Agile Models

What is Agile Software Development?

- Agile: Easily moved, light, nimble, active software processes
- How agility achieved?
  - Fitting the process to the project
  - Avoidance of things that waste time

Agile Model

- To overcome the shortcomings of the waterfall model of development.
  - Proposed in mid-1990s
- The agile model was primarily designed:
  - To help projects to adapt to change requests
- In the agile model:
  - The requirements are decomposed into many small incremental parts that can be developed over one to four weeks each.

Ideology: Agile Manifesto

- Individuals and interactions *over*
  - process and tools

http://www.agilemanifesto.org

- Working Software *over*
  - comprehensive documentation
- Customer collaboration *over*
  - contract negotiation
- Responding to change *over*
  - following a plan

Agile Methodologies

- XP
- Scrum
- Unified process
- Crystal
- DSDM
- Lean

Agile Model: Principal Techniques

- **User stories:**
  - Simpler than use cases.
- **Metaphors:**
  - Based on user stories, developers propose a common vision of what is required.
- **Spike:**
  - Simple program to explore potential solutions.
- **Refactor:**
  - Restructure code without affecting behavior, improve efficiency, structure, etc.

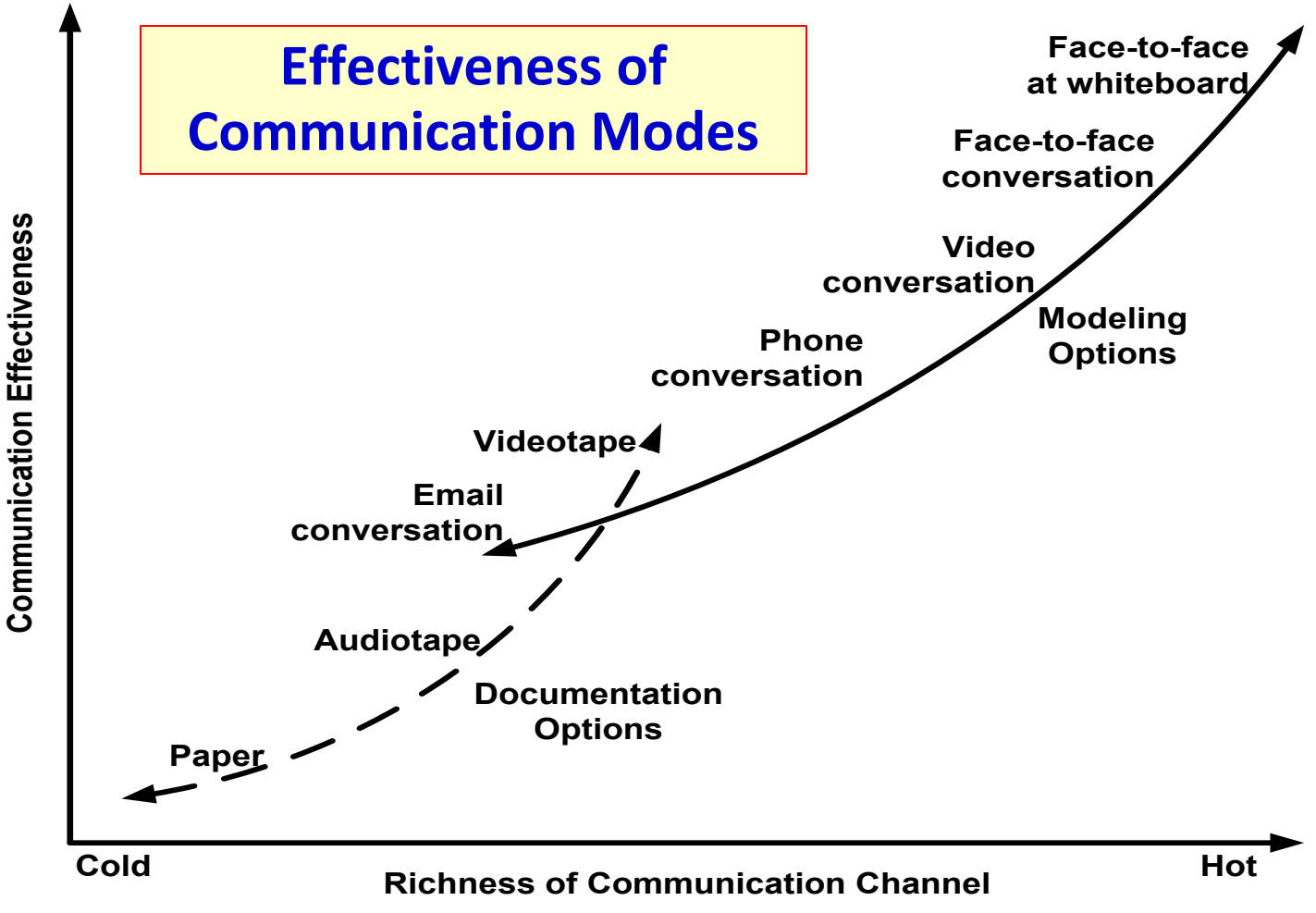
- At a time, only one increment is planned, developed, deployed at the customer site.
  - **No long-term plans are made.**
- An iteration may not add significant functionality,

Agile Model: Nitty Gritty

  - But still a new release is invariably made at the end of each iteration
  - Delivered to the customer for regular use.

Methodology

- **Face-to-face communication favoured over written documents.**
- To facilitate face-to-face communication,
  - Development team to share a single office space.
  - Team size is deliberately kept small (5-9 people)
  - This makes the agile model most suited to the development of small projects.



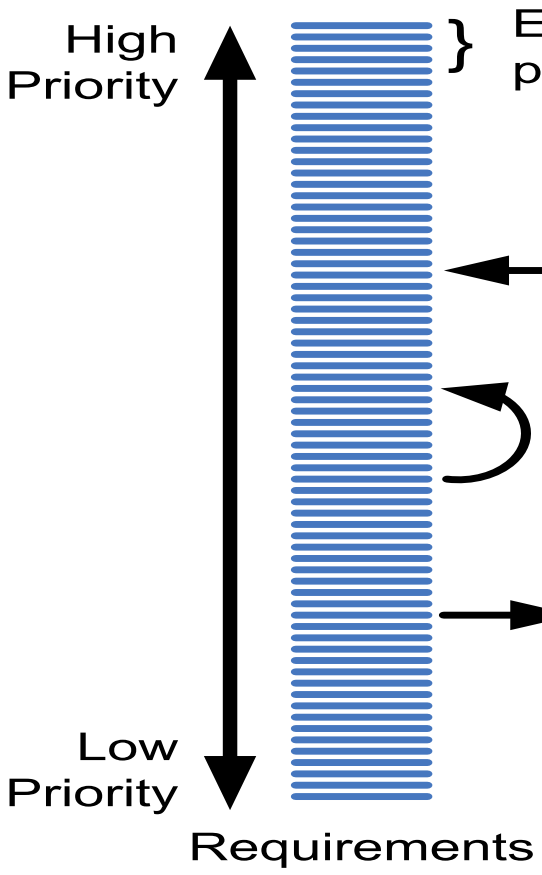
Agile Model: Principles

- The primary measure of progress:
  - **Incremental release of working software**
- Important principles behind agile model:
  - Frequent delivery of versions --- once every few weeks.
  - Requirements change requests are easily accommodated.
  - Close cooperation between customers and developers.
  - Face-to-face communication among team members.

Agile Documentation

- Travel light:
  - You need far less documentation than you think.
- Agile documents:
  - Are concise
  - Describe information that is less likely to change
  - Describe “good things to know”
  - Are sufficiently accurate, consistent, and detailed
- Valid reasons to document:
  - Project stakeholders require it
  - To define a contract model
  - To support communication with an external group
  - To think something through

Agile Software Requirements Management



Each iteration implement the highest-priority requirements

Each new requirement is prioritized and added to the stack

Requirements may be reprioritized at any time

Requirements may be removed at any time

Adoption Detractors

- Sketchy definitions, make it possible to have
  - Inconsistent and diverse definitions
- High quality people skills required
- Short iterations inhibit long-term perspective
- Higher risks due to feature creep:
  - Harder to manage feature creep and customer expectations

Agile Model Shortcomings

- Derives agility through developing tacit knowledge within the team, rather than any formal document:
  - Can be misinterpreted…
  - External review difficult to get…
  - When project is complete, and team disperses, maintenance becomes difficult…

Agile Model versus Waterfall Model

- Steps of Waterfall model are a planned sequence:
  - Requirements-capture, analysis, design, coding, and testing .
- Progress is measured in terms of delivered artefacts:
  - Requirement specifications, design documents, test plans, code reviews, etc.
- In contrast agile model sequences:
  - Delivery of working versions of a product in several increments.

Agile Model versus Iterative Waterfall Model

- As regards to similarity:
  - We can say that Agile teams use the waterfall model on a small scale.

Agile versus RAD Model

- Agile model does not recommend developing prototypes:
  - Systematic development of each incremental feature is emphasized.
- In contrast:
  - RAD is based on designing quick-and-dirty prototypes, which are then refined into production quality code.

Agile versus exploratory programming

- Similarity:
  - Frequent re-evaluation of plans,
  - Emphasis on face-to-face communication,
  - Relatively sparse use of documents.
- Agile teams, however, do follow defined and disciplined processes and carry out rigorous designs:
  - This is in contrast to chaotic coding in exploratory programming.

Extreme Programming (XP)

Extreme Programming Model

- Extreme programming (XP) was proposed by Kent Beck in 1999.
- The methodology got its name from the fact that:
  - **Recommends taking the best practices to extreme levels.**
  - If something is good, why not do it all the time.

Taking Good Practices to Extreme

- **If code review is good:**
  - Always review --- **pair programming**
- **If testing is good:**
  - Continually write and execute test cases --- **test-driven development**
- **If incremental development is good:**
  - Come up with new increments every few days
- **If simplicity is good:**
  - Create the simplest design that will support only the currently required functionality.

Taking to Extreme

- **If design is good,**
  - everybody will design daily (refactoring)
- **If architecture is important,**
  - everybody will work at defining and refining the architecture (metaphor)
- **If integration testing is important,**
  - build and integrate test several times a day (continuous integration)

4 Values

- **Communication**:
  - Enhance communication among team members and with the customers.
- **Simplicity**:
  - Build something simple that will work today rather than something that takes time and yet never used
  - May not pay attention for tomorrow
- **Feedback**:
  - System staying out of users is trouble waiting to happen
- **Courage**:
  - Don’t hesitate to discard code

Best Practices

- **Coding**:
  - without code it is not possible to have a working system.
  - Utmost attention needs to be placed on coding.
- **Testing**:
  - Testing is the primary means for developing a fault-free product.
- **Listening**:
  - Careful listening to the customers is essential to develop a good quality product.

Best Practices

- **Designing**:
  - Without proper design, a system implementation becomes too complex
  - The dependencies within the system become too numerous to comprehend.
- **Feedback**:
  - Feedback is important in learning customer requirements.

Extreme Programming Activities

- **XP Planning**
  - Begins with the creation of “user stories”
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
- **XP Design**
  - Follows the KIS principle
  - Encourage the use of CRC cards
  - For difficult design problems, suggests the creation of “spike solutions”—a design prototype
  - Encourages “refactoring”—an iterative refinement of the internal program design

Extreme Programming Activities

- **XP Coding**
  - Recommends the construction of unit test cases *before* coding commences (test-driven development)
  - Encourages “pair programming”
- **XP Testing**
  - All unit tests are executed daily
  - “Acceptance tests” are defined by the customer and executed to assess customer visible functionalities

Full List of XP Practices

1. **Planning** – determine scope of the next release by combining business priorities and technical estimates
2. **Small releases** – put a simple system into production, then release new versions in very short cycles
3. **Metaphor** – all development is guided by a simple shared story of how the whole system works
4. **Simple design** – system is to be designed as simple as possible
5. **Testing** – programmers continuously write and execute unit tests

Full List of XP Practices

7. **Refactoring** – programmers continuously restructure the system without changing its behavior to remove duplication and simplify

8. **Pair-programming** -- all production code is written with two programmers at one machine

9. **Collective ownership** – anyone can change any code anywhere in the system at any time.

10. **Continuous integration** – integrate and build the system many times a day – every time a task is completed.

Full List of XP Practices

11. **40-hour week** – work no more than 40 hours a week as a rule
12. **On-site customer** – a user is a part of the team and available full-time to answer questions
13. **Coding standards** – programmers write all code in accordance with rules emphasizing communication through the code

Emphasizes Test-Driven Development (TDD)

- Based on user story develop test cases
- Implement a quick and dirty feature every couple of days:
  - **Get customer feedback**
  - **Alter if necessary**
  - **Refactor**
- Take up next feature

Project Characteristics that Suggest Suitability of Extreme Programming

- Projects involving new technology or research projects.
  - In this case, the requirements change rapidly and unforeseen technical problems need to be resolved.
- Small projects:
  - These are easily developed using extreme programming.

Life Cycle Models: Scrum

Practice Questions

- What are the stages of iterative waterfall model?
- What are the disadvantages of the iterative waterfall model?
- Why has agile model become so popular?
- What difficulties might be faced if no life cycle model is followed for a certain large project?

Suggest Suitable Life Cycle Model

- A software for an academic institution to automate its:
  - Course registration and grading
  - Fee collection
  - Staff salary
  - Purchase and store inventory
- The software would be developed by tailoring a similar software that was developed for another educational institution:
  - 70% reuse
  - 10% new code and 20% modification

Practice Questions

- Which types of risks can be better handled using the spiral model compared to the prototyping model?
- Which type of process model is suitable for the following projects:
  - A customization software
  - A payroll software for contract employees that would be add on to an existing payroll software

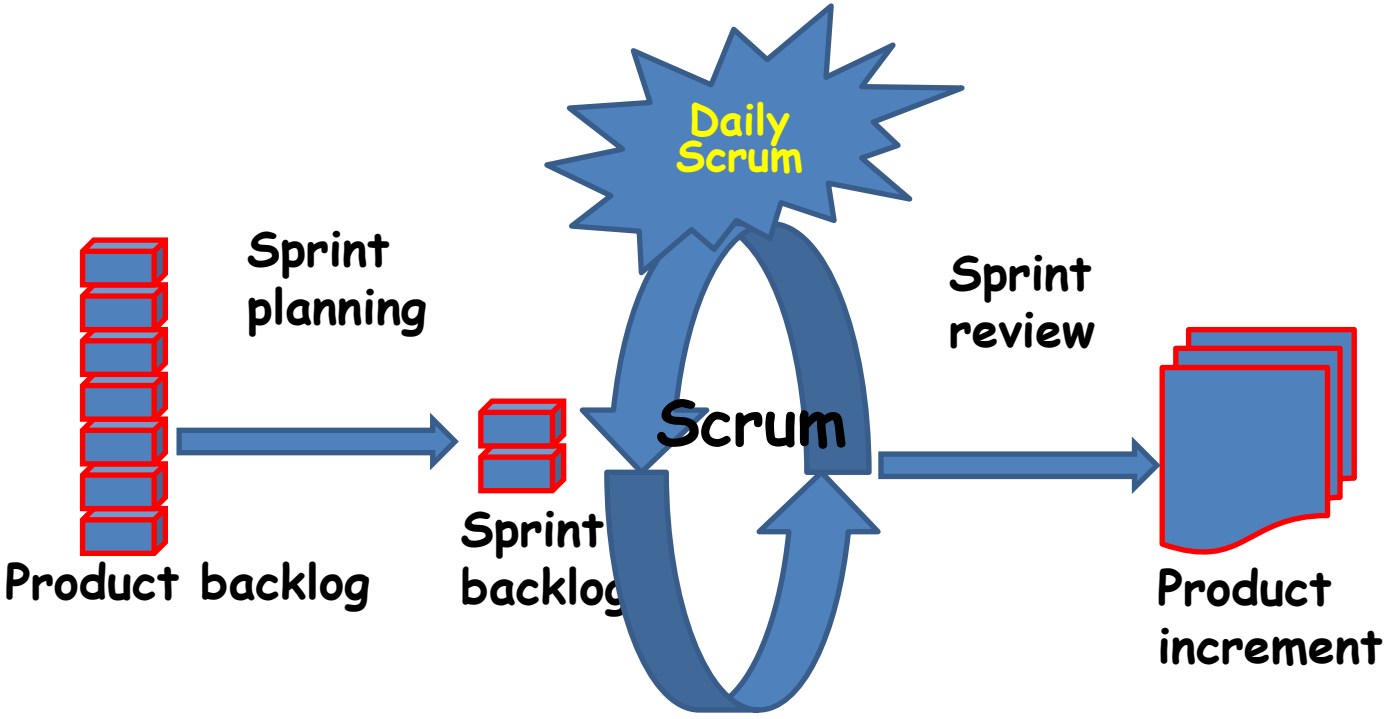
Practice Questions

- Which lifecycle model would you select for the following project which has been awarded to us by a mobile phone vendor:
  - A new mobile operating system by upgrading the existing operating system
  - Needs to work well efficiently with 4G systems
  - Power usage minimization
  - Directly upload backup data on a cloud infrastructure maintained by the mobile phone vendor

Scrum

Scrum: Characteristics

- Self-organizing teams
- Product progresses in a series of month-long **sprints**
- Requirements are captured as items in a list of **product backlog**
- One of the agile processes


Daily
Scrum
Sprint
planning
Sprint
review
Scrum
Product backlog
Sprint
backlog
Product
increment

Sprint

- Scrum projects progress in a series of "sprints"
  - Analogous to XP iterations or time boxes
  - Target duration is one month
- **Software increment is designed, coded, and tested during the sprint**
- **No changes entertained during a sprint**

Scrum Framework

- **Roles :** Product Owner, Scrum Master, Team
- **Ceremonies :** Sprint Planning, Sprint Review, Sprint Retrospective, and Daily Scrum Meeting
- **Artifacts :** Product Backlog, Sprint Backlog, and Burndown Chart

Key Roles and Responsibilities in a Scrum Team

- **Product Owner**
  - Represents customers' views and interests.
- **Development Team**
  - Team of five-nine people with cross-functional skill sets.
- **Scrum Master (aka Project Manager)**
  - Facilitates scrum process and resolves impediments at the team and organization level by acting as a buffer between the team and outside interference.

Product Owner

- Defines the features of the product
- Decides on release date and content
- Prioritizes features according to usefullness
- Adjusts features and priority every iteration, as needed
- Accepts or reject work results.

The Scrum Master

- Represents management in the project
- Removes impediments
- Ensures that the team is fully functional and productive
- Enables close cooperation across all roles and functions
- Shields the team from external interferences

Scrum Team

- Typically 5-10 people
- Cross-functional
  - QA, Programmers, UI Designers, etc.
- Teams are self-organizing
- Membership can change only between sprints

Sprint

- Fundamental process flow of Scrum
- It is usually a month-long iteration:
  - during this time an incremental product functionality completed
- NO outside influence allowed to interfere with the Scrum team during the Sprint
- Each day begins with the Daily Scrum Meeting

Ceremonies

- Sprint Planning Meeting
- Daily Scrum
- Sprint Review Meeting

Sprint Planning

- Goal is to produce Sprint Backlog
- Product owner works with the Team to negotiate what Backlog Items
- Scrum Master ensures Team agrees to realistic goals

Daily Scrum

- Daily
- 15-minutes
- Stand-up meeting
- Not for problem solving
- Three questions:
  1. What did you do yesterday
  2. What will you do today?
  3. What obstacles are in your way?

Daily Scrum

- Is NOT a problem solving session
- Is NOT a way to collect information about WHO is behind the schedule
- Is a meeting in which team members review what is done and make informal commitments to each other and to the Scrum Master
- Is a good way for a Scrum Master to track the progress of the Team

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features
- Informal
  - 2-hour prep time rule
- Participants
  - Customers
  - Management
  - Product Owner
  - Other team members

Sprint Review Meeting

Product Backlog

- A list of all desired work on the project
  - Usually a combination of
    - story-based work (“let user search and replace”)
    - task-based work (“improve exception handling”)
- List is prioritized by the Product Owner
  - Typically a Product Manager, Marketing, Internal Customer, etc.

Product Backlog

- Requirements for a system, expressed as a prioritized list of Backlog Items
  - **Managed and owned by Product Owner**
  - **Spreadsheet (typically)**

Sample Product Backlog

	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	**Admin Program**	-	-
	9	Delete users	4	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	4	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

Sprint Backlog

- A subset of Product Backlog Items, which define the work for a Sprint
  - **Created by Team members**
  - **Each Item has it's own status**
  - **Updated daily**

Sprint Backlog during the Sprint

- Changes occur:
  - Team adds new tasks whenever they need to in order to meet the Sprint Goal
  - Team can remove unnecessary tasks
  - But: Sprint Backlog can only be updated by the team
- Estimates are updated whenever there's new information

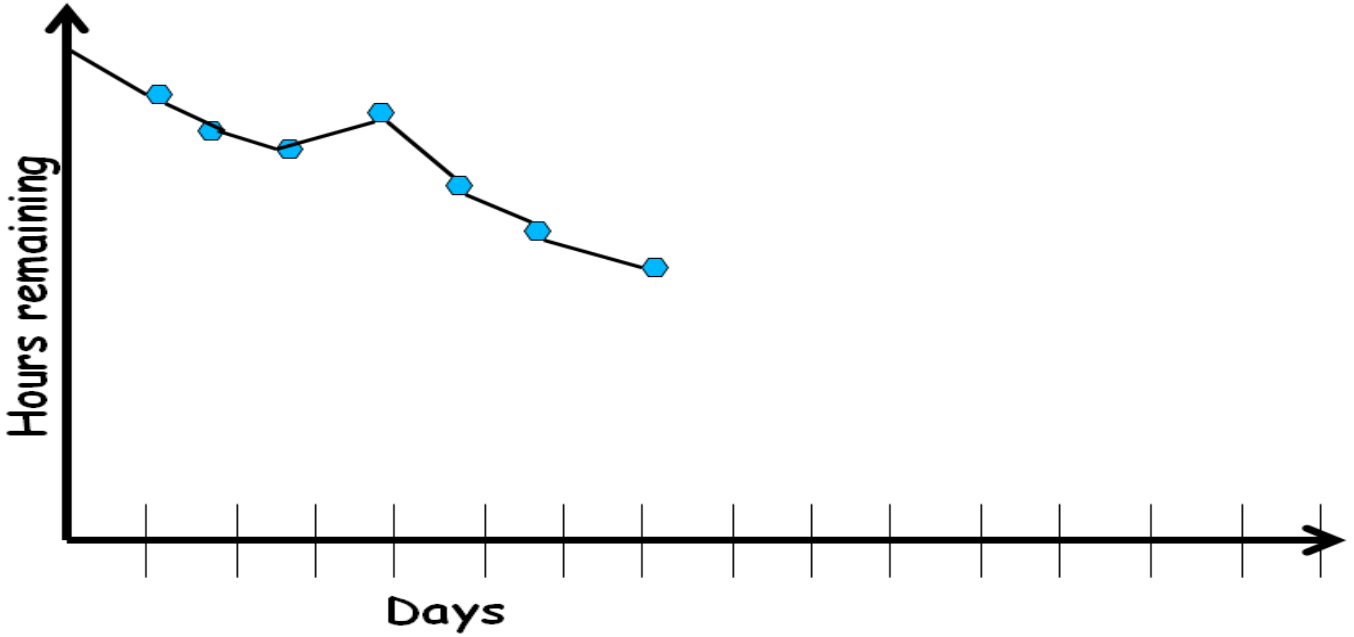
Burn down Charts

- Are used to represent “work done”.
- Are remarkably simple but effective Information disseminators
- 3 Types:
  - **Sprint Burn down Chart (progress of the Sprint)**
  - **Release Burn down Chart (progress of release)**
  - **Product Burn down chart (progress of the Product)**

Sprint Burn down Chart

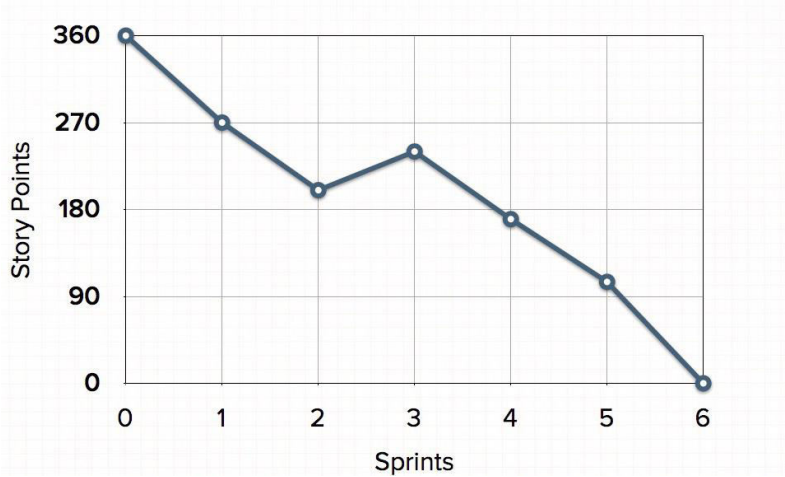
- Depicts the total Sprint Backlog hours remaining per day
- Shows the estimated amount of time to complete
- Ideally should burn down to zero to the end of the Sprint
- Actually is not a straight line

Sprint Burndown Chart



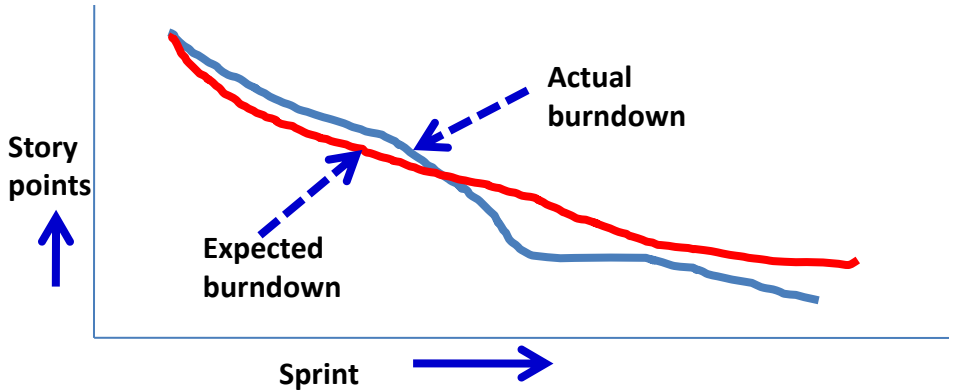
Release Burndown Chart

- Will the release be done on right time?
- How many more sprints?
- X-axis: sprints
- Y-axis: amount of story points remaining



Product Burndown Chart

- Is a “big picture” view of project’s progress (all the releases)



Scalability of Scrum

- A typical Scrum team is 6-10 people
- Jeff Sutherland - up to over 800 people
- "Scrum of Scrums" or "Meta-Scrum“

Thank You!!

Testing Agile Based Software

Prof. Durga Prasad Mohapatra
Dept. of CSE, NIT Rourkela
India

Agile software Development: Motivation

- There are many SDLC models available, such as waterfall model, prototyping model, iterative model, incremental model, spiral model and many more.
- These traditional models follow the *static* working strategy in which everything is fixed, for example cost of project, requirements or scope, schedule etc. Further, the duration is in years.
- Besides, there are several other drawbacks of each model.

Agile software Development

- In order to overcome these drawbacks, several major software vendors have been adopting a form of "intelligent" development in one or more phases of their software development processes.
- Agile for example, is a well-known example of a lifecycle used to build intelligent and analytical systems.
- Agile development does not promote best practices; rather, it is about adaptive planning and evolutionary development.

What is Agile Software Development?

- Agile: Easily moved, light, nimble, and active software processes.
- How agility achieved?
  - Fitting the process to the project
  - Avoiding things that waste time

Agile software Development

- ASD is based on light weight methodologies (less documentation and less planning) having dynamic nature, which is its major strength.
- This model provides an environment to accommodate frequent changes as per market standards and customer needs.
- ASD is an iterative and incremental development method and it is customer centred.

Agile software Development

- The agile process consists of multiple sprints (iterations or runs or development cycles);
  - in each sprint a specific software feature is developed, tested, refined and documented.
- However, because agile development depends on the context of the project,
  - testing is performed differently in every sprint.

Agile Model

- To overcome the shortcomings of the waterfall model of development,
  - it was proposed in mid-1990s
- The agile model was primarily designed:
  - To help projects to adapt to change requests
- In the agile model:
  - The requirements are decomposed into many small incremental parts that can be developed over one to four weeks each.

History: The Agile Manifesto

- On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, seventeen people met to talk, ski, relax, and try to find common ground—and of course, to eat.
- What emerged was the Agile 'Software Development' Manifesto.

History: The Agile Manifesto cont…

- Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened.

History: The Agile Manifesto cont…

- Now, a bigger gathering of organizational anarchists would be hard to find, so what emerged from this meeting was symbolic—a Manifesto for Agile Software Development—signed by all participants.

Manifesto for Agile Software Development

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.

Principles behind the Agile Manifesto

- 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, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Principles behind the Agile Manifesto cont…

- 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.

Principles behind the Agile Manifesto cont…

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- 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.

Principles behind the Agile Manifesto cont…

- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly

Ideology: Agile Manifesto

- Individuals and interactions *over*
  - process and tools

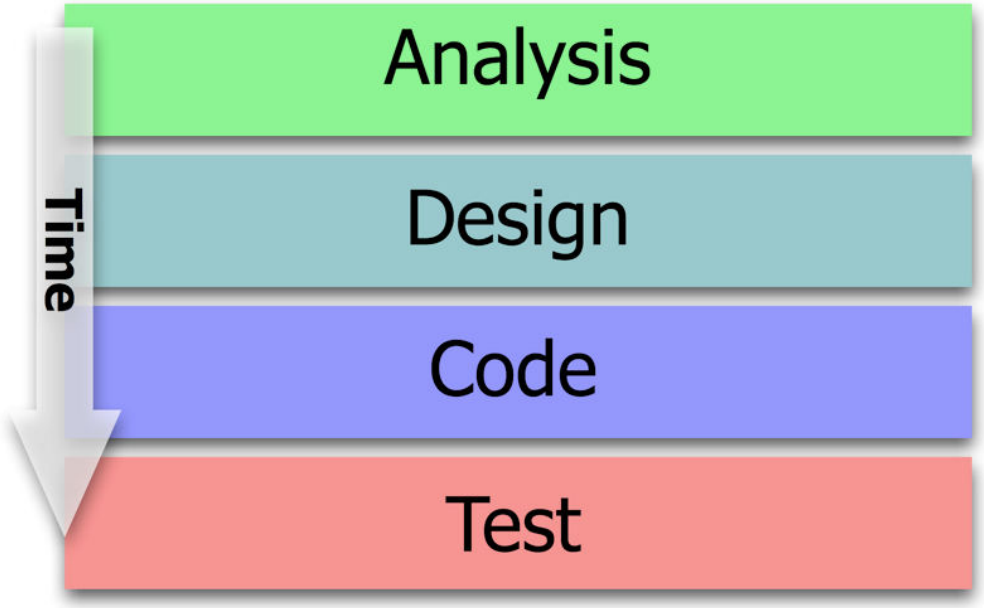
http://www.agilemanifesto.org

- Working Software *over*
  - comprehensive documentation
- Customer collaboration *over*
  - contract negotiation
- Responding to change *over*
  - following a rigid plan

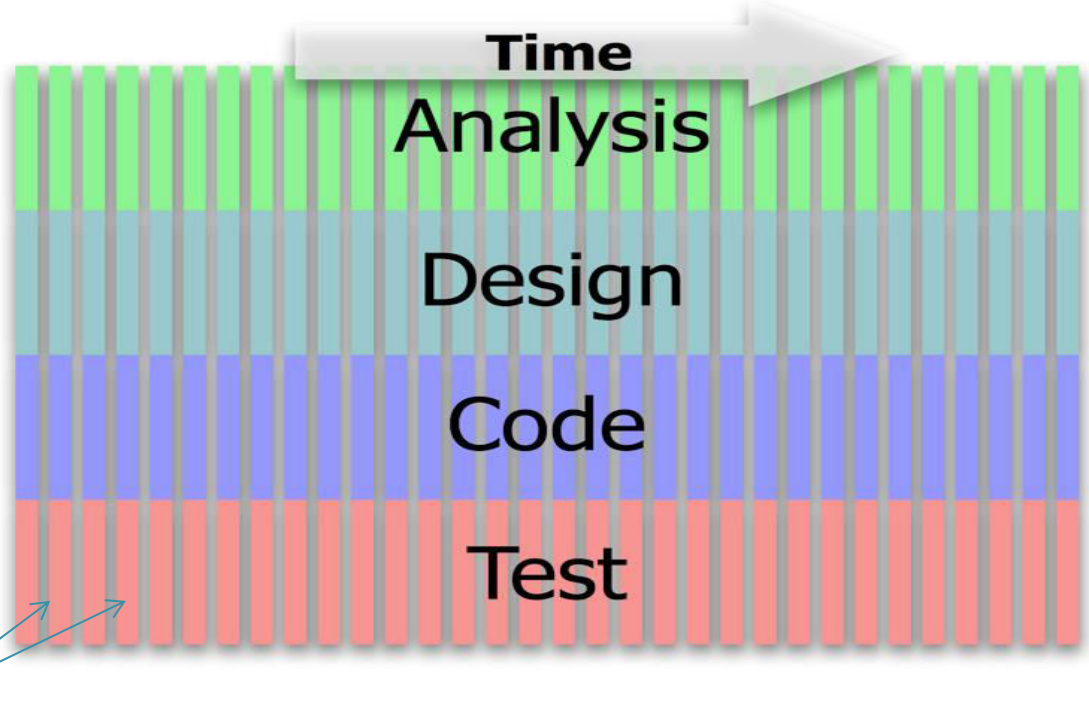
Agile Methodologies

- XP
- Scrum
- Unified process
- Crystal
- DSDM
- Lean

Traditional Software Development



Applying Lean Principles to Software Development ... (a better way of doing the same)



End-to-End small slices of work

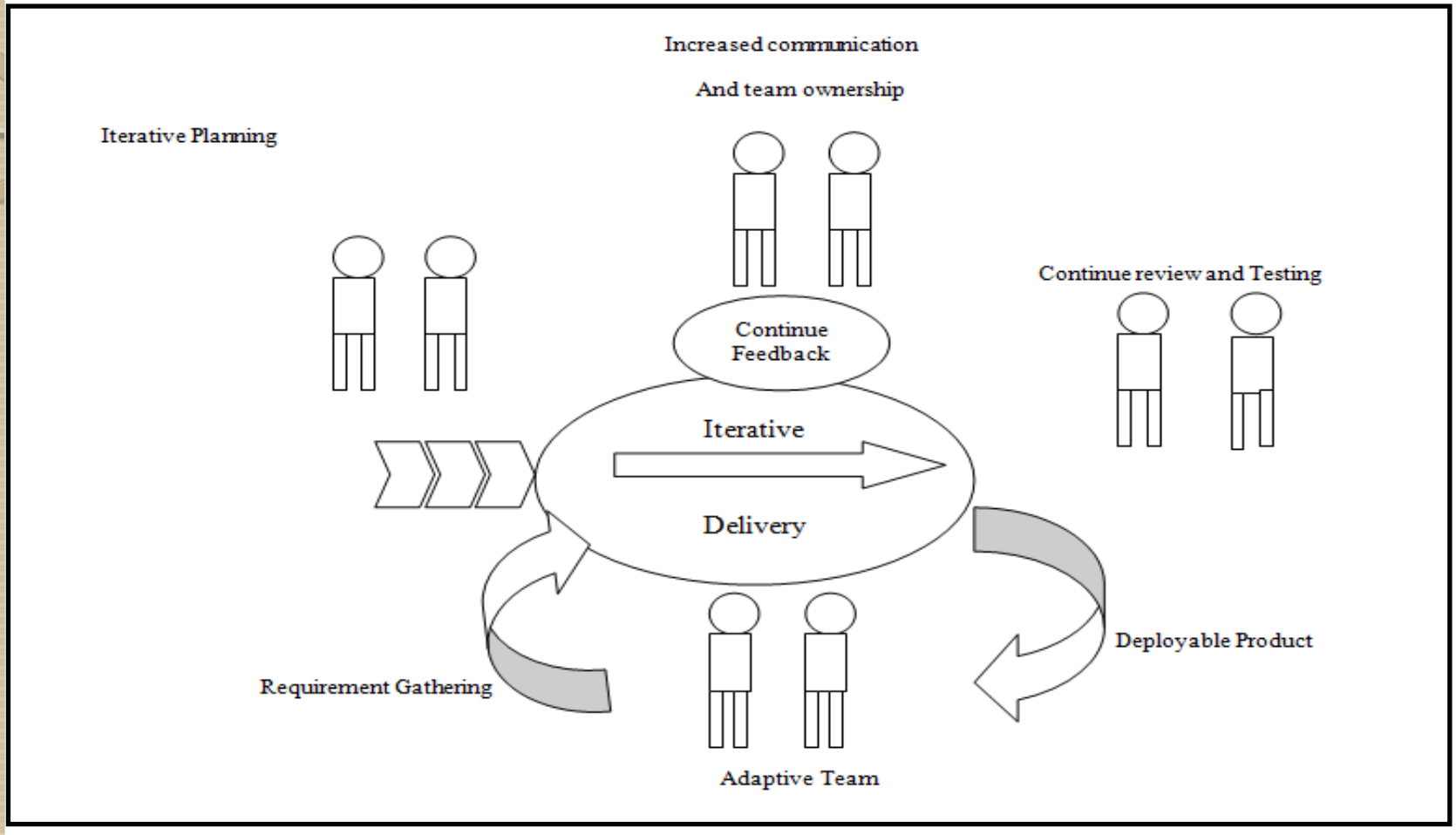
Agile Model

- The Agile model puts emphasis on the fact that whole Agile team should be a tightly integrated unit and is composed of
  - Developers
  - Quality assurance members
  - Testers
  - Project owner
  - Customer

Agile Model cont …

- The key feature of ASD is to have effective communication between all team members. For valuable communication and information exchange, daily meetings are held in ASD.
- Another important feature of agile process is iterative delivery.
  - An iteration or delivery cycle in ASD ranges from 1 – 4 weeks.

Agile Model cont …



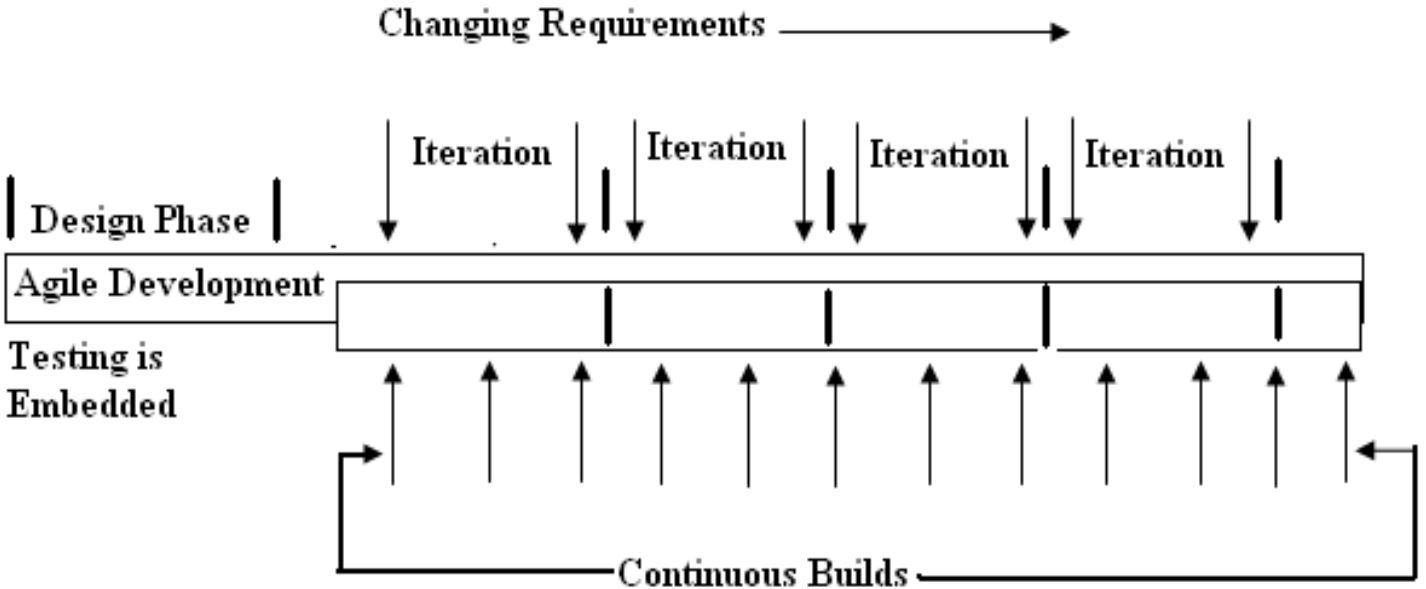
Agile Model: Principal Techniques

- User stories:
  - Simpler than use cases.
- Metaphors:
  - Based on user stories, developers propose a common vision of what is required.
- Spike:
  - Simple program to explore potential solutions.
- Refactor:
  - Restructure code without affecting behavior, improve efficiency, structure, etc.

Agile Model: Nitty Gritty

- At a time, only one increment is planned, developed, deployed at the customer site.
  - No long-term plans are made.
- An iteration may not add significant functionality,
  - But still a new release is invariably made at the end of each iteration
  - Delivered to the customer for regular use.

Agile Software Development Life Cycle



Stakeholders in Agile Life Cycle

- Agile SDLC includes specific activities performed by manager (M), developers (D), testers (T), marketing professional (MP) and customer (C).
- Table I lists the activities performed by different stakeholders
- At the time of production of the code or before producing the code, testing is applied by writing failed test cases, unlike the traditional approach of working.

Stakeholders in Agile Life Cycle

- Testing activity begins as soon as user stories (requirements) are finalized and prioritized, and
  - testers try to move business logic into lower levels in order to test with lower effort in the last stage.

Stakeholders in agile life cycle

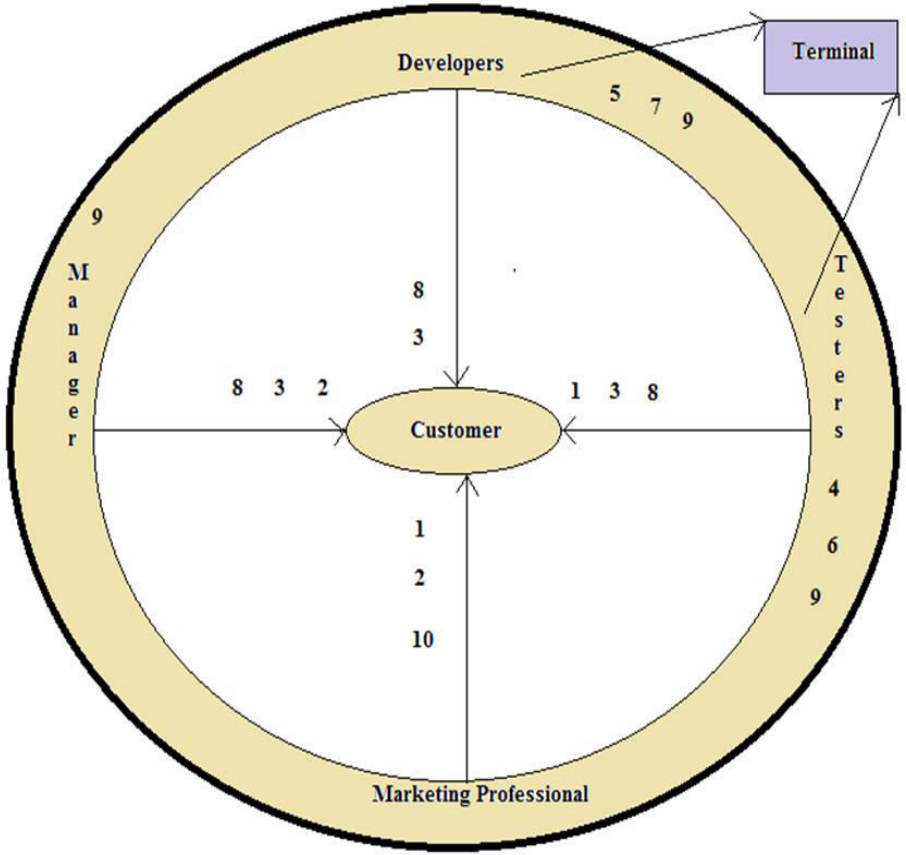


Fig 3: Stakeholders in agile life cycle

Table 1 : Actor Activity Chart - Agile Life Cycle

S. No.	Actor	Activity
1	C, MP, T	Requirements Gathering
2	M, MP, C	Effort estimation [28], cost, risk, capacity & resource Estimation
3	M, D, T, C	User stories Prioritization
4	T	Designing of Test Cases
5	D	Coding for the user story in the iteration
6	T	Feedback
7	D	Refactoring for the user story
8	C, M, D, T	Review meeting with Demonstration
9	D, T, M, MP	Lesson Learning phase or Retrospective session after the iteration
10	C, MP	Release

Agile software development life cycle

- In agile, a quality(Q) product is delivered by operational teams, and acceptance factor (A) is related to the rate at which customer accepts the delivered product.
- Effectiveness (E) of the team is related to two factors as shown in below equation

$$E = Q \times A$$

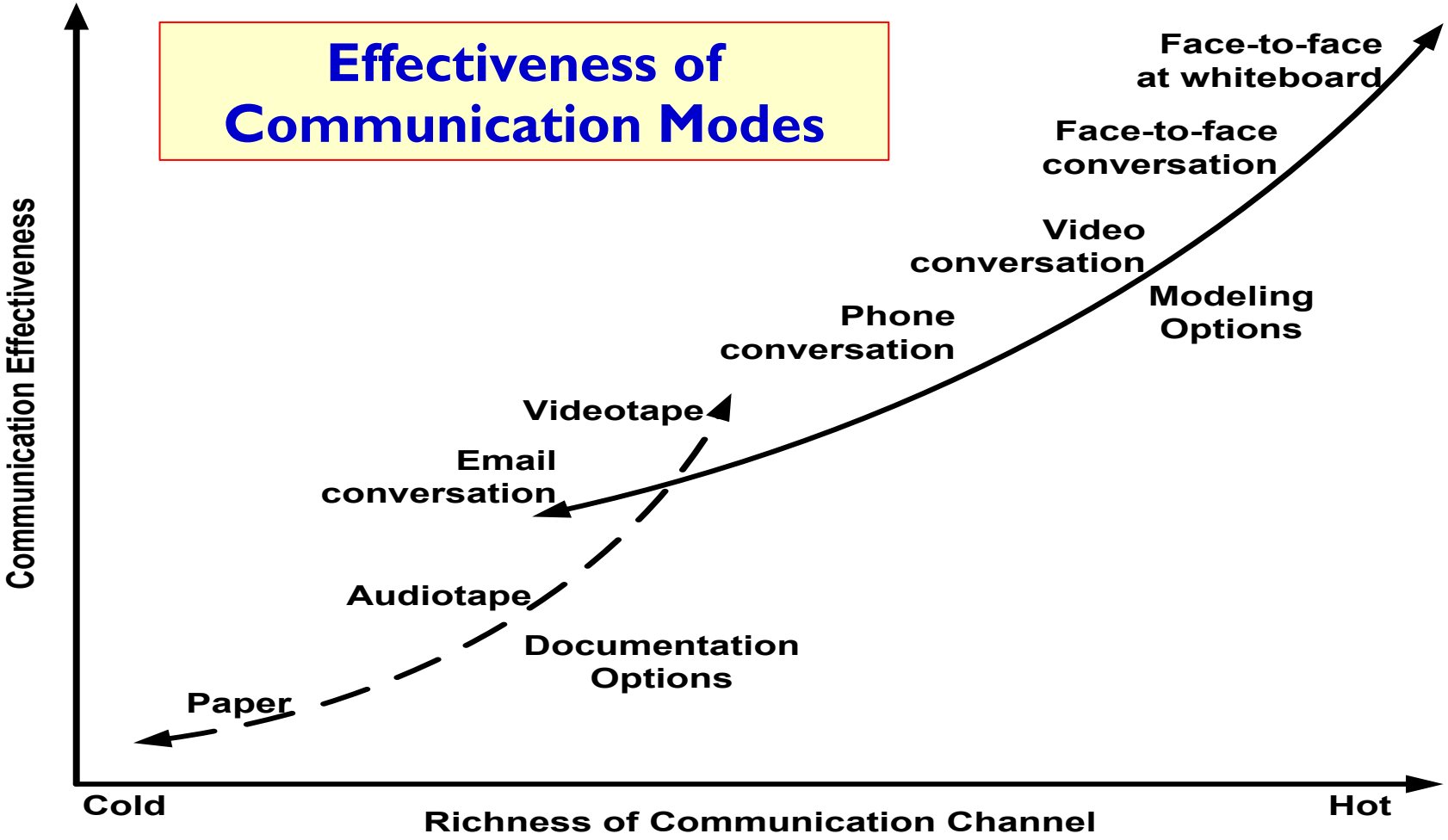
Agile software development life cycle cont…

- Out of these 2 factors, quality is more significant as acceptance is totally dependent on Q.
- Q is more when there is lesser number of backlogs and it is less when backlog items are more.
- Backlogs can be decreased when automation is the preferred approach over a manual way of testing.

Methodology

- Face-to-face communication is favoured over written documents.
- To facilitate face-to-face communication,
  - Development team should share a single office space.
  - Team size is deliberately kept small (5-9 people).
  - This makes the agile model most suited to the development of small projects.

Effectiveness of Communication Modes



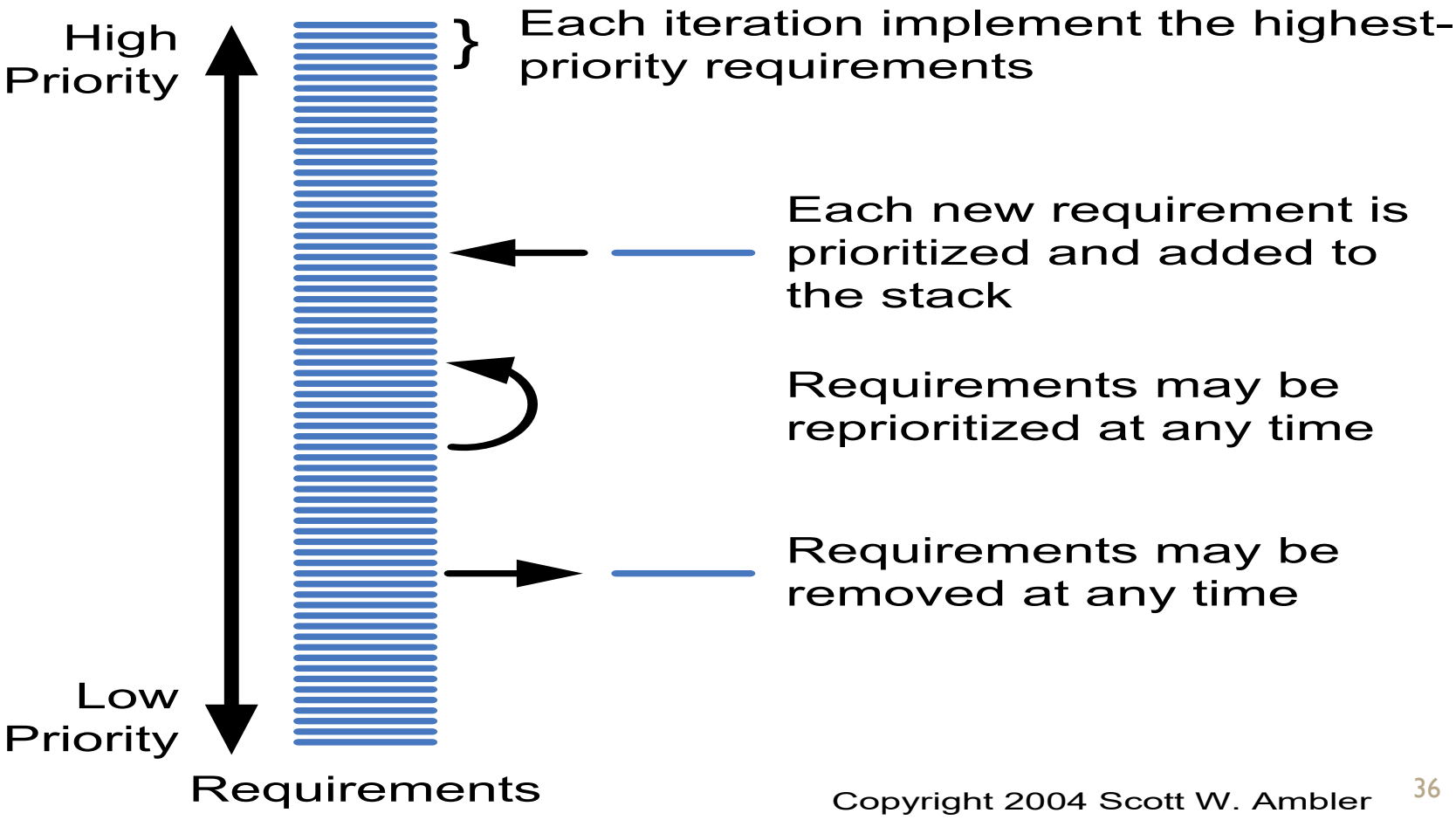
Agile Model: Principles

- The primary measure of progress:
  - Incremental release of working software
- Important principles behind agile model:
  - Frequent delivery of versions --- once every few weeks.
  - Requirements change requests are easily accommodated.
  - Close cooperation between customers and developers.
  - Face-to-face communication among team members.

Agile Documentation

- Travel light:
  - You need far less documentation than you think.
- Agile documents:
  - Are concise
  - Describe information that is less likely to change
  - Describe "good things to know"
  - Are sufficiently accurate, consistent, and detailed
- Valid reasons to document:
  - Project stakeholders require it
  - To define a contract model
  - To support communication with an external group
  - To think something through

Agile Software Requirements Management



Adoption Detractors

- Sketchy definitions, make it possible to have
  - Inconsistent and diverse definitions
- High quality people skills required
- Short iterations inhibit long-term perspective
- Higher risks due to feature creep:
  - Harder to manage feature creep and customer expectations
  - Difficult to quantify cost, time, quality.

Agile Model Shortcomings

- Derives agility through developing tacit knowledge within the team, rather than any formal document:
  - Can be misinterpreted…
  - External review difficult to get…
  - When project is complete, and team disperses, maintenance becomes difficult…

Agile Model vs Waterfall Model

- Steps of Waterfall model are a planned sequence:
  - Requirements-capture, analysis, design, coding, and testing .
- Progress is measured in terms of delivered artefacts:
  - Requirement specifications, design documents, test plans, code reviews, etc.
- In contrast agile model sequences:
  - Delivery of working versions of a product in several increments.

Agile Model vs Waterfall Model cont …

- As regards to similarity:
  - We can say that Agile teams use the waterfall model on a small scale.

Summary

- Discussed the basics of agile software development.
- Highlighted some existing agile methodologies.
- Discussed the agile model in detail.
- Explained the Agile Software Development Life Cycle.
- Presented some of the shortcomings of agile model.
- Compared Agile Model vs Waterfall Model.

References

1. Rajib Mall, Fundamentals of Software Engineering, Fifth Edition, PHI, 2018.
2. Naresh Chauhan, Software Testing: Principles and Practices, (Chapter – 16), Second Edition, Oxford University Press, 2018.

Thank You

Testing Agile Based Software cont …

Prof. Durga Prasad Mohapatra
Dept. of CSE, NIT Rourkela
India

Scrum

Introduction

- Scrum is an Agile framework which concentrates on how team members should function to produce system flexibly in a constantly changing environment.
- Scrum is concerned with the product owner, project lead, and the team working together in an intensive and interdependent manner.

Introduction cont …

- Scrum process includes three phases:
  - Pre-game
  - Development
  - Post-game

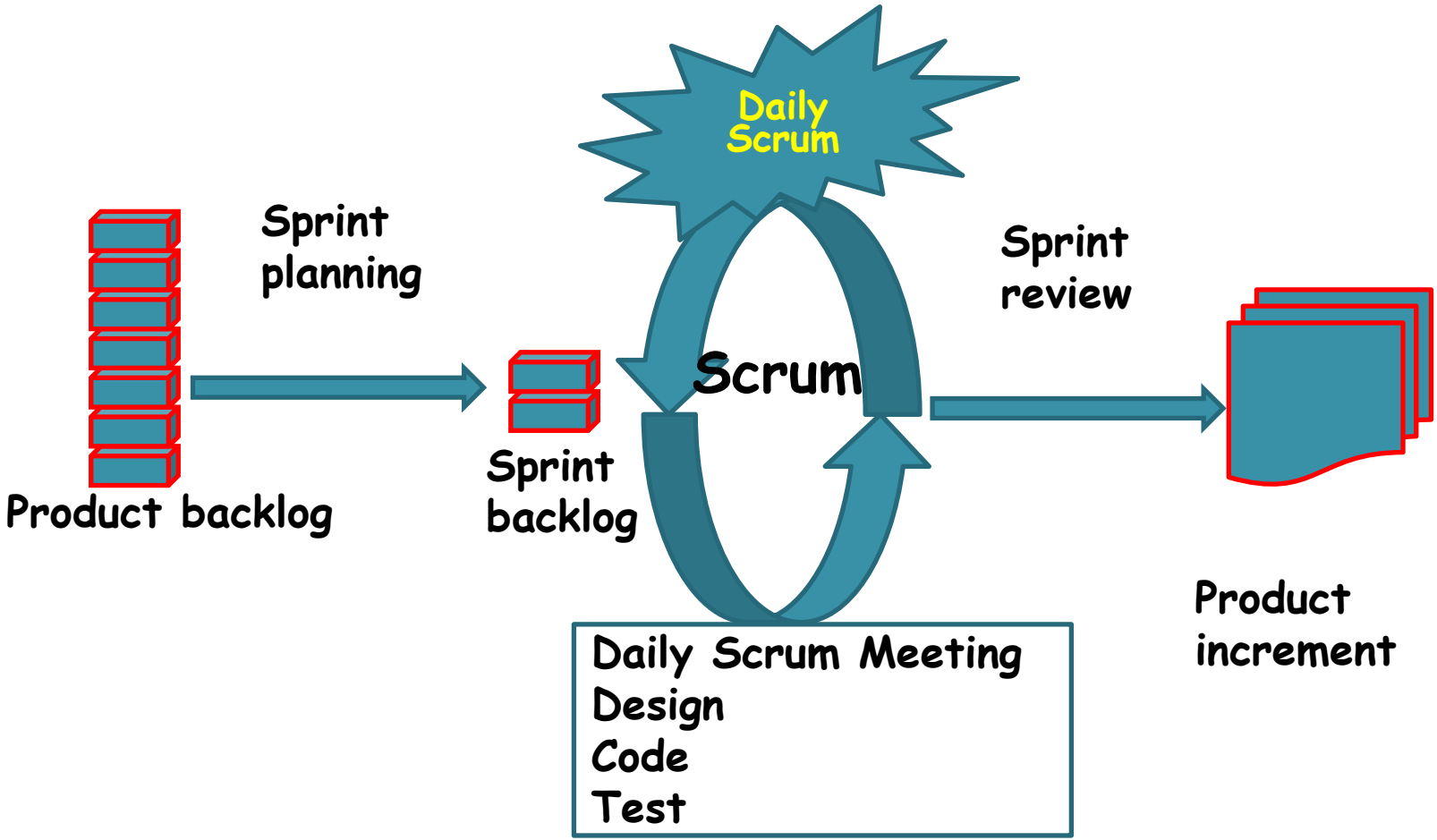
Introduction cont …

- There are six identifiable roles in Scrum that have different tasks and purposes during the process and its practices:
  - Scrum master
  - Product owner
  - Scrum team
  - Customer
  - User
  - Management

Scrum: Characteristics

- Self-organizing teams
- Product progresses in a series of month-long **sprints** (iterations or runs or development cycles)
- Requirements are captured as items in a list, called **product backlog**
- One of the agile processes

Scrum Life Cycle



Sprint

- Scrum projects progress in a series of "sprints"
  - Analogous to XP iterations or time boxes
  - Target duration is one month
- Software increment is designed, coded, and tested during the sprint
- No changes are entertained during a sprint

Scrum Framework

- Roles: Product Owner, Scrum Master, Scrum Team, Customer, End User, Management
- Ceremonies: Sprint Planning, Sprint Review, Sprint Retrospective, and Daily Scrum Meeting
- Artifacts: Product Backlog, Sprint Backlog, and Burndown Chart

Key Roles and Responsibilities in a Scrum Team

- Product Owner
  - Represents customers' views and interests.
- Development Team
  - Team of 5 - 10 people with cross-functional skill sets.
- Scrum Master (aka Project Manager)
  - Facilitates scrum process and resolves impediments at the team and organization level by acting as a buffer between the team and outside interference.

Product Owner

- Defines the features of the product
- Decides on release date and content
- Prioritizes features according to usefullness
- Adjusts features and priority every iteration, as needed
- Accepts or reject work results.

The Scrum Master

- Represents management in the project
- Removes impediments
- Ensures that the team is fully functional and productive
- Enables close cooperation across all roles and functions
- Shields the team from external interferences

Scrum Team

- Typically 5-10 people
- Cross-functional
  - Quality Assurance Engineers, Programmers, UI Designers, etc.
- Teams are self-organizing
- Membership can change only between sprints

Sprint

- Fundamental process flow of Scrum
- It is usually a month-long iteration:
  - during this time an incremental product functionality is completed
- No outside influence is allowed to interfere with the Scrum team during the Sprint
- Each day begins with the Daily Scrum Meeting

Ceremonies

- Sprint Planning Meeting
- Daily Scrum
- Sprint Review Meeting

Sprint Planning

- Goal is to produce Sprint Backlog
- Product owner works with the Team to negotiate what Backlog Items
- Scrum Master ensures that the Team agrees to the realistic goals

Daily Scrum

- Daily
- 15-minutes
- Stand-up meeting
- Not for problem solving
- Three questions:
  1. What did you do yesterday?
  2. What will you do today?
  3. What obstacles are in your way?

Daily Scrum

- Is NOT a problem solving session
- Is NOT a way to collect information about WHO is behind the schedule
- Is a meeting in which team members review what is done and make informal commitments to each other and to the Scrum Master
- Is a good way for a Scrum Master to track the progress of the Team

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features
- Informal
  - 2-hour prep time rule
- Participants
  - Customers
  - Management
  - Product Owner
  - Other team members

Sprint Review Meeting

Product Backlog

- A list of all desired work on the project
  - Usually a combination of
    - story-based work ("let user search and replace")
    - task-based work ("improve exception handling")
- List is prioritized by the Product Owner
  - Typically a Product Manager, Marketing, Internal Customer, etc.

Product Backlog

- Requirements for a system, expressed as a prioritized list of Backlog Items
  - Managed and owned by Product Owner
  - Spreadsheet (typically)

Sample Product Backlog

	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	**Admin Program**	-	-
	9	Delete users	4	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	4	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

Sprint Backlog

- A subset of Product Backlog Items, which defines the work for a Sprint
  - Created by Team members
  - Each Item has it's own status
  - Updated daily

Sprint Backlog during the Sprint

- Changes occur:
  - Team adds new tasks whenever they need in order to meet the Sprint Goal
  - Team can remove unnecessary tasks
  - But, Sprint Backlog can only be updated by the team
- Estimates are updated whenever there's new information

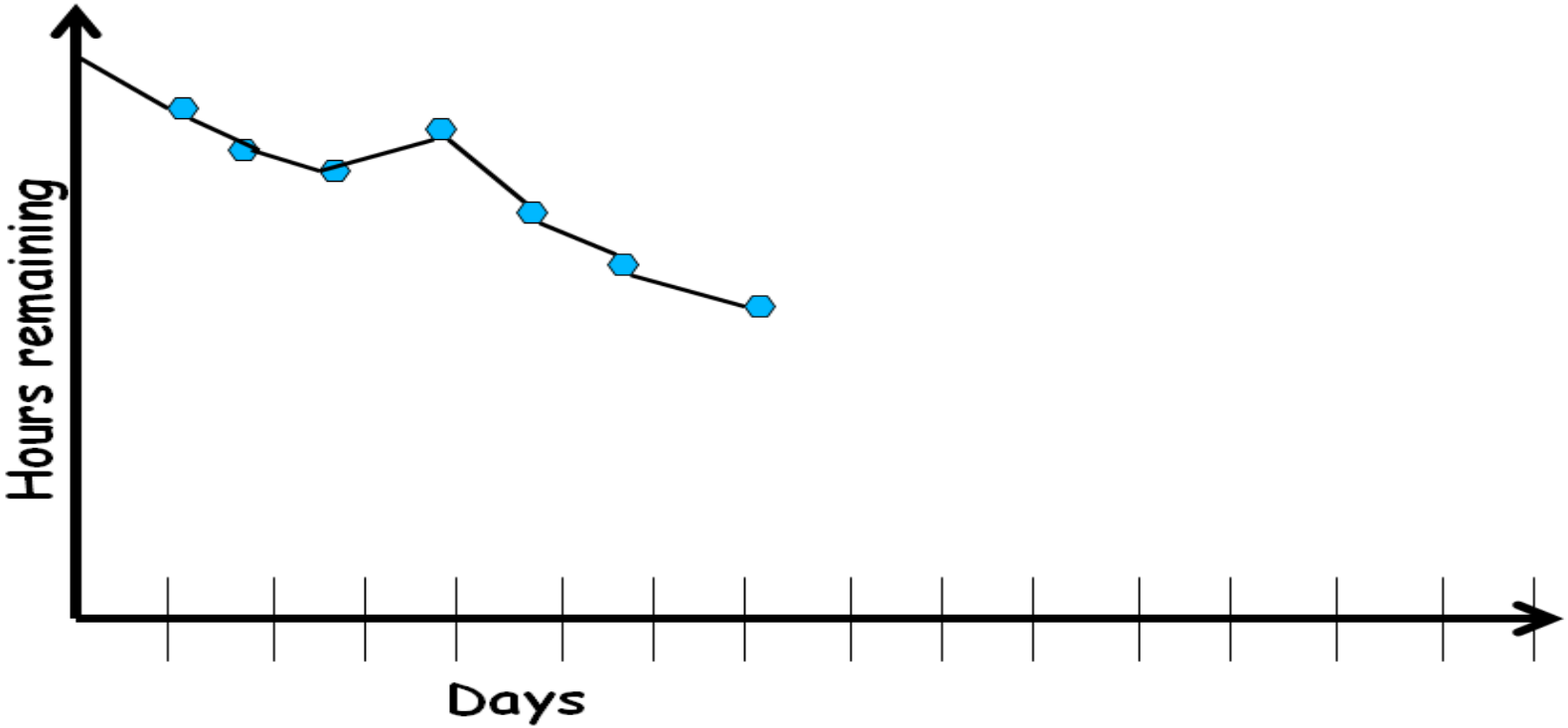
Burn down Charts

- Are used to represent "work done".
- Are remarkably simple but effective Information disseminators
- 3 Types:
  - Sprint Burn down Chart (progress of the Sprint)
  - Release Burn down Chart (progress of release)
  - Product Burn down chart (progress of the Product)

Sprint Burn down Chart

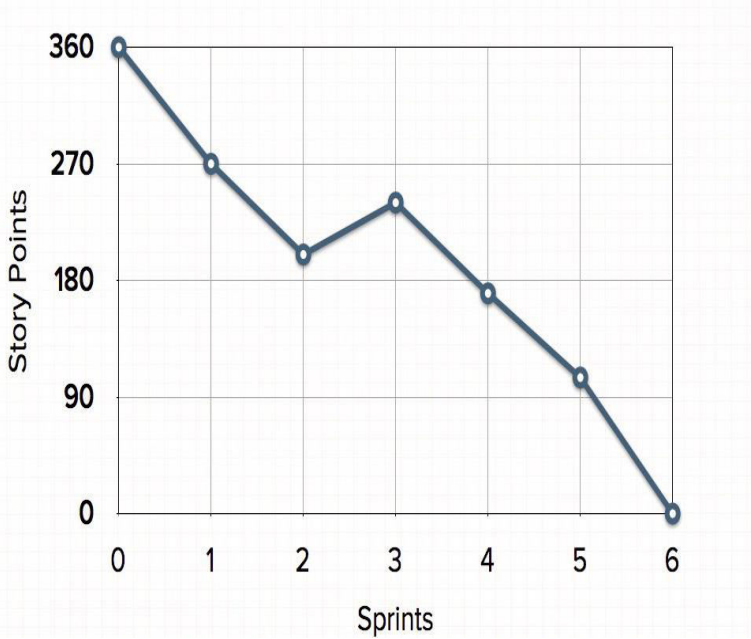
- Depicts the total Sprint Backlog hours remaining per day
- Shows the estimated amount of time to complete
- Ideally should burn down to zero to the end of the Sprint
- Actually is not a straight line

Sprint Burndown Chart



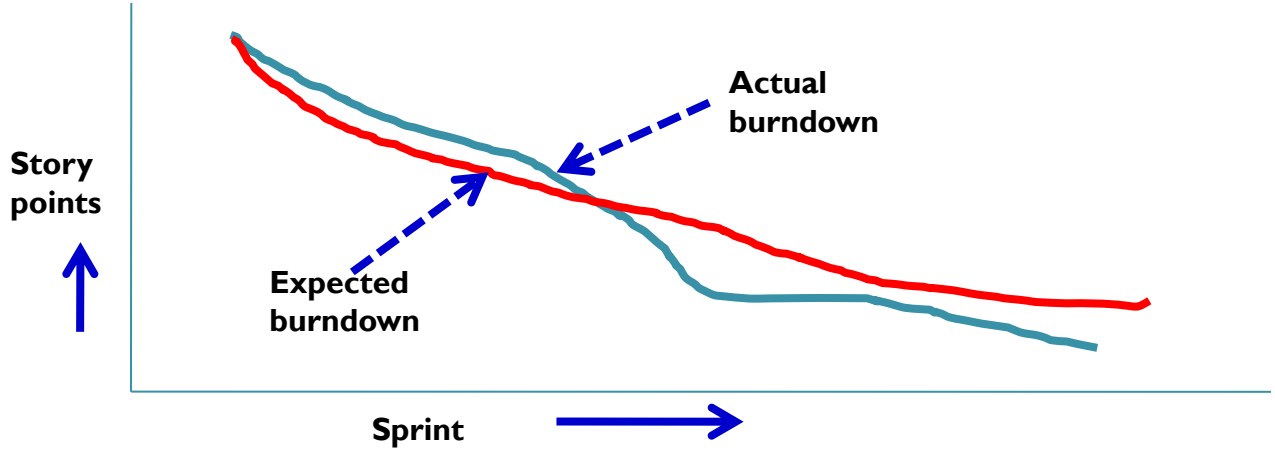
Release Burndown Chart

- Will the release be done on right time?
- How many more sprints?
- X-axis: sprints
- Y-axis: amount of story points remaining



Product Burndown Chart

- It is a "big picture" view of project's progress (all the releases)



Scalability of Scrum

- A typical Scrum team is 6-10 people
- Jeff Sutherland - up to over 800 people
  - "Scrum of Scrums" or "Meta-Scrum"

Summary

- Discussed the basic concepts of Scrum.
- Presented the characteristics of Scrum.
- Explained the Scrum Life Cycle.
- Discussed the Scrum Framework.
  - Roles
  - Ceremonies
  - Artifacts
- Explained the Burn down Charts.

References

1. Rajib Mall, Fundamentals of Software Engineering, Fifth Edition, PHI, 2018.
2. Naresh Chauhan, Software Testing: Principles and Practices, (Chapter – 16), Second Edition, Oxford University Press, 2018.

Thank You

Testing Agile Based Software cont..

Prof. Durga Prasad Mohapatra
Dept. of CSE, NIT Rourkela
India

Agile Testing Life Cycle

- Agile Testing Life Cycle is based on the *'more is less'* principle which focuses on communication management among stakeholders.
- The adoption of this principle results in quality software product as shown in next figure.
- If communication between the tester and other stakeholder is very frequent and effective, then there would be very few doubts and more clarity.

Agile Testing Life Cycle cont..

- This principle ensures that testing activities along with effective communication and collaboration among major stakeholders, such as business analyst, market evaluator, customer, and developer, result in software products having only few defects.

More is Less Principle

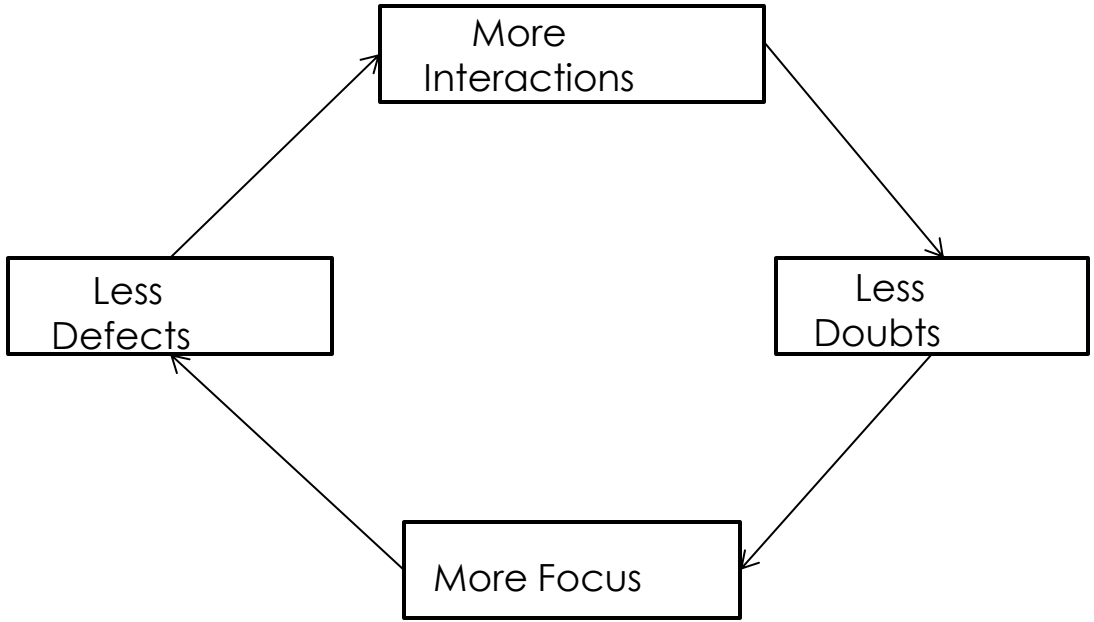


Fig 1: More is less principle

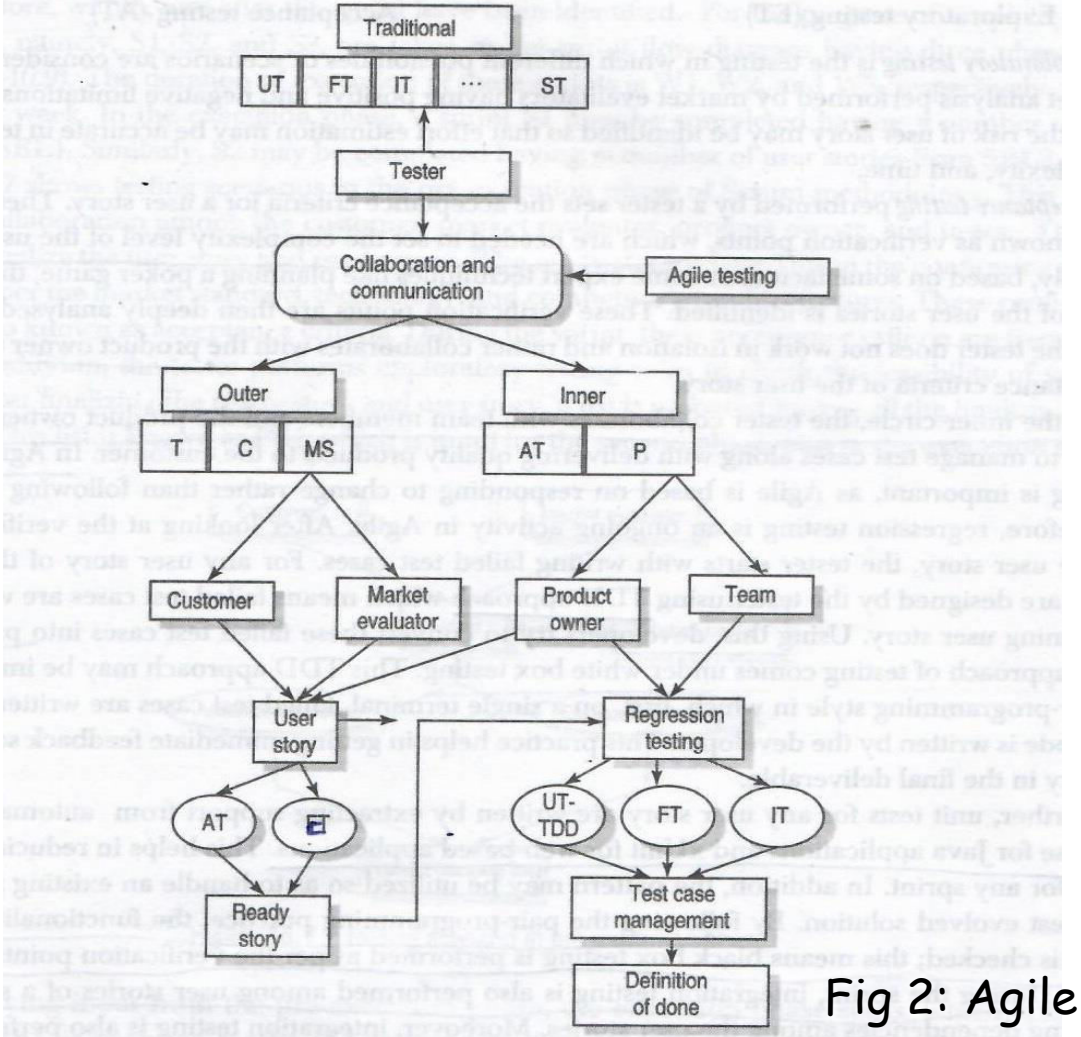


Fig 2: Agile testing life cycle

Agile Testing Abbreviations

Abbreviation	Full Form
UT	Unit Testing
FT	Functional Testing
IT	Integration Testing
ST	System Testing
T	Technology
C	Competitor
MS	Market Standard
AT	Automated Tool
P	Pattern
AT	Acceptance Testing
ET	Exploratory Testing
TDD	Test Driven Development

Agile Testing Life Cycle cont..

- An agile tester interacts and collaborates with 2 circles, namely an outer circle and inner circle.
- The outer circle is connected to the outside world
- In the outer circle the tester collaborates with customers and market evaluators.
- Then convert the informal requirements into a formal set of requirements known as the *user story*.

Agile Testing Life Cycle cont …

- Then the tester converts the user story into a ready story.
- This ready story acts like a checklist at the time of verification or acceptance of the user story by the customer.
- This ready story is the outcome of performing 2 types of testing.
  - Exploratory testing(ET)
  - Acceptance testing(AT)

Agile Testing Life Cycle cont..

- Exploratory testing is the testing in which different possibilities or scenarios are considered/explored as per the market analysis performed by market evaluators having positive and negative limitations.
- At the same time, the risk of user story may be identified so that effort estimation may be accurate in terms of effort, complexity, time.

Agile Testing Life Cycle cont..

- Acceptance testing performed by a tester sets the acceptance criteria for a user story. These criteria also known as verification points, which are needed to set the complexity level of the user story. These verification points are then deeply analysed.
- In this case also, the tester does not work in isolation and rather collaborates with the product owner to finalize the acceptance criteria of the user story.

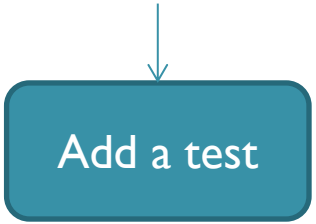
Agile Testing Life Cycle cont..

- In the inner circle, the tester collaborates with team members and product owner.
- In Agile, regression testing is important, as Agile is based on responding to change rather than following a fixed plan.
- So regression testing is an on going activity in Agile.

Agile Testing Life Cycle cont..

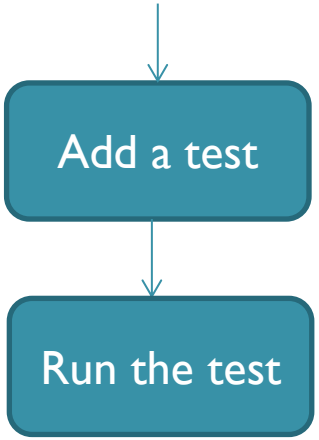
- After looking at the verification points of the user story, the tester starts with writing failed test cases.
- For any user story of the sprint, test cases are designed by the tester using Test Driven Development(TDD) approach which means failed test cases are written for the upcoming user story.

Test Driven Development

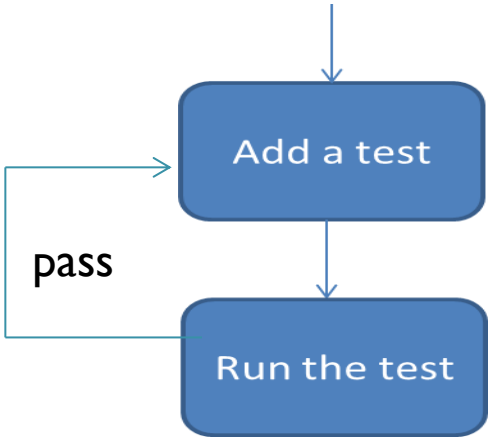


TDD Rhythm-Test, Code ,Refactor

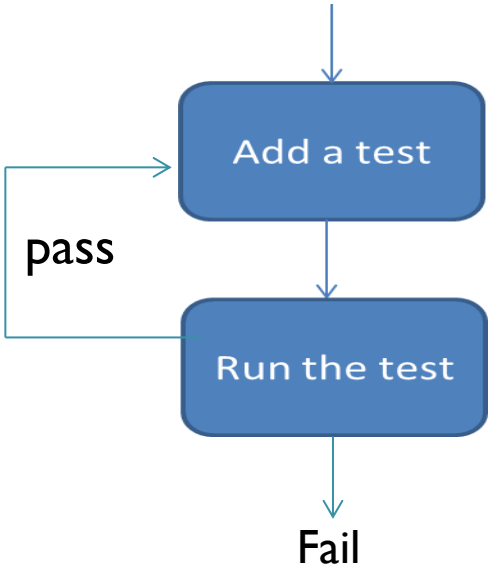
Test Driven Development cont..



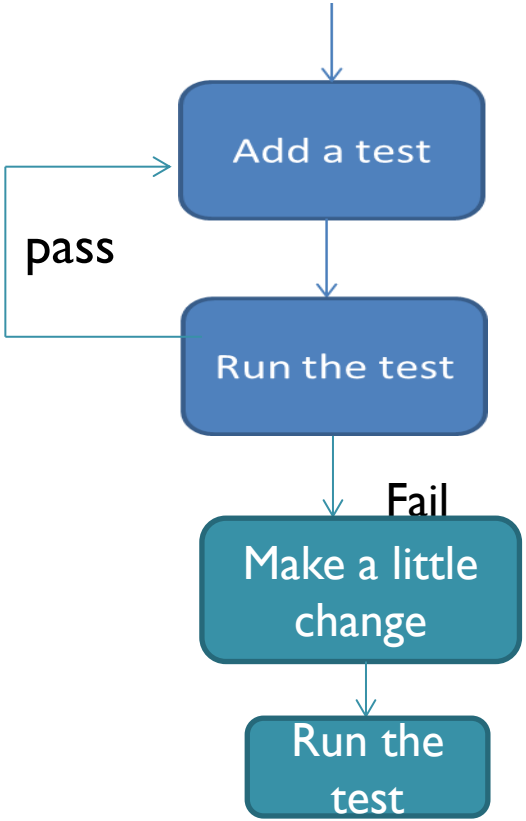
Test Driven Development cont..



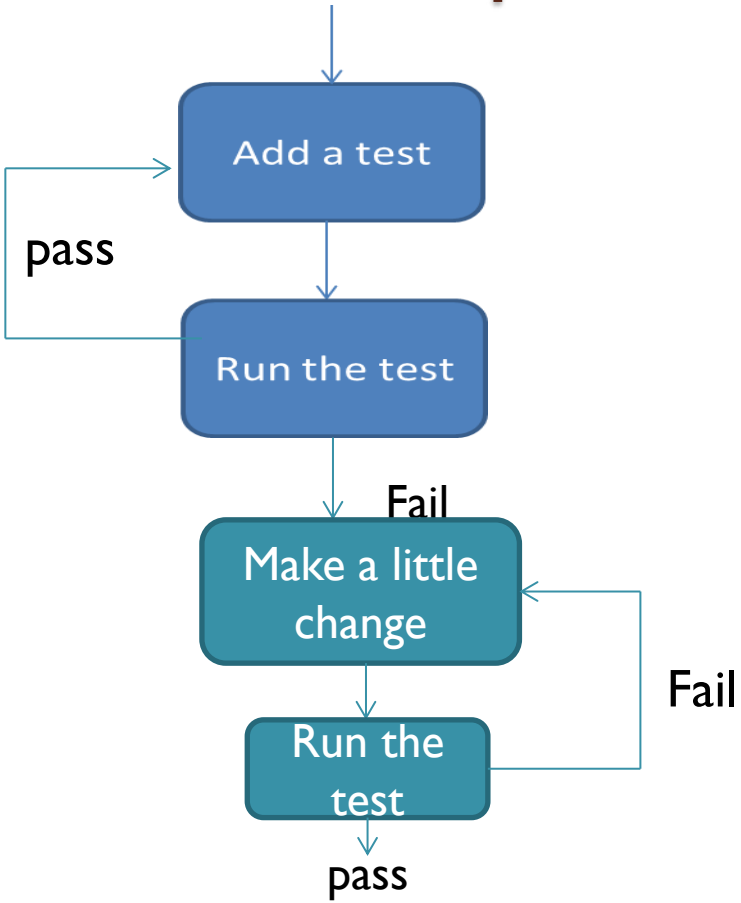
Test Driven Development cont..



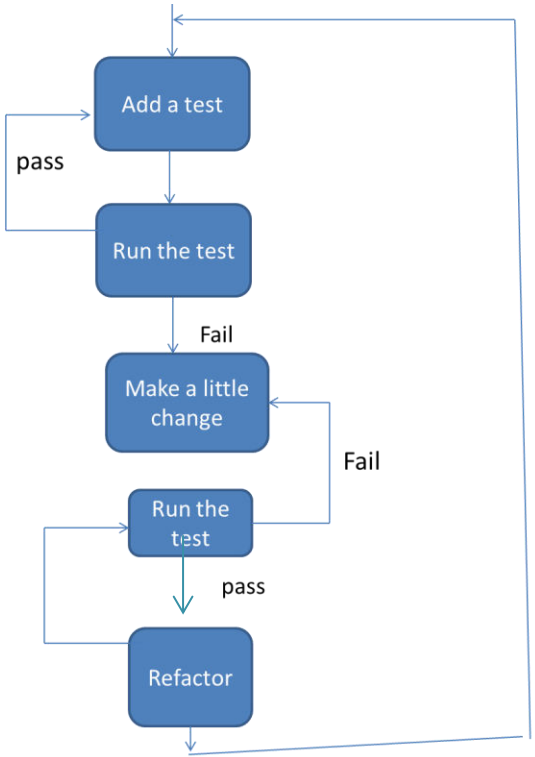
Test Driven Development cont..



Test Driven Development cont..



Test Driven Development cont..



Agile Testing Life Cycle cont..

- Using this, developers try to convert these failed test cases into pass test cases.
- This TDD approach may be implemented in a pair-programming style in which, first, on a single terminal, failed test cases are written, after which the code is written by the developer.
- This practice helps in getting immediate feedback so as to embed quality in final deliverable.

Agile Testing Life Cycle cont..

- Unit tests for any user story are written by extracting support from automated tools like eclipse for Java applications and xUnit for web-based applications.
- This helps in reducing the overall time for any sprint.
- In addition, the pattern may be utilized so as to handle an existing problem with the best evolved solution.

Agile Testing Life Cycle cont..

- During the sprint, integration testing also performed among user stories of a sprint by considering dependencies among the user stories.
- Moreover, integration testing is also performed among user stories of the different sprints.

Agile Testing Life Cycle cont..

- Further, to manage test cases, effective regression techniques, such as regression test selection (RTS) and test case prioritization (TCP) are implemented to run only a subset of test cases out of all the test cases.
- Finally, 'definition of done' is declared by the customer after matching the verification points of the ready story with the actual product.

Testing in scrum phases

- Scrum methodology is based upon small duration sprints having small number of user stories listed in the sprint backlog list(SBL), which is a subset of PBL(product backlog list).
- Testing in scrum is divided into 3 phases called
  - Pre-execution phase
  - Execution phase
  - Post-execution phase

Testing in scrum phases cont..

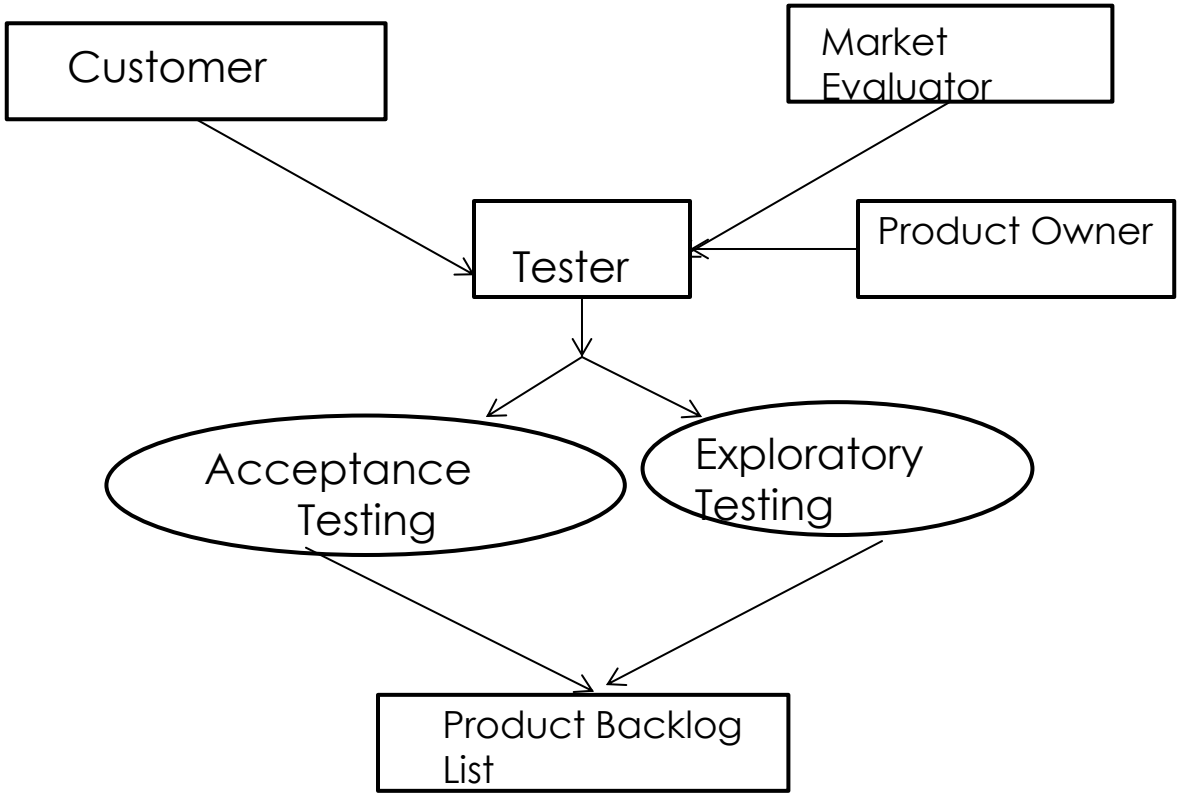
- Here all the testing activities occurring before, within, after the sprint have been identified.
- Here for simplicity, only 3 sprints S1, S2, S3 are taken in sprint flow diagram having 3 phases.
- The duration of execution of these sprints is W1, W2, W3 respectively, where W stands for week.

Testing in scrum phases cont..

- In the execution phase, a sprint S1 may be completed having ‘n’ number of user stories from SBL1.

- Similarly, S2 may be completed having ‚m, number of user stories from SBL2.

Testing Scenario in Pre-execution Phase



Testing Scenario in Pre-execution Phase cont..

- This phase starts with collaboration among the customer, market evaluator, product owner and tester.
- They sit together to finalize the user story and ready story.
- The ready story is based upon the conforming points which are as per the market standard, technology and competitor's product features.

Testing Scenario in Pre-execution Phase cont..

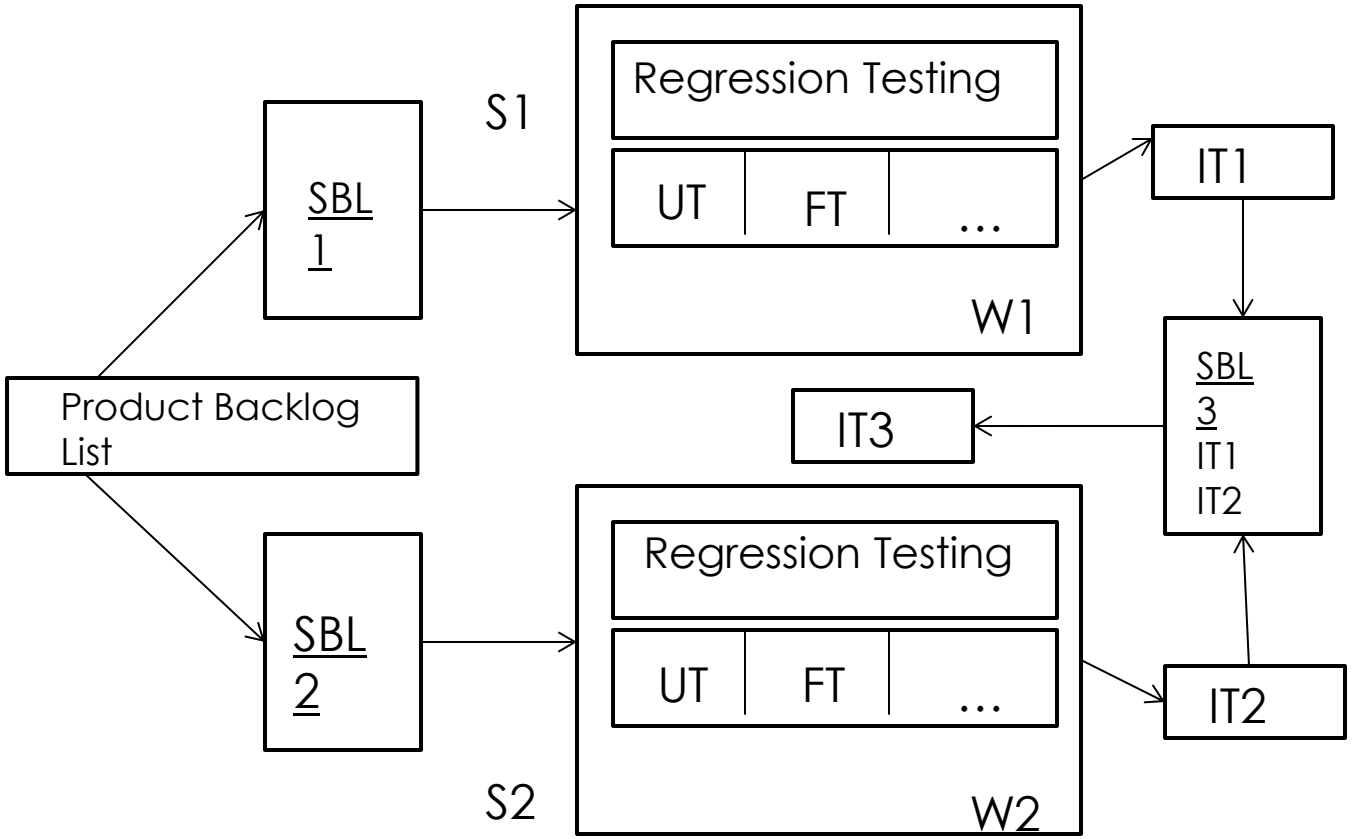
- These conforming points are also known as acceptance criteria.
- During the sprint the acceptance criteria are frequently checked.
- In addition, the tester performs exploratory testing so as to check the feasibility of various scenarios.

Testing Scenario in Pre-execution Phase cont..

- After finalizing the ready story and the user story, a list is prepared having all the finalized set of user stories.

- This list is known as PBL, which is input for the second phase, that is the execution phase.

Testing Scenario in Execution Phase



Testing Scenario in Execution Phase cont..

- After receiving input from the pre-execution phase, the execution phase starts.
- PBL is analyzed by the PO(product owner) and the effort estimation is done for selecting the user stories for SBL1 and SBL2. SBL1 and SBL2 are executed in sprint S1 and S2 respectively.
- In S1 the tester performs unit testing with TDD or white box testing, functional testing or black-box testing, regression testing, integration testing among dependent user stories, and many more depending on the requirements of customer.

Testing Scenario in Execution Phase cont..

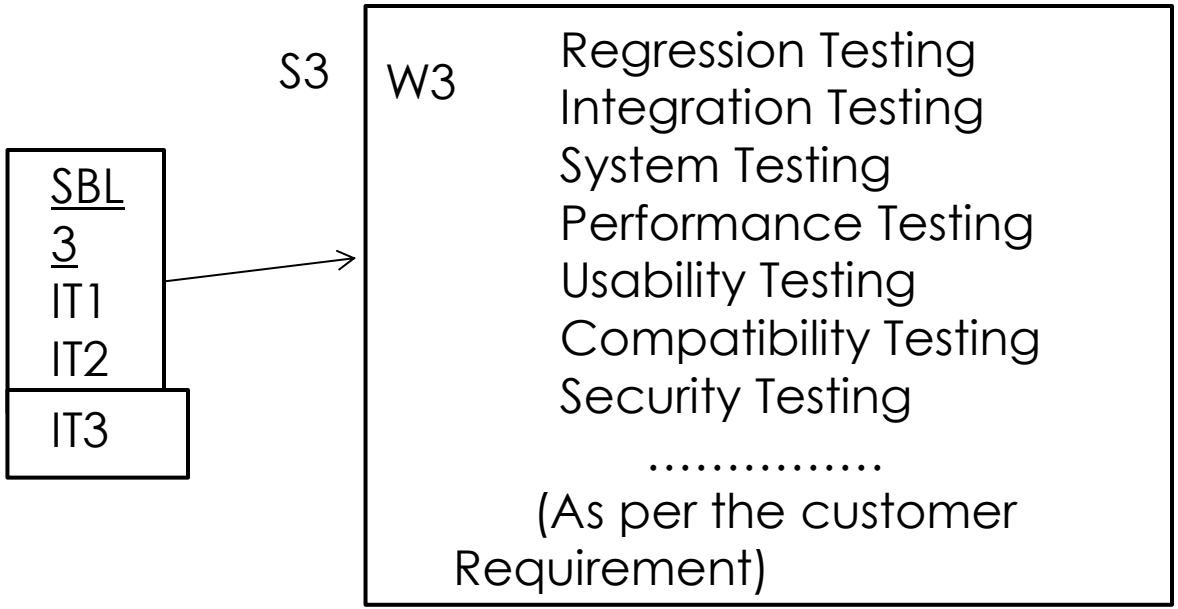
- The output of S1 is an integrated set of user stories, IT1 with regression test suite during W1 duration.

- Similarly , in sprint S2, the same types of testing are performed--- an integrated set of user stories IT2 with regression test suite during W2 duration.

Testing Scenario in Execution Phase cont..

- Further, these integrated set of user stories IT1 and IT2 are considered user stories In SBL3.
- Further, there may be other user stories which need to be developed in W3 duration which are newly added features in the maintenance time of the product.
- SBL3 is input for the post-execution phase.

Testing Scenario in Post-Execution Phase



Testing Scenario in Post-Execution Phase cont..

- In post-execution phase, user stories are selected from SBL3, based on the priority set by the customer, complexity level, risk level, or any other prioritization factor.
- Various types of testing that are performed in S3 are integration of IT1 and IT2, functional testing, system testing, and regression testing depending on the modification suggested by the customer, if any.

Testing Scenario in Post-Execution Phase cont..

- Other optional testing that may be performed in W3 duration are compatibility testing, security testing, performance testing, usability testing etc.

- Finally, a software product is delivered to the customer.

Regression testing in Agile

- Regression testing in Agile environment is practised under two major categories.
- **Sprint level regression testing (SLRT):** Focuses on testing new functionalities that have been incorporated since the last production release.
- **End-to-end regression testing(EERT):** Refers to the regression testing that incorporates all the fundamental functionalities.

Regression testing in Agile cont..

- Each sprint cycle is followed by a small span of SLRT.
- The completed code goes through further regression cycles but not released into production.
- After few successful sprint cycles-typically 3 to 4 – the application goes through one round of EERT before being released to production.

Challenges to Agile testing

- Frequent changes in agile may have crucial after-effects if necessary steps are not taken, so there is a need to perform regression testing using effective techniques so as to reduce the size of pending backlogs of user stories and the test suite.
- The changes that are introduced later may have several unnoticed effects in the working system. These effects must be controlled in a planned manner by team members to deliver the quality.

Challenges to Agile testing cont..

- In distributed Agile, testing using pair-programming practice may be cumbersome as the first team member of the pair may be at one location and the second member of the pair may be at different location.
- In this scenario, other issues also arise, such as language barrier, cultural barrier, and time zone barrier for effective communication among team members of the sprint. So quality may lag in software products.
- As the number of sprints increases, the test suite size also grows, hence, management of test cases becomes a problem in a distributed environment.

Agile Testing Tool

- TestRail – A Modern Agile Testing Tool to Boost Your Software Testing Efforts.
- Organizes test cases, manages test runs, tracks test results, and measures progress.
- Helps you meet your quality goals and complete your tests on time.

Critical points for agile testing

- Testing is very critical from agile perspective. Typically in scrum, there are many changes and test team must be capable of handling huge regression testing cycle as one progresses from iteration to iteration.
- Understanding of requirements, creation of reusable test cases, integration testing along with regression testing are key factors in successful testing .

Critical points for agile testing cont..

- Competencies/Maturity of Agile Development and Test Team
- Development and Test Process Variability
- Change Management and Communication
- Test Process Flexibility
- Focus on Business Objective
- Stakeholder Maturity/Involvement

Competencies/Maturity of Agile Development and Test Team

- For undertaking agile, one must have the teams, customer and management who psychologically accept agile approach.
- It talks about ability to change very fast, build good working product and communicate with team members and stakeholders effectively as well as efficiently.
- Agile implementation may prefer generalist approach as against specialist approach.
- It needs people with very high maturity as well as technical competence to adapt to changing needs of customer.

Development and Test Process Variability

- Every process has an inborn variability.
- One my have to attack the generic reasons of variations while there may be some controls to identify special causes of variations.
- One must be able to plot development and test process, and try to remove personal factor from the processes.

Change Management and Communication

- Change is inevitable in agile. It flows from customer to development team and goes back to customer.
- There must be a very close communication between development, test team, customer, and other stakeholders to adapt to changing scenario.
- Requirement change must be welcomed and all people together must decide how customer can be served best.

Test Process Flexibility

- Change is must in agile, and one may have to adapt to the changes. Test process is not an exception to it.
- Different parts of software need different strategies of testing.
- There may be different test plans or one may keep flexibility in a test plan to adapt to these changes. There may be changes in focus in each iteration.
- Initially, there may be heavy unit testing, then it may have integration testing where different iterations come together.
- It may be followed by heavy regression testing.

Focus on Business Objective

- There is always a time pressure in agile development.
- Pressure may come from stake holders to deliver things faster or it may come from development, if they get delayed.
- One may have to focus on business while defining test process.
- Cost-benefit analysis may be done when it come to defect fixes and release of software.
- Nobody can find all defects but user must be protected from any accidental failure.
- Testing has to achieve both extremes.

Stakeholder Maturity/Involvement

- Agile development also needs a good maturity from stakeholders.
- Internal and external service providers must understand and work with time pressure.
- Good process of development and testing must be supported by tools and techniques required for agile implementation.
- There may be some specific requirements of stake holders, and these requirements must be rearranged to suite agile development.

Summary

- Discussed the agile testing life cycle in detail.
- Highlighted the test driven development (TDD) approach.
- Discussed agile testing in different scrum phases.
- Explained regression testing in agile.
- Presented some of challenges to agile testing.
- Discussed some critical points for agile testing.

References

1. Naresh Chauhan, Software Testing: Principles and Practices, (Chapter – 16), Second Edition, Oxford University Press, 2018.

Thank You