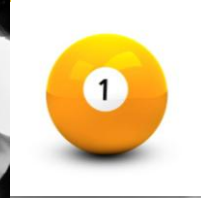




- We create new opportunities



ISTQB® Agile tester

Agile software development

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

The fundamentals of Agile software development

Learning objectives

- FA-1.1.1 Recall the basic concept of Agile software development based on the Agile Manifesto (K1)
- FA-1.1.2 Understand the advantages of the whole-team approach (K2)
- FA-1.1.3 Understand the benefits of early and frequent feedback (K2)

From the ISTQB
Agile Tester syllabus

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

The fundamentals of Agile Agile software development and the Agile manifesto



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Agile manifesto

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Individuals and interactions

- People-centered
- People build software
- Tools or processes might support us, but we cannot rely on them to create efficiency
- To get an efficient team we need continuous communication and interaction



Image courtesy of Stuart Miles / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



Working software

- For a customer, working software is more useful and valuable than overly detailed documentation
- Gives opportunity to provide rapid feedback to development team
- Time-to-market advantage
- Agile is specifically useful when we have:
 - Rapidly changing business environments
 - Unclear problems and/or solutions
 - Innovation in new problem domains



Image courtesy of Master isolated images / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



Customer collaboration

- It is hard for customers to specify their requirements
- Collaboration improves likelihood of understanding customer needs
- Collaboration is more likely to bring success to the project than contracts
- Contracts may however still be important



Image courtesy of photostock / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



Responding to change

- Change is inevitable
- Many factors might have influence on the project and objectives
 - Environment of the business
 - Legislation
 - Competitors
 - Technology advances
 - Etc.
- Must take change into account in the development process
- Flexibility in work practices to embrace change



Image courtesy of renjith krishnan / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



Agile manifesto principles 1-6

- Our highest priority is to satisfy the customer through early and **continuous delivery** of **valuable** software.
- Welcome **changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver **working software frequently**, at intervals of between a few weeks to a few months, with a preference to the shorter timescale.
- Business people and developers must **work together** daily throughout the project.
- Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.

© COPYRIGHT SYSTEM VERIFICATION 2014



Agile manifesto principles 7-12

- **Working software** is the primary measure of progress.
- Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to **technical excellence** and good design enhances agility.
- **Simplicity**--the art of maximising the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from **self-organising teams**.
- At regular intervals, the team **reflects** on how to become more effective, then tunes and **adjusts** its behaviour accordingly.

© COPYRIGHT SYSTEM VERIFICATION 2014



The fundamentals of Agile Whole-team approach



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

The whole-team approach

- Involve everyone with the knowledge and skills needed for the project to succeed
- Include representatives from customer and other business stakeholders
- Team should be relatively small; 3-9 people
- Co-location of the team is preferred
- Supported through daily stand-up meetings
- Promotes effective and efficient team dynamics



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Benefits

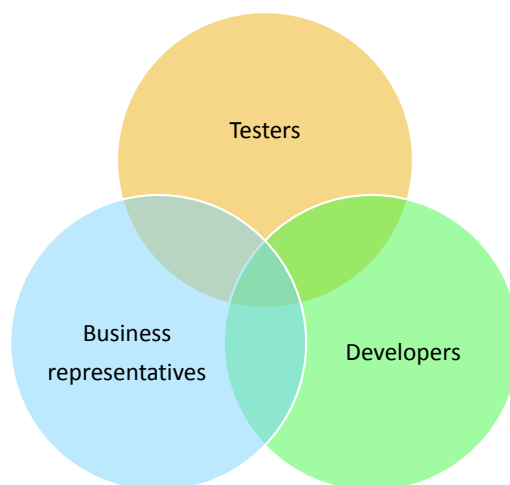
- Enhancing communication and collaboration in the team
- Enabling the various skill sets within the team
- Quality becomes everyone's responsibility



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

The power of three



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

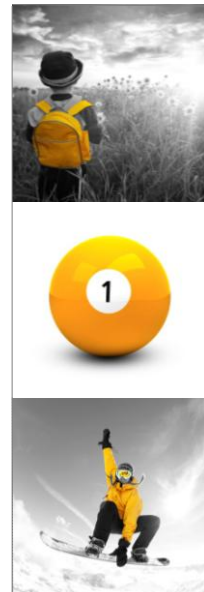
Testers' part

- Work closely with developers and business representatives
- Help business representatives create suitable acceptance tests
- Agree on test strategy and test automation approaches with developers
- Transfer and extend test knowledge to other team members
- Influence the development of the product

© COPYRIGHT SYSTEM VERIFICATION 2014



The fundamentals of Agile Early and frequent feedback

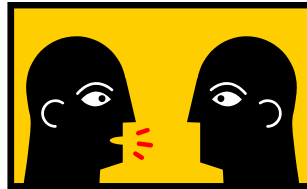


© COPYRIGHT SYSTEM VERIFICATION 2014



Early and frequent feedback

- Enabled by short iterations
- In sequential development customer does not see the product until late in the development
 - Often too late to address the issues effectively
- Enables the team to incorporate changes into the development process
- Helps team to focus on features with highest business value or risk
- The team's capability becomes transparent
- Continuous integration can provide rapid feedback



© COPYRIGHT SYSTEM VERIFICATION 2014



Benefits

- Avoiding requirement misunderstandings
- Clarifying customer feature requests by making them available to customers early
- Discovering, isolating and resolving (quality) problems early
- Providing information to the team about productivity and ability to deliver
- Consistent project momentum

© COPYRIGHT SYSTEM VERIFICATION 2014



Aspects of Agile approaches

Learning objectives

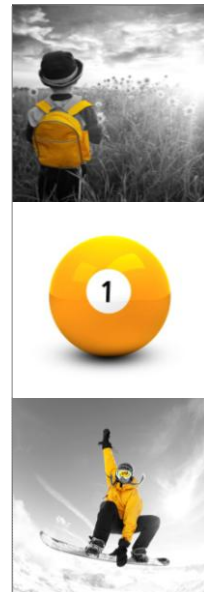
- FA-1.2.1 Recall Agile software development approaches (K1)
- FA-1.2.2 Write testable user stories in collaboration with developers and business representatives (K3)
- FA-1.2.3 Understand how retrospectives can be used as a mechanism for process improvement in Agile projects (K2)
- FA-1.2.4 Understand the use and purpose of continuous integration (K1)
- FA-1.2.5 Know the differences between iteration and release planning, and how a tester adds value in each of these activities (K1)

**From the ISTQB
Agile Tester syllabus**

© COPYRIGHT SYSTEM VERIFICATION 2014

**SYSTEM
VERIFICATION**

Aspects of Agile approaches Agile software development approaches



© COPYRIGHT SYSTEM VERIFICATION 2014

**SYSTEM
VERIFICATION**

Common practices in Agile

- Planning for overall release
- Planning for each iteration
- Collaborative user story creation
- Continuous integration
- Retrospectives



Image courtesy of Ponsulak/ FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Agile software development approaches

- Agile approaches implement values and principles of the Agile manifesto
- Commonly used
 - Extreme programming (XP)
 - Scrum
 - Kanban



Image courtesy of David Castillo Dominici / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

XP

- eXtreme Programming
- One of the first Agile methods to become widely known
- Introduced by Kent Beck
- Can be used together with other models, such as Scrum or Kanban
- A collection of values, principles and development practices
- Influences other Agile software development methods
- Two versions, XP1 (1998) and XP2 (2004)



Image courtesy of Gualberto107/ FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

XP values to guide development

- Communication
- Simplicity
- Feedback
- Courage
- Respect



Image courtesy of creativedoxfoto / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

XP principles

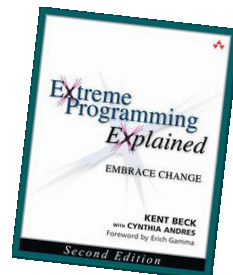
- Humanity
- Economics
- Mutual benefit
- Self similarity
- Improvement
- Diversity
- Reflection
- Flow
- Opportunity
- Redundancy
- Failure
- Quality
- Baby steps
- Accepted responsibility

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

XP practices

- Sit together
- Whole team
- Informative workspace
- Energised work
- Pair programming
- Stories
- Weekly cycle
- Quarterly cycle
- Slack
- Ten-minute build
- Continuous integration
- Test first programming
- Incremental design



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Scrum

- Lightweight model
- Iterative
- Incremental
- Scrum team plays central role
- Can be seen as a management framework
- Must be filled with “how to”
- Easy to understand - Hard to master
- Does not dictate software development techniques



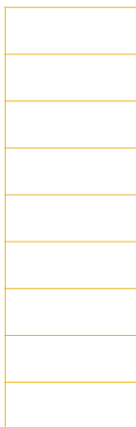
Image courtesy of Stuart Miles / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

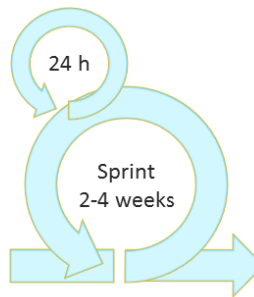
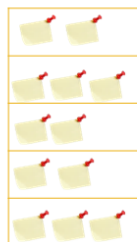


Scrum

Product backlog



Sprint backlog



Potentially shippable product increment



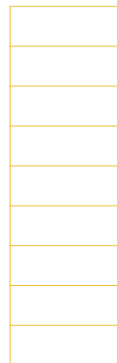
© COPYRIGHT SYSTEM VERIFICATION 2014



Product and sprint backlog

- Product backlog
 - High-level requirements
 - Often user stories
 - Prioritised / Ordered
 - Also known as release backlog
- Sprint backlog
 - User stories for a sprint
 - Broken down to tasks

Product backlog



Sprint backlog



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM VERIFICATION

Definition of done

- Can be made for one story or as a general checklist
- Most often used for sprint completion
- Can also use several definition of done as tollgate/checklist for moving tasks on at the Scrum board

STORY	TO DO	IN DEV	IN TEST	DONE
As a receptionist I want to book conference rooms for guests				
As a receptionist...	 			

Definition of done (example)








- ☒ Check in code
- ☒ Run automatic regression test suite successfully
- ☒ Run integration test successfully
- ☒ Run automatic unit tests successfully
- ☒ Make DB migration scripts
- ☒ All code has been reviewed by another person
- ☒ All acceptance criteria are fulfilled
- ☒ Status in backlog-tool is updated
- ☒ Deployed to demo environment

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM VERIFICATION

Timeboxing in a sprint

- A sprint can contain no more than the items the team expects to finish
- Unfinished tasks moves the related story back to product backlog
- Other things which can be timeboxed:
 - Tasks
 - Meetings
 - Investigations
 - ...

Sprint backlog	
US	Tasks
US1	  
US2	 
US3	 

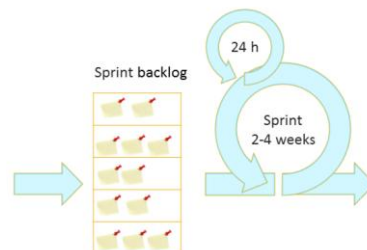
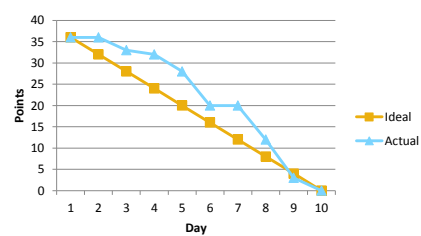
© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Transparency through daily Scrum

- Three questions for each team member
 - What did you do yesterday?
 - What will you do today?
 - Is there any obstacle which obstructs your work?
- 15 minutes
- Any one who wants can attend
- Only the team should speak
- Short and effective

Burndown chart

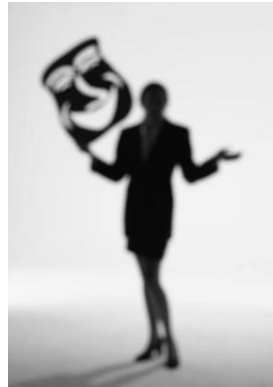


© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Roles in Scrum

- Scrum master
 - Guards the practices and rules of Scrum
 - Resolves impediments
 - Not team lead, but coach
- Product owner
 - Represents customer(s)
 - Owns and prioritises product backlog
 - Not team lead
- Development team
 - Develops and tests the product
 - Self-organised
 - Cross-functional
 - Make decisions



© COPYRIGHT SYSTEM VERIFICATION 2014



Kanban





















- Management approach
- Visualise and optimise flow within a value-added chain
- Three instruments
 - Kanban board
 - Work-in-progress limit (WIP)
 - Lead time
- Optional iterations
- Similarities to Scrum
 - Visualising active tasks
 - Backlog for unscheduled tasks
- Enables item by item release



© COPYRIGHT SYSTEM VERIFICATION 2014



Kanban board

Story Backlog	Requirements (4)		Development (5)		System test (3)		Acceptance test (2)		To be deployed
	In progress	Done	In progress	Done	In progress	Done	In progress	Done	
									
									
									
									

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

WIP limit

- Limits the amount of simultaneous work
- Purpose is to get items through the value-added-chain
- Used to find a healthy production rate



Image courtesy of mapichai / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Kanban – what is not included?

- Roles
- Phases
- Artefacts
- Meetings or events
- Iterations
- Releases
- Team
- Prioritisation

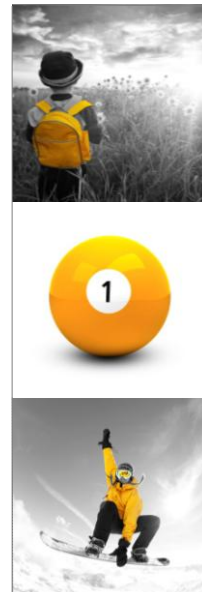


Image courtesy of 2nix / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Aspects of Agile approaches Collaborative user story creation



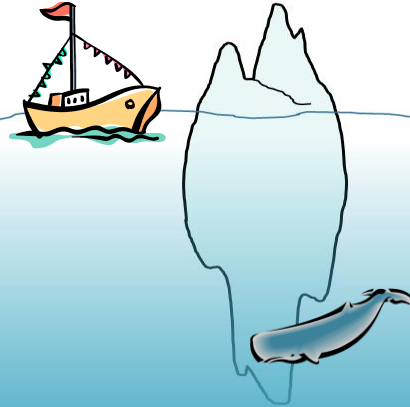
© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Requirement problems

Poor specifications are often the major reason for project failure

- Users lack insight of true needs
- No global vision for the system
- Redundant or contradicting features
- Other miscommunications



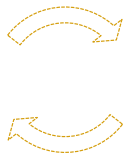
© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Shared vision of a feature is accomplished

Agile

- Frequent informal reviews while requirements are being written
- User stories capture perspectives from
 - Developers
 - Testers
 - Business representatives



Sequential

- Formal reviews after requirements are written
- The team is not involved in producing requirements



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

User stories

- Each user story should cover
 - Functional characteristics
 - Non-functional characteristics
 - Acceptance criteria
- Collaboration
 - Business representatives
 - Developers
 - Testers
- Documentation
 - Concise, sufficient and necessary



Gives testers and developers a greater understanding - and lets the business representatives validate



Image courtesy of Stockimages/ FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



The tester adding value to the user story

- Unique perspective
- Identifying missing details
- Identifying missing non-functional requirements
- Asking open-ended questions
- Proposing ways to test
- Confirming acceptance criteria

INVEST technique

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

© COPYRIGHT SYSTEM VERIFICATION 2014



3C concept

- Card
- Conversation
- Confirmation



Image courtesy of Stuart Miles, Ambro and Cooldesign/ FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

User story examples

- As an Admin I want to update permissions for a user so that I can restrict certain users to log in
- As a Customer I want to be able to filter the list of Computers so that I can see all Computers from a specific vendor
- As an Account owner I want to be able to withdraw money from my account so that I can use them to buy milk at the store



Image courtesy of Hin255/FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Exercise 1

Collaborative user story creation

(K3) Write testable user stories in collaboration with developers and business representatives



Use the case description (to be handed out) and analyse and work with the user stories

- How can they be improved?

Exercise 1

Collaborative user story creation

(K3) Write testable user stories in collaboration with developers and business representatives

Each user story should cover

- Functional characteristics
- Non-functional characteristics
- Acceptance criteria

INVEST technique

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Aspects of Agile approaches

Retrospectives



© COPYRIGHT SYSTEM VERIFICATION 2014



What are retrospectives?

- Meeting held at the end of each iteration
- Discusses
 - What was successful
 - What could be improved
 - How to incorporate improvements
 - How to retain success in future iterations
- Covers
 - Process
 - People
 - Organisations
 - Relationships
 - Tools
- Regular retrospectives with follow up activities are critical to self-organisation and continuous improvement

© COPYRIGHT SYSTEM VERIFICATION 2014



Result of the retrospective meeting

Improvements may concern:

- Test effectiveness
- Test productivity
- Test case quality
- Team satisfaction
- Testability of
 - Applications
 - User stories
 - Features
 - System interfaces



Root cause analysis of defects can drive test and development improvements

Continuous improvement must be done at a sustainable pace – only a few improvements per iteration

Image courtesy of 2nd/FredigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

How and when to do retrospectives

- When and how depends on Agile method
- Attendants
 - Business representatives
 - Team
 - Facilitator organises and runs meeting
 - Sometimes the team invite other participants
- The testing part
 - Testers bring unique perspective
 - Testing vitally contributes to success in each sprint
 - All team members (testers as well as non-testers) can provide input on testing and non-testing activities



Professional environment and mutual trust is essential



Image courtesy of colldesign/FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Are the success factors for reviews from Foundation applicable?

- Clear predefined objective
- Right people
- Testers are valued reviewers who contribute
- Defects found are welcomed and expressed objectively
- People issues and psychological aspects are dealt with
- Atmosphere of trust, outcome will not be used to evaluate participants
- Suitable review techniques
- To increase effectiveness checklist or roles are used, if applicable
- Training is given in review techniques, especially more formal ones
- Management supports a good review process
- Emphasis on learning and process improvement

© COPYRIGHT SYSTEM VERIFICATION 2014



Are the success factors for reviews from Foundation applicable?

- Clear predefined objective
- Right people
- Testers are valued reviewers who contribute
- Defects found are welcomed and expressed objectively
- People issues and psychological aspects are dealt with
- Atmosphere of trust, outcome will not be used to evaluate participants
- Suitable review techniques
- To increase effectiveness checklist or roles are used, if applicable
- Training is given in review techniques, especially more formal ones
- Management supports a good review process
- Emphasis on learning and process improvement

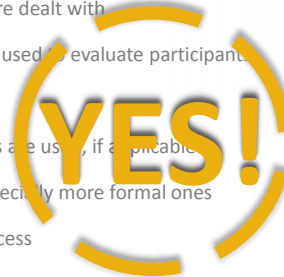


Image courtesy of photostock/FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



Aspects of Agile approaches

Continuous integration

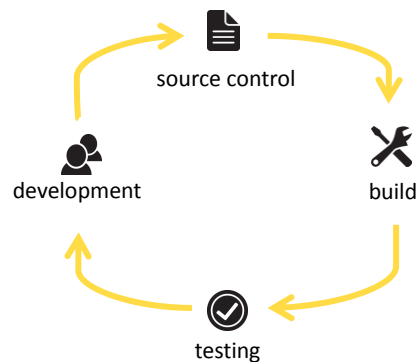


© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

What is continuous integration?

- Instead of building only now and then
- When checking in code, builds automatically
- Merging all changes made to the software and integrating
- Wrapped into a single, automated, repeatable process



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Continuous integration

Advantages

- Able to deliver every sprint
- Integrating components often
- Finding root causes when they are added
- Helps build large integrated systems, one change at a time
- Stable foundation, easier refactoring
- Executable software available for test, demo, education

CI steps:

1. Static code analysis
2. Compile
3. Unit test
4. Deploy
5. Integration test
6. Report

© COPYRIGHT SYSTEM VERIFICATION 2014



Continuous integration

Automatic analysis, unit and integration tests

- Continuous quality control
- Running automated tests at least every day
- Automated running tests free testers to manual test of new features, changes, bugs
- Detecting errors earlier, find regressions when change is introduced

CI steps:

1. **Static code analysis**
2. Compile
3. **Unit test**
4. Deploy
5. **Integration test**
6. Report

© COPYRIGHT SYSTEM VERIFICATION 2014



Continuous integration

Deployment

- Reduces the error rate of manual deploy
- Less delays than waiting for specialised staff or developers to install releases
- Deployment can be on development, test, staging environment
- Can even automatically deploy to production, if quality ok

CI steps:

1. Static code analysis
2. Compile
3. Unit test
- 4. Deploy**
5. Integration test
6. Report

© COPYRIGHT SYSTEM VERIFICATION 2014



Continuous integration

Automatic CI report

- Quick feedback to the team on quality
- Makes progress of product visible

CI steps:

1. Static code analysis
2. Compile
3. Unit test
4. Deploy
5. Integration test
- 6. Report**

© COPYRIGHT SYSTEM VERIFICATION 2014



Continuous integration

Risks and challenges

- Effort to get, install, maintain the tools
- Decide process for continuous integration
- Effort to create and maintain test automation
- Need thorough test coverage to trust the test results
- May rely too much on unit tests instead of system and acceptance testing

CI steps:

1. Static code analysis
2. Compile
3. Unit test
4. Deploy
5. Integration test
6. Report

© COPYRIGHT SYSTEM VERIFICATION 2014



Aspects of Agile approaches Release and iteration planning



© COPYRIGHT SYSTEM VERIFICATION 2014



Agile planning

- Planning is an ongoing activity
- Release planning
- Iteration planning



Image courtesy of cooldesign / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Release planning

- Looks ahead to the release of the product
- Done some time before project start
- Defines and redefines the product backlog
- High level
- Defines the basis for a test approach and a test plan spanning all iterations
- Collaboration between business representatives and the team

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Release planning

- Establishment and prioritisation of user stories
- May involve refining larger stories into a set of smaller stories
- Collaboration between business representatives and the team
- Identification of project and quality risks
- High level effort estimation



Image courtesy of Stuart Miles / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



The tester adding value in release planning

- Defining testable user stories, including acceptance criteria
- Participating in project and quality risk analysis
- Estimating testing effort associated with the user stories
- Defining the necessary test levels
- Planning for testing for the release



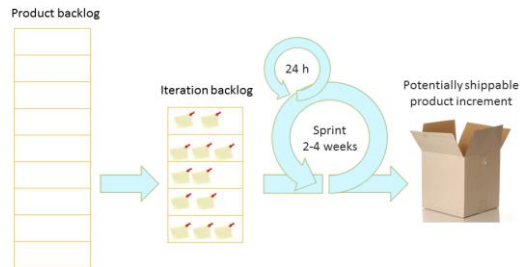
Image courtesy of renjith krishnan / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



Iteration planning

- Looks ahead to the end of a single iteration
- Iteration backlog
- Too vague user stories can be rejected by the team
- Stories are picked in priority order
- Business answers questions about the stories
- Number of stories selected is based on
 - Team velocity
 - Size of the selected user stories



© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Iteration planning

- Team selects user stories from release backlog
- Elaborate user stories
- Risk analysis on user stories
- Estimate work needed for each user story
 - Team's estimates should cover both test and development
- After contents of iteration backlog is clear and established it is time for tasking

US/item in Iteration backlog

As a receptionist, I want to...	3
As a receptionist, I want to...	5
As a receptionist, I want to...	2

Tasks for US/item

Code UI	2
Code Business layer	3
Integrate with Service DB	3
Exploratory testing	3
Write tests	2
Automate tests	3

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

The tester adding value in iteration planning

- Detailed risk analysis
- Determining testability
- Creating acceptance tests
- Tasking (esp. testing tasks)
- Estimating testing effort for testing tasks
- Identifying functional and non-functional aspects to test
- Supporting and participating in test automation at multiple levels of testing



Image courtesy of renjith krishnan / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

Changing plans

"In preparing for battle I have always found that plans are useless, but planning is indispensable."
– Dwight D. Eisenhower

- Changes will come
- Changing release plans
- Individual user stories change
- Iteration plan changes during iteration
 - Simple story proves to be complex

Triggered by

- Delivery capabilities
- Velocity
- Technical issues
- Discovery of new markets
- New opportunities
- New competitors
- Business threats

© COPYRIGHT SYSTEM VERIFICATION 2014

SYSTEM
VERIFICATION

The testers dilemma

- Difficulties with changes
 - Test planning requires understanding the release
 - Adequate test basis
 - Adequate test oracle
- Required information must be available early
- Change must be embraced
- What to do?



Image courtesy of isolated images / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014



Plan for test during release and iteration planning

- Scope of testing (and reason)
- Extent of testing for areas in scope (and reason)
- Test goals
- Who will carry out the activities
- Test environment and test data
- Timing, sequences, dependencies, prerequisites
- Project and quality risks to be addressed



Image courtesy of David Castillo / FreeDigitalPhotos.net

© COPYRIGHT SYSTEM VERIFICATION 2014

