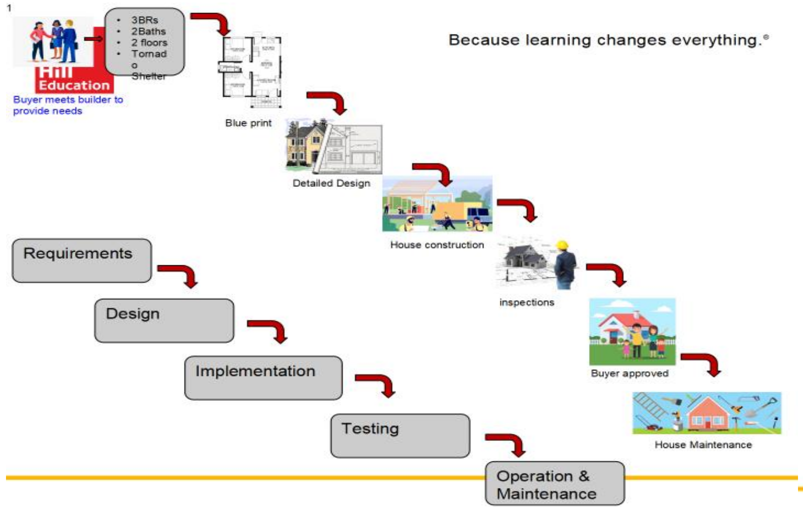
1



Because learning changes everything.®

Process Framework Activities

Communication

- The intent is to understand stakeholders' objectives for the project and to gather requirements that help define software features and functions

Planning

- Software project plan—defines the software engineering work by describing the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.

Modeling

- Analysis of requirements
- Design

Construction

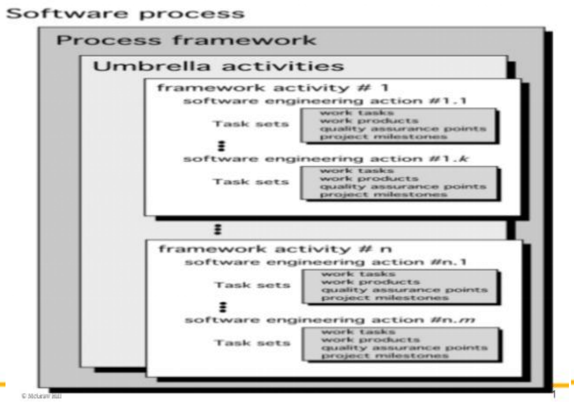
- Code generation
- Testing

Deployment

Umbrella Activities

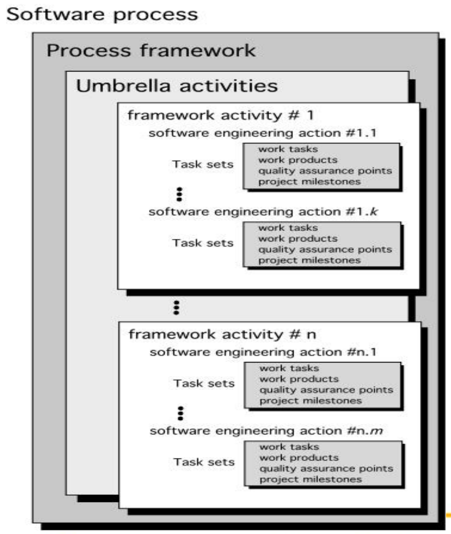
Software engineering process framework activities are complemented by a number of umbrella activities.

In general, umbrella activities are applied throughout a software project and help a software team manage and control progress, quality, change, and risk.

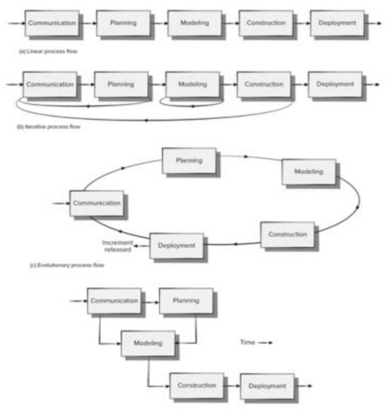


Umbrella Activities

- Software project tracking and control.
- Risk management.
- Software quality assurance.
- Technical reviews.
- Measurement.
- Software configuration management.
- Reusability management.
- Work product preparation and production.

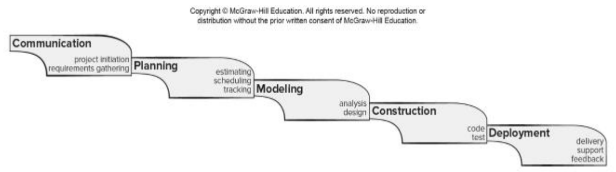


Process Flow



Access the text alternative for slide images.

Waterfall Process Model



Pros

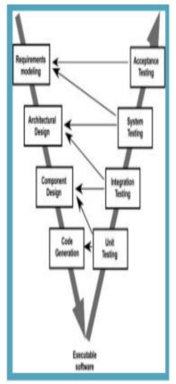
- It is easy to understand and plan.
- It works for well-understood small projects.
- Analysis and testing are straightforward.

Cons

- It does not accommodate change well.
- Testing occurs late in the process.
- Customer approval is at the end.

Access the text alternative for slide images.

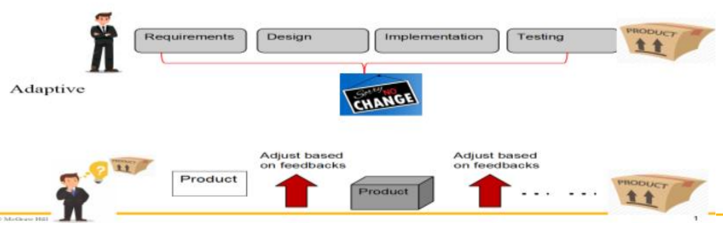
The V-Model



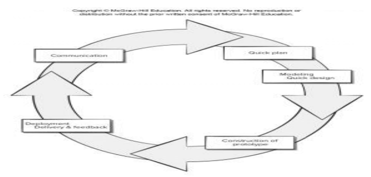
Prescriptive (Predictive) vs Adaptive

Prescriptive (Predictive)

Adaptive



Prototyping Process Model


Copyright © McGraw-Hill Education. All rights reserved. No reproduction or distribution without the prior written consent of McGraw-Hill Education.
Communication
Quick plan
Modeling Quick design
Construction of prototype
Deployment Delivery & feedback

Pros

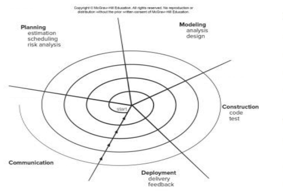
- Reduced impact of requirement changes.
- Customer is involved early and often.
- Works well for small projects.
- Reduced likelihood of product rejection.

Cons

- Customer involvement may cause delays.
- Temptation to "ship" a prototype.
- Work lost in a throwaway prototype.
- Hard to plan and manage.

Access the text alternative for slide images.

Spiral Process Model


Copyright © McGraw-Hill Education. All rights reserved. No reproduction or distribution without the prior written consent of McGraw-Hill Education.
Planning
estimation
scheduling
risk analysis
Modeling
analysis
design
Construction
code
test
start
Communication
Deployment
delivery
feedback

Pros

- Continuous customer involvement.
- Development risks are managed.
- Suitable for large, complex projects.
- It works well for extensible products.

Cons

- Risk analysis failures can doom the project.
- Project may be hard to manage.
- Requires an expert development team.

Access the text alternative for slide images.

The Agile Manifesto–a statement of values

Mc Graw Hill Education

Because learning changes everything.®

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

Source: www.agilemanifesto.org

Agility Principles 1

- Customer satisfaction is achieved by providing value through software that is delivered to the customer as rapidly as possible.
- Developers recognize that requirements will change and welcome changes.
- Deliver software increments frequently (weeks not months) to stakeholders to ensure feedback on their deliveries is meaningful.
- Agile team populated by motivated individuals using face-to-face communication to convey information.
- Team process encourages technical excellence, good design, simplicity, and avoids unnecessary work.

Agility Principles 2

- Working software that meets customer needs is the primary goal.
- Pace and direction of the team’s work must be “sustainable,” enabling them to work effectively for long periods of time.
- An agile team is a “self-organizing team”—one that can be trusted develop well-structured architectures that lead to solid designs and customer satisfaction.
- Part of the team culture is to consider its work introspectively with the intent of improving how to become more effective its primary goal (customer satisfaction).

Culture Change

Adaptive

- Deliver often and quickly
- Welcome change
- Technical excellence and good design
- Continuous improvement

⇨

Strong Engineering Culture:
- Automated build and deployment
- Continuous integration
- Automated tests
- Collective ownership

Adaptive Planning

People and Interaction

- Developer and business personnel work together
- Face-to-face conversation
- Self organizing teams
- Motivated individuals

⇨

- Mutual trust and respect
- Customer is a part of the team
- Collaborative environment
- Cross functional team

How Agile Helps

The most widely used agile process, originally proposed by Kent Beck

XP Planning Begins with the creation of "user stories"

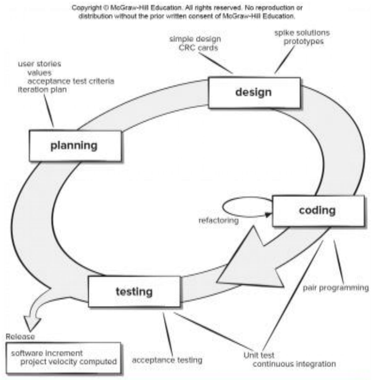
User story is the unit of functionality

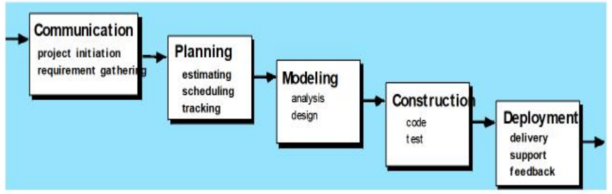
The customer assigns a value (i. e., a priority) to each story based on the overall business value of the feature or function

Agile team assesses each story and assigns a cost

At least two members participate in the assessment

Measured in development weeksWhat if a story is estimated to require a long time, e.g., more than three development weeks?





XP Details

- **XP Planning** – Begins with user stories, team estimates cost, stories grouped into increments, commitment made on delivery date, compute project velocity.
- **XP Design** – Follows KIS principle, encourages use of CRC cards, design prototypes, and refactoring.
- **XP Coding** – construct unit tests before coding, uses pair.
- **XP Testing** – unit tests executed daily, acceptance tests define by customer.

Pros

- Emphasizes customer involvement.
- Establishes rational plans and schedules.
- High developer commitment to the project.
- Reduced likelihood of product rejection.

Cons

- Temptation to “ship” a prototype.
- Requires frequent meetings increasing costs.
- Allows for excessive changes.
- Depends on highly skilled team members.

Kanban Details

- Visualizing workflow using a Kanban board.
- Limiting the amount of *work in progress* at any given time.
- Managing workflow to reduce waste by understanding the current value flow.
- Making process policies explicit and the criteria used to define “done”.
- Focusing on continuous improvement by creating feedback loops where changes are introduced.
- Make process changes collaboratively and involve all stakeholders as needed.

Pros

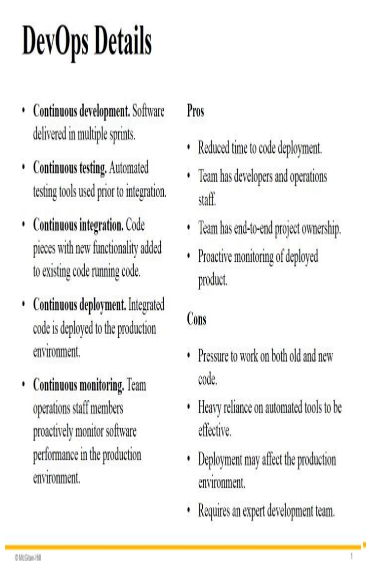
- Lower budget and time requirements.
- Allows early product delivery.
- Process policies written down.
- Continuous process improvement.

Cons

- Team collaboration skills determine success.
- Poor business analysis can doom the project.
- Flexibility can cause developers to lose focus.
- Developer reluctance to use measurement.

The Refactoring Process

- Make a small change - a single refactoring
- Run all the tests to ensure everything still works
- If everything works, move on to the next refactoring
- If not, fix the problem, or undo the change, so you still have a working system



DevOps Details

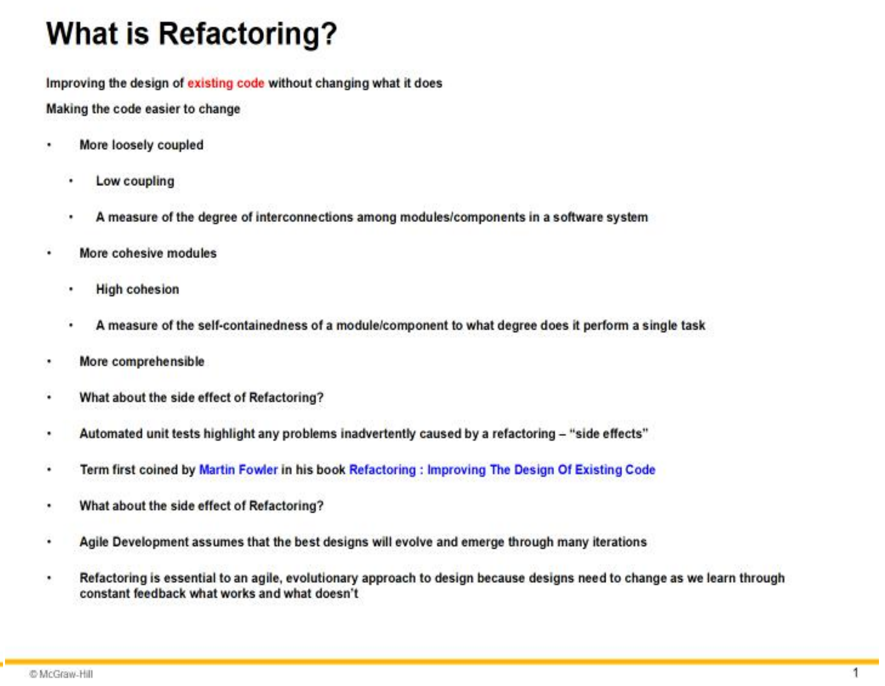
- **Continuous development.** Software delivered in multiple sprints.
- **Continuous testing.** Automated testing tools used prior to integration.
- **Continuous integration.** Code pieces with new functionality added to existing code running code.
- **Continuous deployment.** Integrated code is deployed to the production environment.
- **Continuous monitoring.** Team operations staff members proactively monitor software performance in the production environment.

Pros

- Reduced time to code deployment.
- Team has developers and operations staff.
- Team has end-to-end project ownership.
- Proactive monitoring of deployed product.

Cons

- Pressure to work on both old and new code.
- Heavy reliance on automated tools to be effective.
- Deployment may affect the production environment.
- Requires an expert development team.

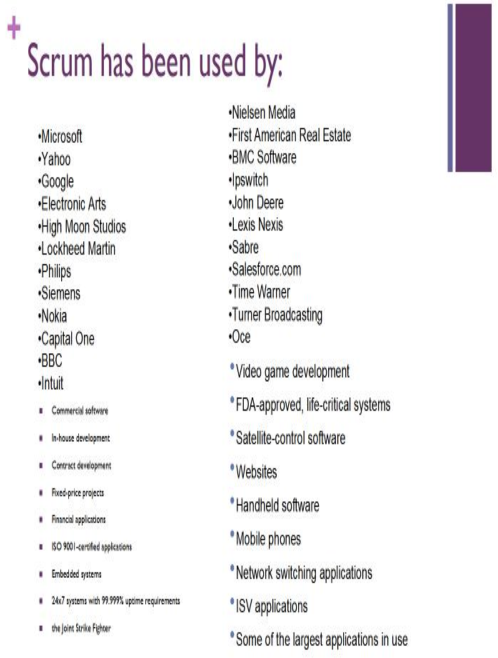


What is Refactoring?

Improving the design of existing code without changing what it does

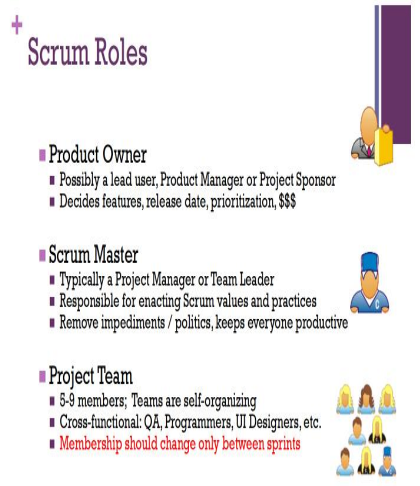
Making the code easier to change

- **More loosely coupled**
  - **Low coupling**
  - **A measure of the degree of interconnections among modules/components in a software system**
- **More cohesive modules**
  - **High cohesion**
  - **A measure of the self-containedness of a module/component to what degree does it perform a single task**
- **More comprehensible**
- **What about the side effect of Refactoring?**
- **Automated unit tests highlight any problems inadvertently caused by a refactoring – "side effects"**
- **Term first coined by Martin Fowler in his book Refactoring : Improving The Design Of Existing Code**
- **What about the side effect of Refactoring?**
- **Agile Development assumes that the best designs will evolve and emerge through many iterations**
- **Refactoring is essential to an agile, evolutionary approach to design because designs need to change as we learn through constant feedback what works and what doesn't**



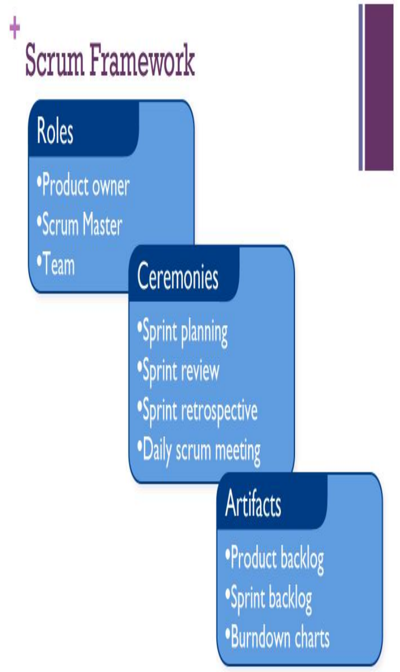
Scrum has been used by:

- •Microsoft
- •Yahoo
- •Google
- •Electronic Arts
- •High Moon Studios
- •Lockheed Martin
- •Philips
- •Siemens
- •Nokia
- •Capital One
- •BBC
- •Intuit
  - Commercial software
  - In-house development
  - Contract development
  - Fixed-price projects
  - Financial applications
  - ISO 9001-certified applications
  - Embedded systems
  - 24x7 systems with 99.999% uptime requirements
  - the Joint Strike Fighter
- •Nielsen Media
- •First American Real Estate
- •BMC Software
- •Ipswitch
- •John Deere
- •Lexis Nexis
- •Sabre
- •Salesforce.com
- •Time Warner
- •Turner Broadcasting
- •Oce
- Video game development
- FDA-approved, life-critical systems
- Satellite-control software
- Websites
- Handheld software
- Mobile phones
- Network switching applications
- ISV applications
- Some of the largest applications in use



Scrum Roles

- Product Owner
  - Possibly a lead user, Product Manager or Project Sponsor
  - Decides features, release date, prioritization, $$$
- Scrum Master
  - Typically a Project Manager or Team Leader
  - Responsible for enacting Scrum values and practices
  - Remove impediments / politics, keeps everyone productive
- Project Team
  - 5-9 members; Teams are self-organizing
  - Cross-functional: QA, Programmers, UI Designers, etc.
  - Membership should change only between sprints

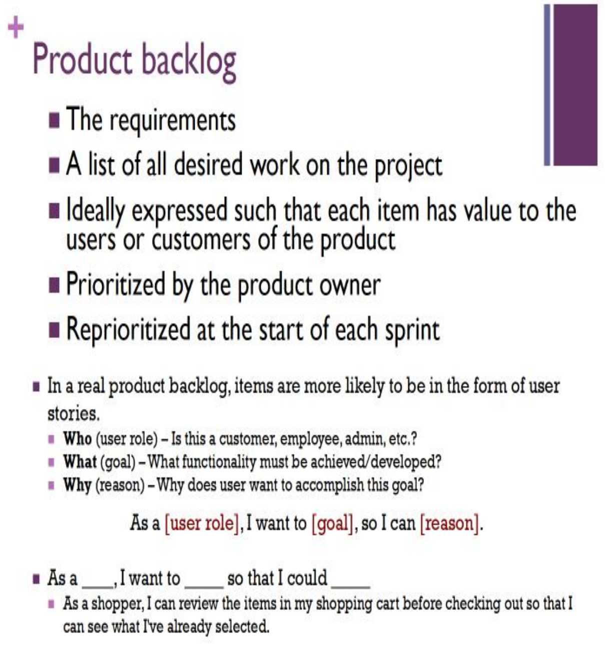


Scrum Framework

Roles
- •Product owner
- •Scrum Master
- •Team

Ceremonies
- •Sprint planning
- •Sprint review
- •Sprint retrospective
- •Daily scrum meeting

Artifacts
- •Product backlog
- •Sprint backlog
- •Burndown charts



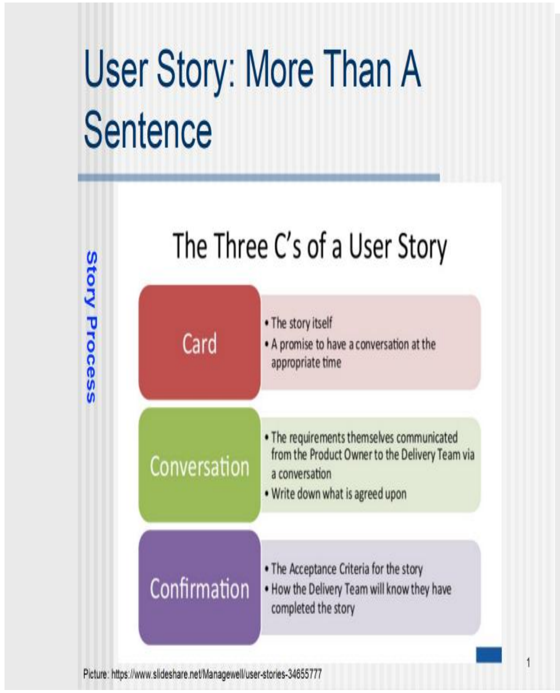
Product backlog

- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint

- In a real product backlog, items are more likely to be in the form of user stories.
  - **Who** (user role) – Is this a customer, employee, admin, etc.?
  - **What** (goal) – What functionality must be achieved/developed?
  - **Why** (reason) – Why does user want to accomplish this goal?

As a [user role], I want to [goal], so I can [reason].

- As a ____, I want to ____ so that I could ____
  - As a shopper, I can review the items in my shopping cart before checking out so that I can see what I've already selected.



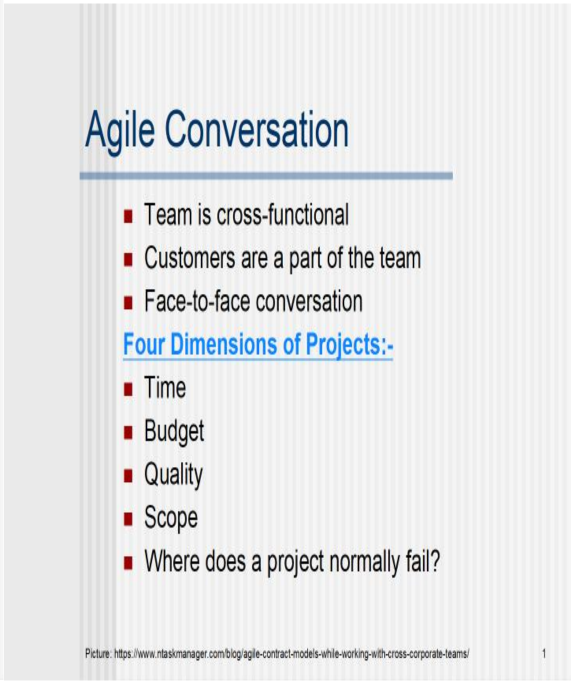
User Story: More Than A Sentence

Story Process

The Three C's of a User Story

Card	• The story itself • A promise to have a conversation at the appropriate time
Conversation	• The requirements themselves communicated from the Product Owner to the Delivery Team via a conversation • Write down what is agreed upon
Confirmation	• The Acceptance Criteria for the story • How the Delivery Team will know they have completed the story

Picture: https://www.slideshare.net/Managewell/user-stories-34655777 1



Agile Conversation

- Team is cross-functional
- Customers are a part of the team
- Face-to-face conversation

Four Dimensions of Projects:-

- Time
- Budget
- Quality
- Scope
- Where does a project normally fail?

Picture: https://www.ntaskmanager.com/blog/agile-contract-models-while-working-with-cross-corporate-teams/ 1

Team Leader

- The MOI Model
  - **Motivation.** The ability to encourage (by "push or pull") technical people to produce to their best ability.
  - **Organization.** The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
  - **Ideas or innovation.** The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

Characteristics of Good User Stories

- INVEST
  - Independent – User Stories should be as independent as possible.
  - Negotiable – a User Story is not a contract. It is not a detailed specification. It is a reminder of features for the team to discuss and collaborate to clarify the details near the time of development.
  - Valuable – User Stories should be valuable to the user (or owner) of the solution. They should be written in user language. They should be features, not tasks.
  - Estimatable – User Stories need to be possible to estimate. They need to provide enough information to estimate, without being too detailed.
  - Small – User Stories should be small. Not too small and not too big.
  - Testable – User Stories need to be worded in a way that is testable, i.e. not too subjective and to provide clear details of how the User Story will be tested.

Prioritize Stories in a Backlog

- A collection of stories for a software product is referred to as the product backlog
- Agile customers or product owner prioritize stories in a backlog
- The backlog is prioritized so that the most valuable items have the highest priorities

Effective Software Project Management Focuses on the Four P's

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

Stakeholders

- *Senior managers* who define the business issues that often have significant influence on the project.
- *Project (technical) managers* who must plan, motivate, organize, and control the practitioners who do software work.
- *Practitioners* who deliver the technical skills that are necessary to engineer a product or application.
- *Customers* who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- *End-users* who interact with the software once it is released for production use.

Software Teams

The following factors must be considered when selecting a software project team structure ...

- the difficulty of the problem to be solved
- the size of the resultant program(s) in lines of code or function points
- the time that the team will stay together (team lifetime)
- the degree to which the problem can be modularized
- the required quality and reliability of the system to be built
- the rigidity of the delivery date
- the degree of sociability (communication) required for the project

Agile Teams

- Team members must have trust in one another.
- The distribution of skills must be appropriate to the problem.
- Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.
- Team is "self-organizing"
  - An adaptive team structure
  - Significant autonomy

Project Management Terminology

– Task,Duration,Start and Finish Dates,Predecessor and Successor,Resources,Project Manager,Scope, Quality, Risk.

Software scope describes

Functions and features to be delivered to end-users.

Data input and output.

Content presented to users of using the software.

Performance, constraints, interfaces, and reliability that *bound* the system.

Scope is defined using one of two techniques:

A narrative description of software scope is developed after communication with all stakeholders.

A set of use-cases is developed by end-users