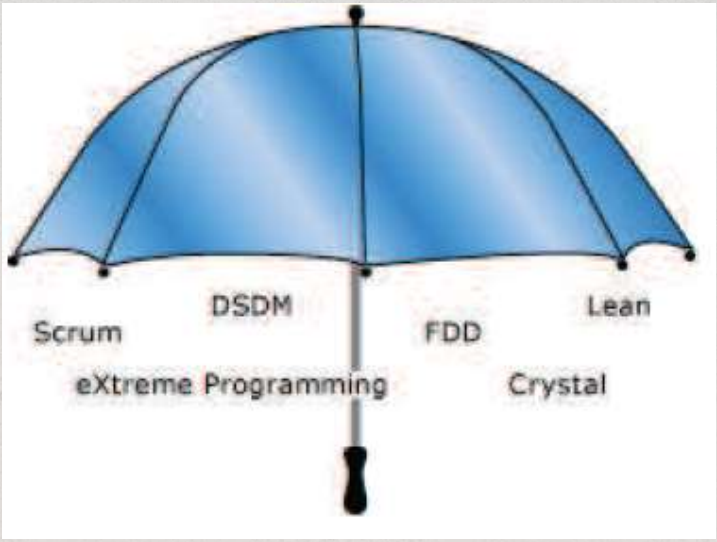
Rapid software development

- Rapid development and delivery is now an important requirement for software systems
  - Businesses operate in a fast-changing environment
  - Software has to evolve quickly to reflect changing business needs
  - Not practically possible to produce a set of stable software requirements
- Specification, design and implementation are interleaved
- System is developed as a series of versions with stakeholders involved in version evaluation

Agile development

Concept

Concept



Concept – Manifesto

Individuals and interactions

over

Processes and tools

Concept – Manifesto

Individuals and interactions

over

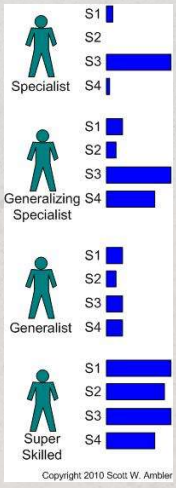
Processes and tools

- Projects are built around motivated individuals, who should be trusted
- Face-to-face conversation is the best form of communication (co-location)
- Continuous attention to technical excellence and good design
- Self-organizing teams

Agile is People-Centred

- Agile teams should be
  - *Self-disciplined*: only commit to work that can be accomplished, complete that work as effectively as possible
  - *Self-organizing*: estimate and plan own work and then collaborate iteratively to complete it
  - *Self-aware*: strive to identify what works well for the team, what doesn't, learn and adjust accordingly
- Team members should be "generalizing specialists"
  - Jack-of-all-trades and a master-of-few

Generalising Specialists

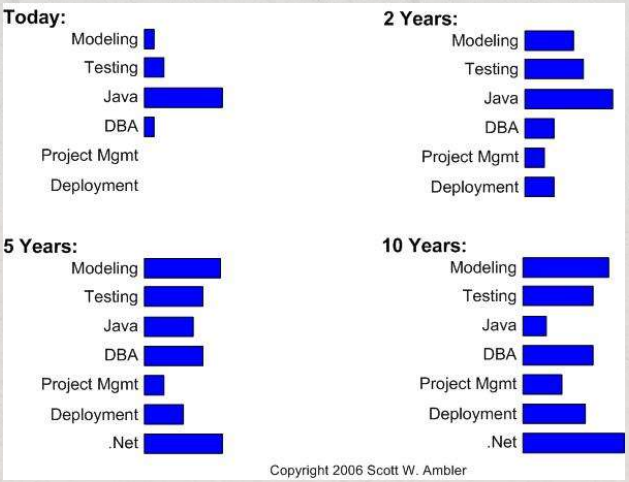


- **Specialists** have *deep skills* in a single speciality
- **Generalists** have *broad skills* in a range of specialities
- **Generalizing specialists** have broad skills in a range of disciplines and deep skills in one or more specialities – All referred to as *T-skilled*
- **True experts** have deep skills in many specialities

Source: http://agilemodeling.com/essays/generalizingSpecialists.htm

Generalising Specialist

- Ideally, agile teams have generalizing specialists as it enables collaboration
- Specialists can become generalizing specialists with training, coaching and non-solo development



Source: http://agilemodeling.com/essays/generalizingSpecialists.htm

Concept – Manifesto

Working software

over

Comprehensive documentation

(working software)

OVER

Concept – Manifesto

Working software

over

Comprehensive documentation

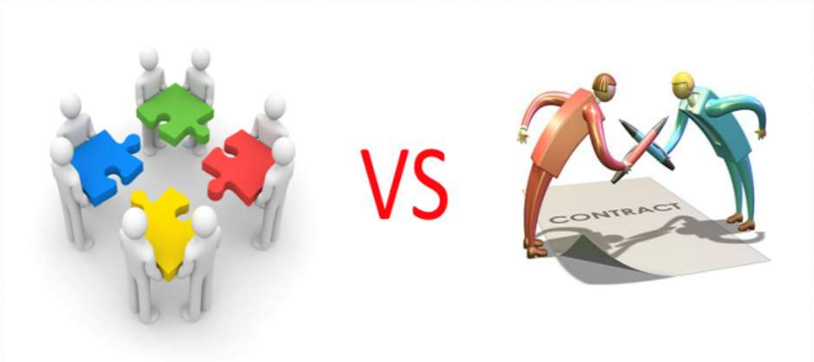
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Simplicity is essential

Concept – Manifesto

Customer collaboration

over

Contract negotiation



Concept – Manifesto

Customer collaboration

over

Contract negotiation

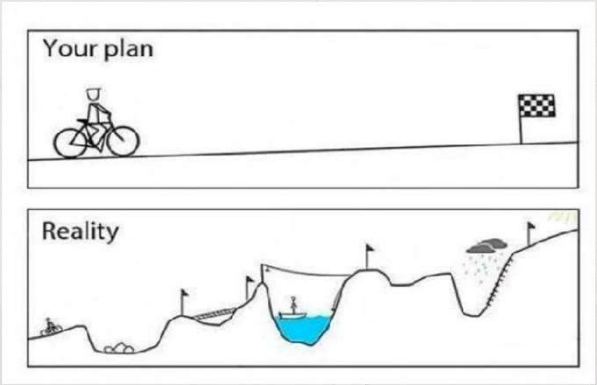
- Customer satisfaction by rapid delivery of useful software
- Close, daily cooperation between business people and developers

Concept – Manifesto

Responding to change

over

Following a plan



Concept – Manifesto

Responding to change

over

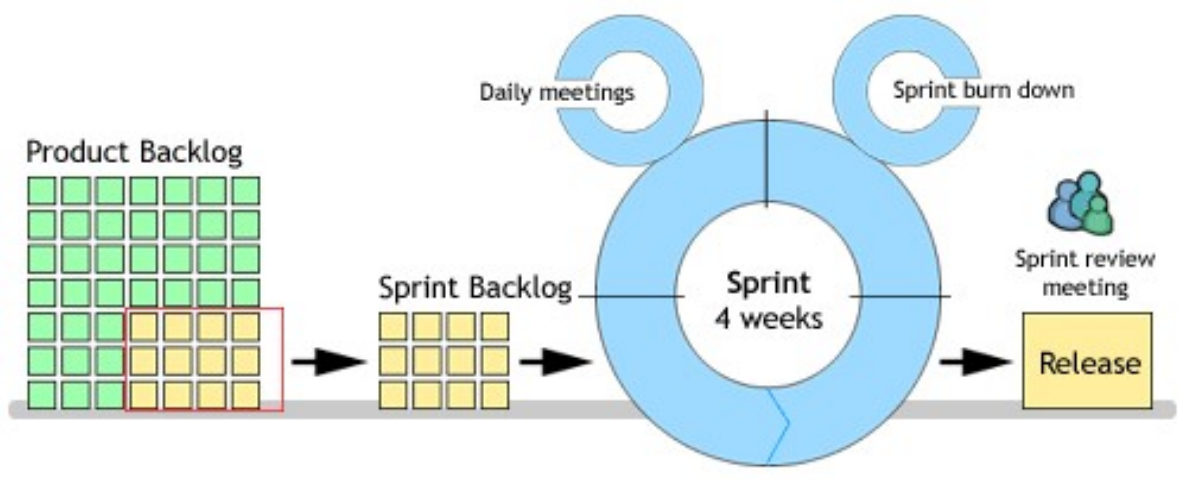
Following a plan

- Welcome changing requirements, even late in development
- Regular adaptation to changing circumstances

Concept

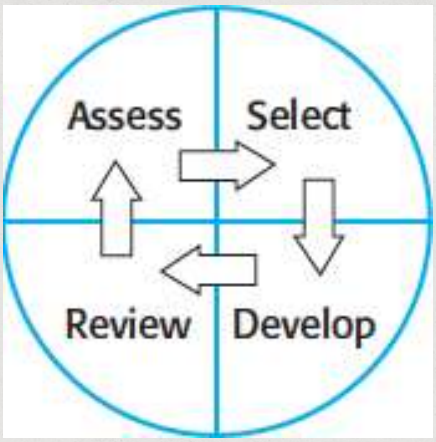
- Rapid iterations
- Small and frequent releases
- Evolving requirements
- Direct customer involvement

Scrum



Scrum

- Sprint cycle
  - 2-4 weeks
  - Self-organised
  - Daily meeting

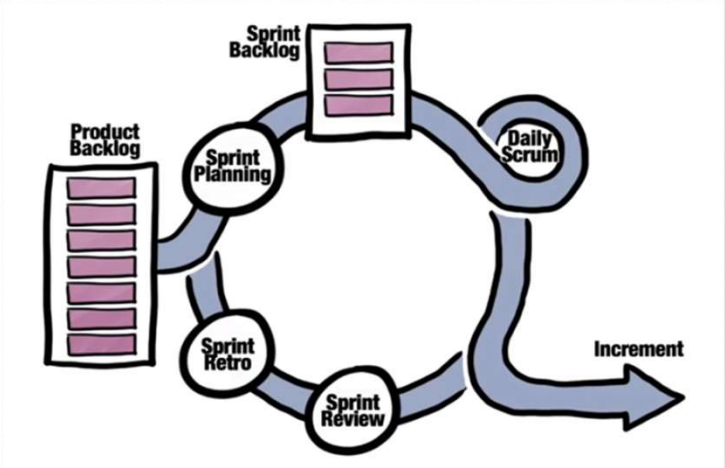


Scrum

- An agile framework for managing software development
- Lightweight, iterative and incremental
- Comprises three phases
  - *Pre-Game (Planning + Architecture)*
    - Establish the general objectives for the project
    - Determine team structure
    - High level design of system architecture
  - *Game phase* (a.k.a - Construction)
    - Series of sprint cycles
    - Each cycle develops an increment of the system
  - *Post-Game phase* (Project wrap up)
    - Prepare for release
    - Complete required documentation
    - Assess the lessons learned from the project

Anatomy of Scrum

- **Roles** – Product Owner, Scrum Master (or Team Leader), Team Member
- Meetings – Sprint Planning, Daily Scrum, Sprint Review/Retrospective
- Artifacts – Product Backlog, Sprint Backlog, Tasklist, ScrumBoard



Source: https://www.developmentthatpays.com/posts/132-scrum-increment

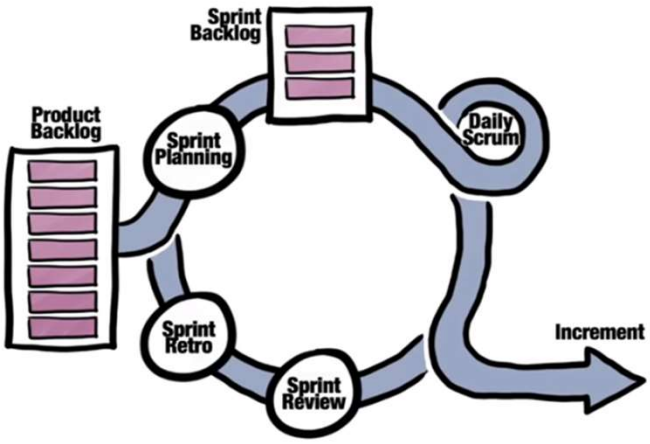
Scrum - Roles

- Product Owner
  - Remember "Customer Collaboration"?
  - Product owner represents the stakeholders
  - Responsible for maximizing value of product resulting from development
- Scrum Master
  - Leads the team by facilitating, NOT managing the team
  - Removes impediments to development team's progress
  - Helps product owner to maximize value from product development
- Team Member
  - Scrum teams are cross-functional
  - Team members have different skill sets (*developers*, *testers*, *designers*, *Dev-Opers*) and cross train each other
  - They are self-organizing (can organize and manage their own work)

https://www.scrumalliance.org/agile-resources/scrum-roles-demystified

Anatomy of Scrum

- Roles – Product Owner, Scrum Master (or Team Leader), Team Member
- **Meetings** – Sprint Planning, Daily Scrum, Sprint Review/Retrospective
- Artifacts – Product Backlog, Sprint Backlog, Tasklist, ScrumBoard



Meeting Agenda Example

Date/Location: 22-March-2018 at 14:30 in ABC123

Information Updates/Reminders:

- Internal target date for Test plan is 29-March-2018
- Bob is concerned about team members not using git yet!

Decisions Needed :

- Need to confirm recommended coding standard (Coding standard selected and rationale is attached to this agenda)
- Formal team meeting day/time needs to change. ZZ is unable to attend due to changed part-time work schedule.

General Items :

- Why is git not being used?
- The graphics library currently in use is not very effective. KK will present issues and potential ideas to resolve situation.
- Categories for defect classification. Orthogonal or not?

Roles

- Team Leader: Chairs the meeting and is responsible for controlling the meeting
- Deputy Team Leader: Records the Meeting minutes and is responsible for circulation of Meeting minutes as well as the agenda for the Meeting.

Meeting Process (1)

- Team Leader starts the Meeting formally
- Team Leader presents a short summary of the current state of the project, highlighting any key accomplishments (max. 3 minutes)

Meeting Process (2)

- All team members talk for 1 min. giving a status update.
- The format is: *"I completed ...."*, *"I am currently working on ..."*, *"The next tasks for me are ...."*. Members present this information clock-wise from the team-leader.
- Team Leader now reads all items on the agenda. At the end, they formally call for any other items to be added to the agenda. New items can be added to the agenda in this step.

Meeting Process (3)

- Deputy Team Leader starts the stop-watch and records the “Start Time for the Meeting” in HH:mm 24-hour format. [CRITICAL STEP]
- Each item on the agenda is discussed sequentially. If item is “NOT” on the agenda it is **“NOT”** discussed.

Meeting Process (4)

- If a decision is made, it is recorded in the “decisions” section of the meeting minutes. Notes to provide the rationale for the decision is also recorded.
- If an action arises due to a decision – it is recorded in the “actions” section of the meeting minutes.

Meeting Process (5)

- At the end of the Meeting, the Team Leader formally closes the Meeting. Timer is stopped. Team is informed of the Time spent in the Meeting.
- The Deputy Team Leader reads out the Meeting minutes
- All team members that have Actions verbally acknowledge the Actions assigned to them.

Meeting Process (6)

- The Meeting minutes are emailed to the “team” as well as the academic supervisor within 2 hours of the end of the Meeting.
- This email must attach the “agenda” as well as the “Meeting Minutes” and serves as the Meeting archive.
- The agenda is added, since most of the Time there are changes made to it at the start of the Meeting and new items are added.
- Meeting agenda and minutes are recorded in “plain text format”

Format for Meeting Minutes

- The Meeting Minutes are always recorded by the Deputy Team Leader and at a minimum has two sections:
  - (i) Actions, and
  - (ii) Decisions

Meeting Minutes Example

Date/Location: 22–Mar–2018, EN310
Attendees:
Decisions:

- Development environment will be Eclipse IDE 3.6.x.
- IEEE 888 will be standard used for quality control. This standard has been selected based on the recommendation of GP who investigated 4 alternatives.
- Rudolf will be responsible for the web site. He is replacing Bart who will be assisting programming team.

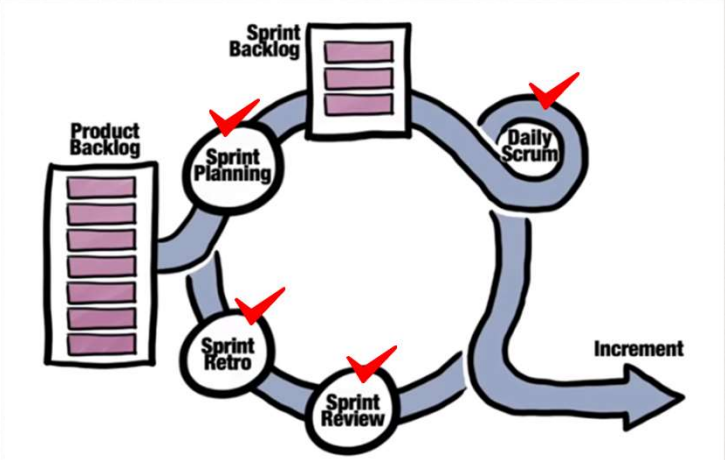
[**NOTE**: Decision is recorded first, followed by an optional rationale]
Actions:

- 26-Mar, RA, New template for web site will be submitted for review
- 28-Mar, CM, Identify libraries for MD5 (working with KP)

[**NOTE**: Columns in Actions are: Target date, Person responsible and brief note about the task].

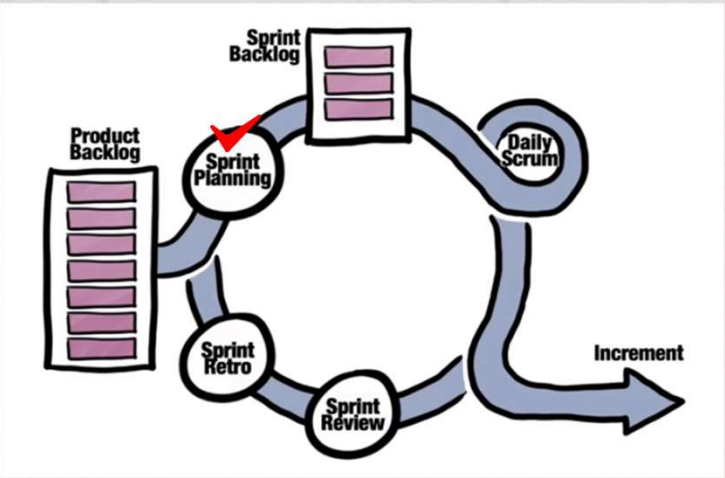
Scrum Meetings

- 4 Types of meetings
- *Timeboxing*
  - Timebox is a defined period of time during which a task must be accomplished
  - The meeting duration is fixed depending upon type of meeting
- Clear Agendas
- Involved Participants
  - Participants vary depending upon type of meeting



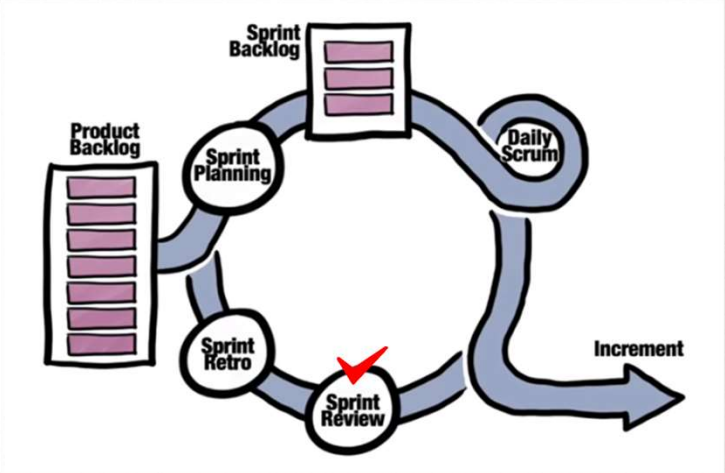
Sprint Planning

- Sets the team's goals for this Sprint
- Deliverable: a Sprint Backlog & an Estimated Task List for the Sprint
- Timeboxed: 1-2 hours (two times length of sprint in hours)
- Participants: Product Owner, Scrum Master & Development Team



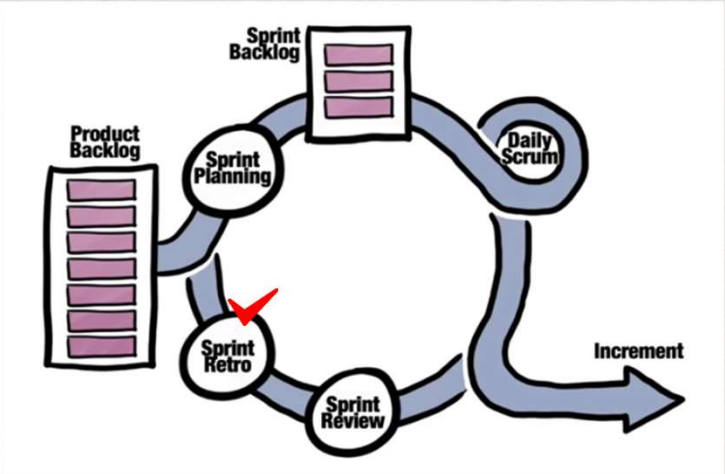
Sprint Review

- Demonstrate working product to Product Owner
- Review & evaluate product
- Review & update product backlog
- Timeboxed: 1-2 hours
- Participants: Product Owner, Development Team, Scrum Master, Mixture of management, outside stakeholders, customers, and even developers from other teams



Sprint Retrospective

- Things to keep doing
- Things to stop doing
- New things to try
- Timeboxed: 0.5 hour
- Participants: Development Team & Scrum Master. Product Owner is an optional attendee. No outside stakeholders.



Retrospectives

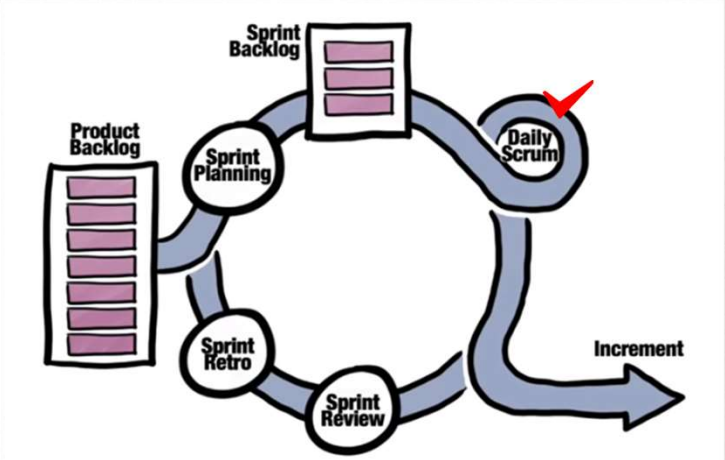
- Retrospectives are held at the end of each iteration and cover topics such as:
  - Process
  - People
  - Organisations
  - Relationships
  - Tools
- Looking for what was good, what was bad, what should be changed or retained

Attributes of Retrospectives

- Focus on improvement
- Remember to acknowledge what went well
- Be constructive

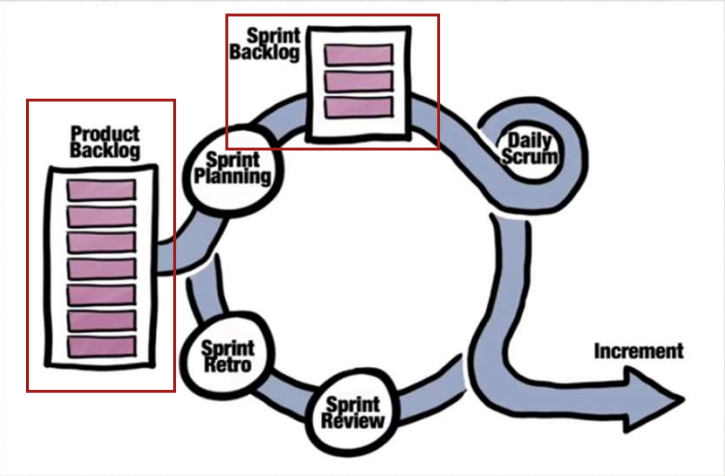
Daily Scrum Meeting (a.k.a Standup)

- What did you do yesterday? (since our last meeting)
- What will you do today? (Until our next meeting)
- What is blocking you?
- Timeboxed: 10 minutes
- Participants: Scrum Master & Development Team. Product Owner - optional



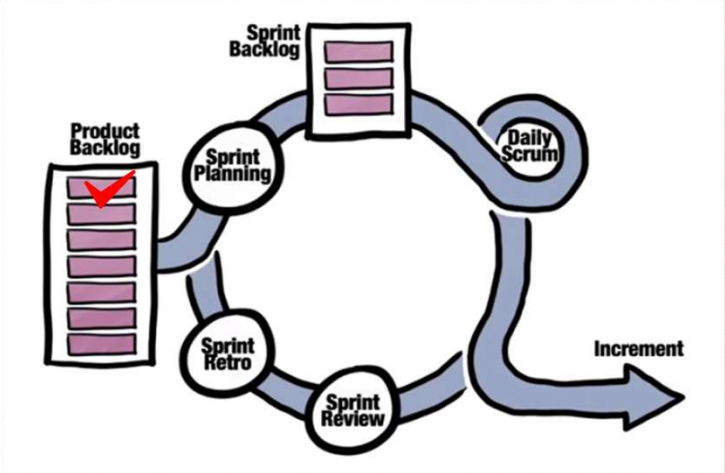
Anatomy of Scrum

- Roles – Product Owner, Scrum Master (or Team Leader), Team Member
- Meetings – Sprint Planning, Daily Scrum, Sprint Review/Retrospective
- **Artifacts** – Product Backlog, Sprint Backlog, Tasklist, ScrumBoard



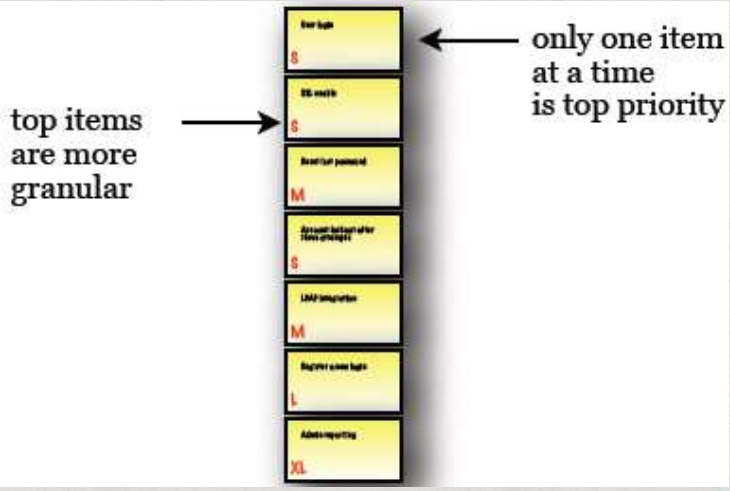
Product Backlog

- An ordered list of everything that might be needed in the product
- Single source of requirements for any changes to be made in the product
- Maintained and prioritized by the Product Owner
- Living Document
  - New requirements can be added
  - Existing requirements can be modified
  - Requirements are iteratively developed



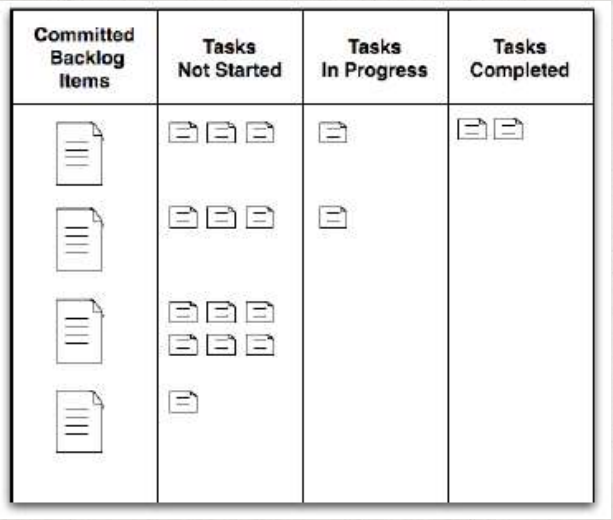
Scrum

- Product backlog
  - Requirements
  - Grow and change



Scrum

- Sprint backlog
  - List of work
  - To be addressed



User Stories

- A simplified, less formal technique for capturing user requirements
- Represents a placeholder to remind the team to have a discussion about the story with the users
- A story captures one user-centric functional flow

What is a User Story?

Creating a sprint backlog

Example Product Backlog

ToDo List

ID	Story	Estimation	Priority
7	As an unauthorized User I want to create a new account	3	1
1	As an unauthorized User I want to login	1	2
10	As an authorized User I want to logout	1	3
9	Create script to purge database	1	4
2	As an authorized User I want to see the list of items so that I can select one	2	5
4	As an authorized User I want to add a new item so that it appears in the list	5	6
3	As an authorized User I want to delete the selected item	2	7
5	As an authorized User I want to edit the selected item	5	8
6	As an authorized User I want to set a reminder for a selected item so that I am reminded when item is due	8	9
8	As an administrator I want to see the list of accounts on login	2	10
Total		**30**	

Source: https://www.scrum-institute.org/The_Scrum_Product_Backlog.php

Agile Testing

Agile Testing

- A testers on an agile project will work differently than one working on a traditional project.
  - Not casual
  - Highly disciplined

Release and Iteration Planning

- Planning is an ongoing activity
- Two types of planning in agile
  - Release
  - Iteration
- Release and iteration planning should address testing activities

Release Planning

- Looks at the full release
- Defines the product backlog
- Defines the test approach
- Prioritises stories
- Identified risks

Release Planning – Tester Activities

- Define testable user stories, including the acceptance criteria
- Participate in project and quality risk analysis sessions
- Estimate test effort per story
- Define the necessary test levels
- Plan the testing for the release

Iteration Planning

- Concentrates on a single iteration
- Creates iteration backlog with estimates
- Number of stories targeted is based on velocity of team
- Stories are estimated in size
- Stories are divided into tasks
- Tasks are associated with testing tasks

Iteration Planning – Tester Activities

- Participate in detailed risk analysis
- Determine testability of selected stories
- Create acceptance tests for stories
- Create testing tasks and estimate
- Identify functional and non-functional aspects to be tested
- Support and participate in test automation

Test-Related Issues

- What is the proper scope of testing?
- Who will do the testing?
- Do we have the test environment and test data and how will we handle changes?
- Will the order of the code released match the order in which it needs to be tested?
- Are the project and quality risks adequately addressed?

Project Work Products

- Project work products typically fall into three categories:
  - Business
  - Development
  - Test
- Focus is on working software – document only what adds values to the customer

Agile vs. Sequential

- Agile projects are characterised by:
  - Lightweight documentation
  - Less scripted tests and more exploratory
  - Less reporting and metrics
  - Less formality in general
- BA and developer work products:
  - User stories and acceptance criteria are used to derive tests and validate requirements
  - Unit test results help increase test coverage
  - Development documentation helps us understand how the system is built so testing can be better targeted

Acceptance Criteria & Adequate Coverage

- Initial requirements are usually user stories defined at the start of the project
- These can be functional or non-functional requirements
- These are then developed during an iteration

Acceptance Criteria

- To be testable, acceptance criteria should address the following where relevant:
  - Functional behaviour
  - Quality characteristics
  - Scenarios (use cases)
  - Business rules
  - External interfaces
  - Constraints
  - Data definitions

Definition of Done

- This can be defined on a number of dimensions:
  - Test level
  - User story
  - Feature
  - Iteration
  - Release

DoD at Integration Test Level

- All functional requirements tested, including both positive and negative tests, with the number of tests based on size, complexity and risks
- All interfaces between units tested
- All quality risks covered according to the agreed extent of testing
- No unresolved major defects (prioritised according to risk and important)
- All defects found are reported
- All regression tests automated, where possible, with all automated tests stored in a common repository

DoD for a Story

- The user stories selected for the iteration are complete, understood by the team and have detailed, testable acceptance criteria
- All the elements of the user story are specified and reviewed, including the user story acceptance tests, have been completed
- Tasks necessary to implement and test the selected user stories have been identified and estimated by the team

DoD for a Feature

- All constituent user stories, with acceptance criteria, are defined and approved by the customer
- The design is complete, with no known technical debt
- The code is complete, with no known technical debt or unfinished refactoring
- Unit tests have been performed and have achieved the defined level of coverage
- Integration tests and system tests for the feature have been performed according to the defined coverage criteria
- No major defects remain to be corrected
- Feature documentation is complete, which may include release notes, user manuals and on-line help functions

DoD for an Iteration

- All features for the iteration are ready and individually tested according to the feature level criteria
- Any non-critical defects that cannot be fixed within the constraints of the iteration added to the product backlog and prioritised
- Integration of all features for the iteration completed and tested
- Documentation written, reviewed and approved

DoD for a Release

- Coverage: All relevant test basis elements for all contents of the release have been covered by testing.
- Quality: The defect intensity, the defect density, estimated number of remaining defects are within acceptable limits. Outstanding risk is understood and acceptable.
- Time: The pre-determined delivery date has been reached and cost of releasing vs not releasing have been evaluated.
- Cost: ROI, including maintenance has been calculated.

TDD, ATDD, BDD

- Three complementary development techniques
  - Test-driven development
  - Acceptance test-driven development
  - Behaviour-driven development

Test-Driven Development (TDD)

- Add a test that captures the programmer's concept of the desired functioning of a small piece of code
- Run the test, which should fail since the code doesn't exist
- Write the code and run the test in a tight loop until the test passes
- Refactor the code after the test has passed, re-running the test to ensure it continues to pass against the refactored code
- Repeat this process for the next small piece of code, running the previous tests as well as the added tests

Acceptance Test-Driven Development (ATDD)

- ATDD defined acceptance criteria and tests during the creation of user stories
- Collaborative approach that allows every stakeholder to understand how the software component has to behave
- Create reusable tests for regression testing
- Specific tools support creation and execution of such tests, often within the continuous integration process
- Allow quick resolution of defects and validation of feature behaviours. It helps determine if the acceptance criteria are met for the feature.

Behaviour-Driven Development (BDD)

- Given some initial context,
- When and event occurs,
- Then ensure some outcomes.

Risk

- Possibility of a negative or undesirable event occurring
- Expressed as likelihood and impact
- Project and product (quality) risk

Assessing Quality Risks in Agile Projects

- In Agile, risk analysis is carried out during release and iteration planning
- Tasks can be prioritised and visible on task board
- Testing mitigates and finds new quality risks

Risk Analysis – Iteration Planning

- Gather the agile team members together, including tester(s)
- List all the backlog items for the current iteration (e.g., on a task board)
- Identify the quality risks associated with each item, considering all relevant quality characteristics
- Assess each identified risk, determine likelihood and impact
- Determine the extent of testing proportional to the level of risk
- Select the appropriate test technique(s) to mitigate each risk, based on the risk, the level of risk, and the relevant quality characteristic

Collaborative User Story Creation

- Collaborative authorship of the user story can use techniques such as brainstorming and mind mapping
- The tester may use the INVEST technique:
  - Independent
  - Negotiable
  - Valuable
  - Estimable
  - Small
  - Testable

User Story Elements

- Card: the physical media describing a user story
- Conversation: explains how the software will be used
- Confirmation: the acceptance criteria, discussed in the conversation, are used to confirm that the story is done

Estimating Test Effort

- During release planning, the agile team estimate the effort required to complete the release
- A common estimation technique is Planning Poker

Planning Poker

- The product owner or customer reads a story to the estimators
- The estimators discuss the feature and ask questions
- Each estimator selects one of their cards (Fibonacci sequence (0, 1, 2, 3, 5, 8 …) or T-shirt sizes or fingers) indicating their estimate of effort
- All cards are revealed at the same time
- If everyone agrees, that's the estimate
- If there is disagreement, there is more discussion and rules my be applied (use the median, use the highest)

Common Attributes of Agile Methods

- Incremental & Iterative
  - Iterative with short cycles that enable quick verification
  - Use of time-boxing
- Cooperative
  - Emphasis on communication, people, interactions and collaboration
- Adaptive
  - Responds to change, focuses on reducing risk and able to change course as required

Key Difference between Agile and Traditional

- Short iterations delivering working software
- Everyone is responsible for testing (pairing)
- More automation and less scripted testing
- Testing activities occur throughout the iteration
- Hardening or stabilisation iteration occur periodically
- No feature is considered done until it is integrated and tested

What is Different about an Agile Project?

- Everyone is responsible for testing
- Test planning and risk assessment occurs for each iteration
- Test levels vary depending on the iteration content
- Testers often quality coaches
- More automation
- More unscripted experience-based testing
- Lightweight documentation of tests
- Testers are involved earlier
- Testers may pair with developers to help with TDD

Test Levels

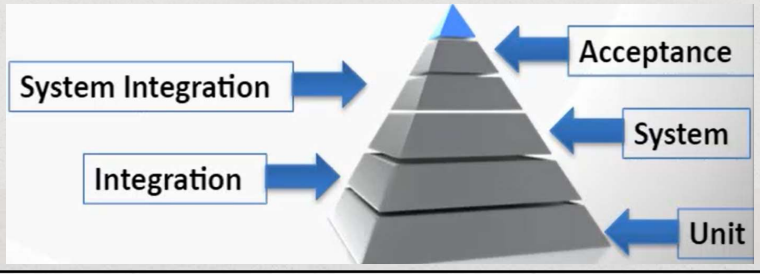
- During an iteration, user stories will progress sequentially through the following activities
  - Unit testing
  - Feature acceptance testing, which is sometimes broken:
    - Feature verification testing
    - Feature validation testing
  - Regression testing occurs throughout the iteration

How does Acceptance Testing Differ?

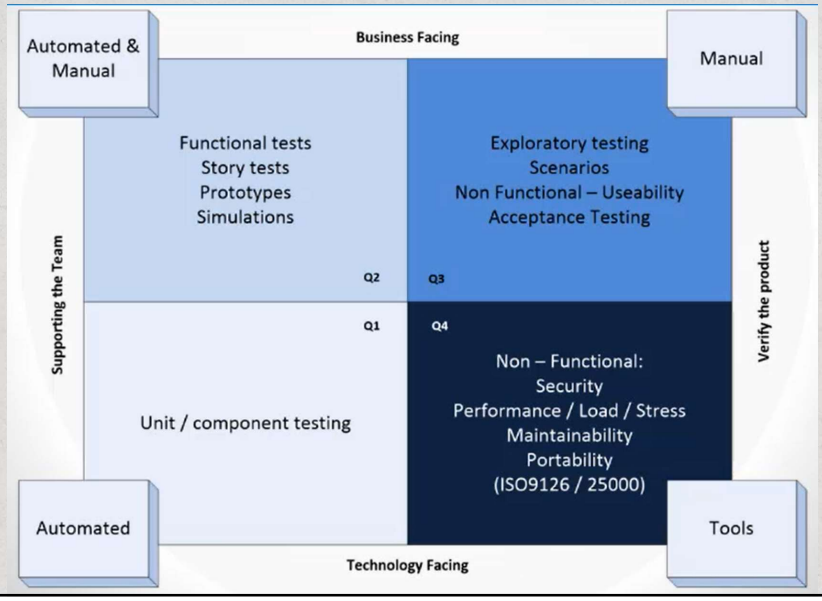
- More frequent, always at the end of an iteration
- Tests could be functional and non-functional
- Demonstration to product owner piece by piece, no full UAT
- Tests may be automated
- There may be a hardening or stabilising iteration

Test Pyramid

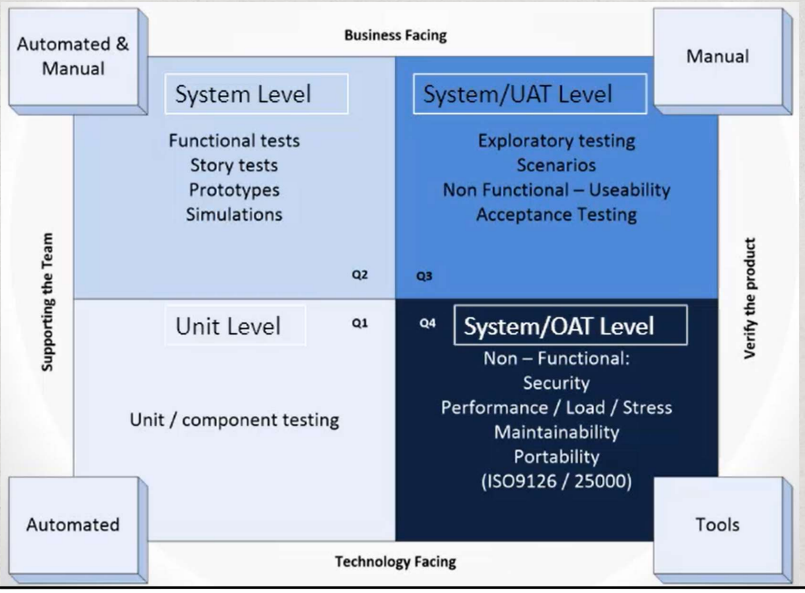
- Based on the concept of early testing
- Software is tested using test levels form the base of the pyramid to the top
- Large number of tests at the lower levels and the number of tests decreases towards the top



Testing Quadrants, Test Levels and Testing Types



Testing Quadrants, Test Levels and Testing Types



Agile Teams

Agile Teams

- Whole-team approach
- Test independence
- Role and skills of a tester in an agile team
- Status of testing in agile projects

Whole-Team Approach

- Everyone who has the knowledge and skills necessary to ensure project success is involved, including customer representative and other business stakeholders
- Teams should be relatively small (3-9)
- Co-location, stand-up meetings
- Promotes more effective and efficient team dynamics

97

Benefits of Whole-Team Approach

- Enhancing communication and collaboration within the team
- Enabling the various skill sets within the team to be leveraged to the benefit of the project
- Making quality everyone's responsibility
- Power of three: work together for all feature discussions
  - Developers
  - Testers
  - Business representatives

98

Independence

- From low to high
  - Developers test own code
  - Testers within development team
  - Testers reporting in different chain
  - Testers from outside the organisation
  - Testing specialists
  - Outsourced testing

Organisational Options

- Testers embedded in agile team
- Fully independent test teams
- Fully independent test teams with testers assigned to agile teams on a long term basis
- Specialist testers available to teams

Early & Frequent Feedback

- Feedback helps the team focus on the features with the highest business value or associated risk
- Agile projects have a specific focus on early and frequent feedback via
  - Short iterations and short term focus
  - Daily stand up meetings
  - Customer involvement form the beginnin

Benefits of Early & Frequent Feedback

- Avoiding requirements misunderstandings
- Clarifying feature requests
- Get the product into the customer's hands
- Discover, isolate and resolve quality problems
- Provide information to the team regarding productivity and ability to deliver
- Promotes consistent project momentum

Role of Tester in Agile Team

- Provide feedback on
  - Test status
  - Test progress
  - Product quality
  - Process quality
- Remember: Each team member if responsible for product quality and this includes testing!

Tester Activities in Agile

- Understanding/implementing strategy
- Measuring and reporting coverage
- Ensure proper use of tools
- Configure test environments
- Reporting defects
- Scheduling testing tasks for iteration/release
- Collaborating to get testable requirements
- Participating in retrospectives

Cautions

- It's easy for a tester to lose their mindset
- Testers can tire of championing good quality practices
- Testers can be overwhelmed with the number of changes in a time-constrained iteration
- A tester can't do it all – the whorl team is needed

Tester Skills

- Curiosity
- Attention to detail
- Professional pessimism
- Critical eye
- Good error guessing
- Communication

Agile Tester Skills

- Test automation
- White-box
- Black-box
- TDD and ATDD
- Experience-based testing

Helpful Skills for Agile Projects

- Be positive!
- Quality-oriented, sceptical thinking
- Talk to stakeholders
- Evaluate test results
- Get testable user stories
- Collaborate

People, Domain & Testing Skills

People	Domain	Testing
Communication	Error Guessing	Curiosity
Be Positive	ATDD	Attention to Detail
Quality oriented	Experience-based testing	Professional Pessimism
Skeptical thinking	Black-box	Critical Eye
Work with stakeholders	Evaluate results	Automation
Collaboration	User stories	White-box
		TDD

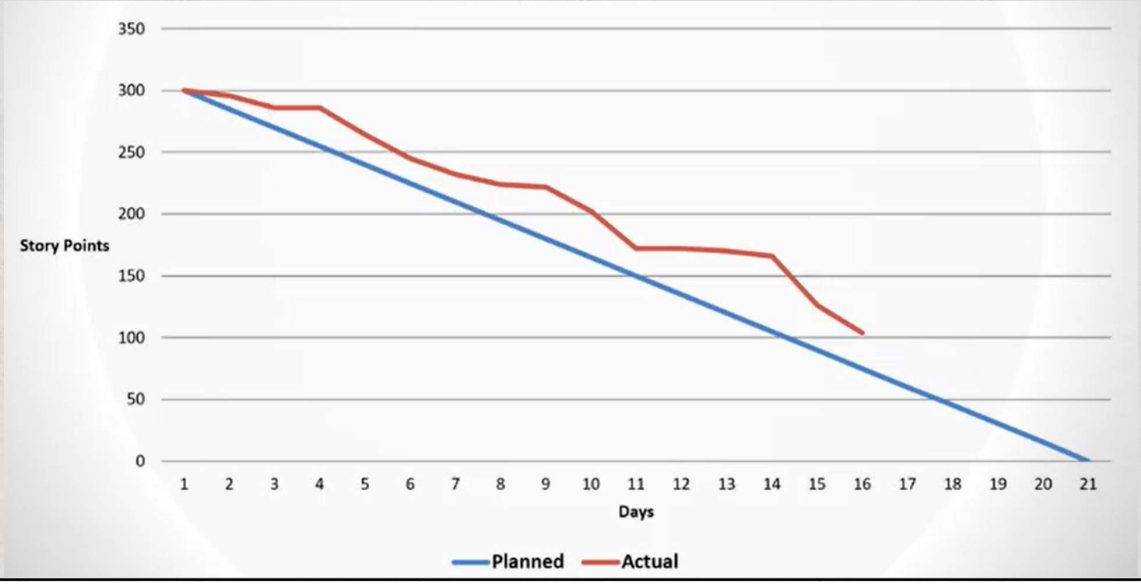
Status of Testing in Agile Projects

- Change takes place rapidly in agile projects
- Testers must devise ways to track progress and make decisions

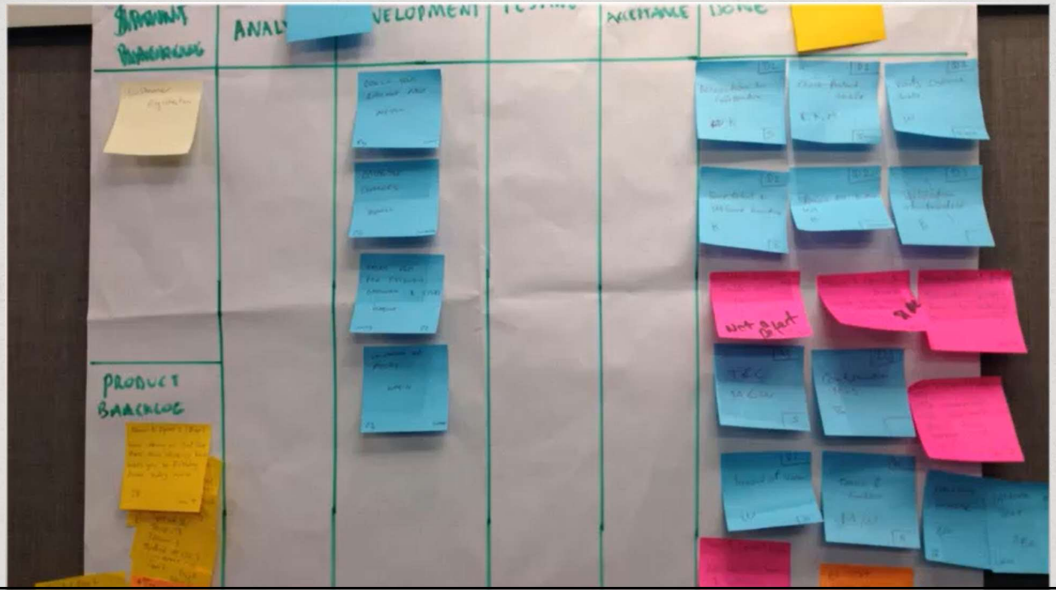
Recording Status and Progress

- Traditional
  - Dashboards
  - Status reports
- Agile
  - Wiki
  - Burndown charts
  - Task boards
  - Stand-up meetings

Burndown Chart



Agile Task Board



Stand-Up Meetings

- Daily, 15 minutes or so
- Everyone's agenda:
  1. What have you completed since the last meeting?
  2. What do you plan to complete by the next meeting?
  3. What is getting in your way?

Efficient Reporting

- Automated reporting into dashboards
  - Test automation progress
  - Test progress against stories
  - Defect status
- Less time spent reporting means more time testing

Benefits

- Automated test results: Free tester time to focus on designing/executing more tests
- Burndown charts: Show the amount of work left to be done vs. the time allocated
- Task boards: Provide instant visual status for whole team
- Stand-ups: Blocking issues are reported and resolved quickly

Tools and Automation

Tools & Automation

- Continuous integration
- Testing and configuration management
- Managing regression risk
- Tools in agile projects

Continuous Integration

- Goal: Reliable, working, integrated software every iteration
- Steps:
  1. Write code
  2. Run static code analysis
  3. Build on dev machine
  4. Unit test
  5. Check-in code for build

Steps Continued

6. Build code
7. Deploy (continuous deployment)
8. Integration test (automated)
9. Build acceptance test (automated)
10. Functional/regression tests (manual and automated)
11. Report results – although results are reported throughout

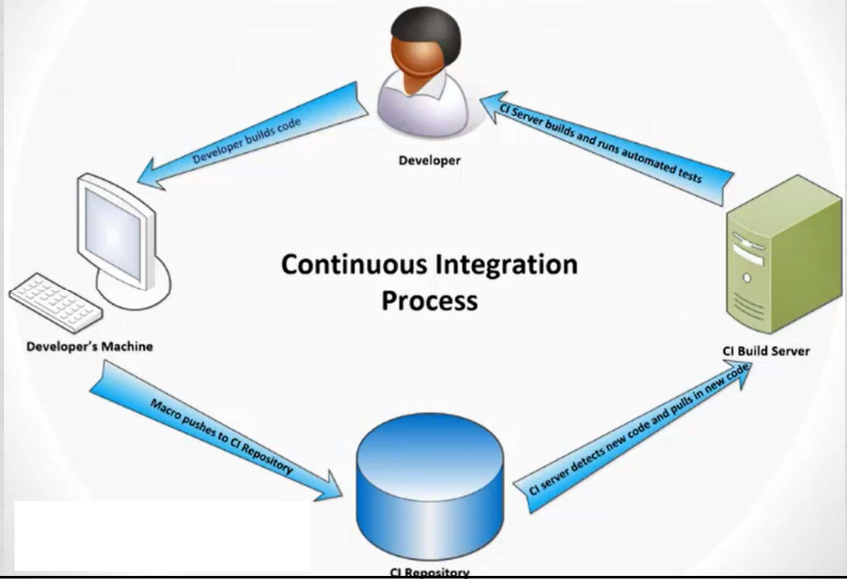
Who Broke the Build???

- If the build breaks or the automation fails, everything stops and the team investigates!

Benefits of Continuous Integration

- Earlier detection of problems
- Regular feedback to the developers
- Version tested is close to version developed
- Regression risk reduced
- Confidence that progress if forward
- Reduces integration risk
- Executable software always available
- Repetitive manual testing reduced
- Quality assessment can be made quickly

Continuous Integration Process



Automation Needed?

- Automated unit tests
- Automated integration tests
- Automated build acceptance tests
- Automated functional tests
- Automated regression tests

Managing Regression Risk

- As each iteration completes, the product grows (and the code churns)
- Risk of introducing defects in agile is high
- Test automation at all test levels is needed as early as possible
- Configuration management tools are required for testware

Out of Date Tests are Useless

- Testes must be reviewed for each iteration
- Test automation must be as consistent investment
- Cull the tests that are no longer applicable
- Good test design practices, such as traceability, are vital

Reasons to Review Regression Tests

- Pesticide paradox
- Impact analysis to determine if the tests meet new acceptance criteria
- Minimise false positives and false negatives
- Target the regression tests to the most failure prone areas
- Reinstate any tests that were commented out to avoid a defect

Tools in Agile Project

- All tool types are also used by agile teams
- Not all tools are used the same way
- Some agile teams opt for an all-inclusive tool having:
  - Task boards
  - Burndown charts
  - User stories

Test Management Tools

- Traditionally:
  - Test Management Tools
  - Requirements Management Tools
  - Defect Tracking Tools
  - Configuration Management Tools

Test Management Tools

- For agile:
  - Physical story/task boards to manage and track stories and tests
- May use application lifecycle management or task management software
  - Record stories and tasks
  - Capture estimates/priorities
  - Provide visual representation of status
  - Integrate with configuration management tools

Communication Tools

- Traditionally:
  - Status reports
  - Emails
  - Verbal reports

Communication Tools

- Wikis:
  - Allow sharing online knowledge base
  - Product feature diagrams
  - Metrics and dashboards
  - Team conversations
- Instant messaging, teleconference, video
  - Real time, direct communication
  - Stand-ups for distributed teams

Communication Tools

- Desktop sharing and capturing tools
  - Product demonstrations
  - Code reviews
  - Pairing
  - Saving demonstrations to wiki

Software Build and Distribution

- Traditionally:
  - Configuration management tools
  - Source control
  - Manual build and deployment

Software Build and Distribution

- Configuration management tools
- Test design, implementation and execution
  - Test design tools (e.g. mind maps)
  - Test case management tools (task trackers, ALM)
  - Test data preparation/load tools
  - Automated test execution (ATDD, BDD)
- Virtualisation and cloud environment tools

Summary

- Agile development
  - Scrum
- Testing in agile
  - TDD, ATDD, BDD; test pyramid, test quadrant
- Agile teams
  - Early and frequent feedback
- Tools and automation
  - CI; burndown chart, task boards

References

- Sommerville, I. *Software Engineering* (Chapter 3)
- International Software Testing Qualifications Board. ISTQB Foundation Level Agile Tester Extension Syllabus

Next

- Quiz
- Assignment 2
  - Due on 26 May 2024, 11.59pm