THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Agile Development

Slide Set - 13

Organized & Presented By:
Software Engineering Team CSED
TIET, Patiala

The 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."

Kent Beck et al

What is “Agility”?

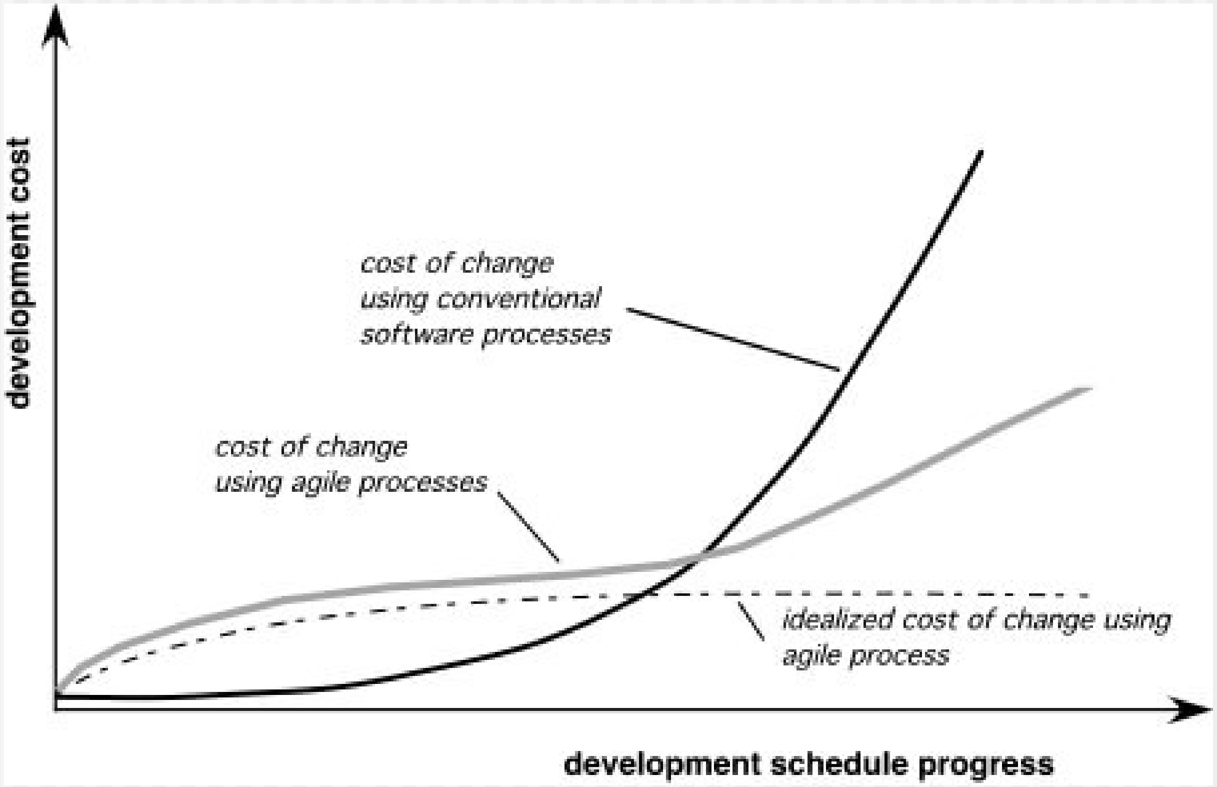
- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software

These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 7/e*
(McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

Agility and the Cost of Change



These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

An Agile Process

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers multiple ‘software increments’
- Adapts as changes occur

These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

Agility Principles - I

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

Agility Principles - II

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

Human Factors

- *the process molds to the needs of the people and team,* not the other way around
- key traits must exist among the people on an agile team and the team itself:
  - **Competence.**
  - **Common focus.**
  - **Collaboration.**
  - **Decision-making ability.**
  - **Fuzzy problem-solving ability.**
  - **Mutual trust and respect.**
  - **Self-organization.**

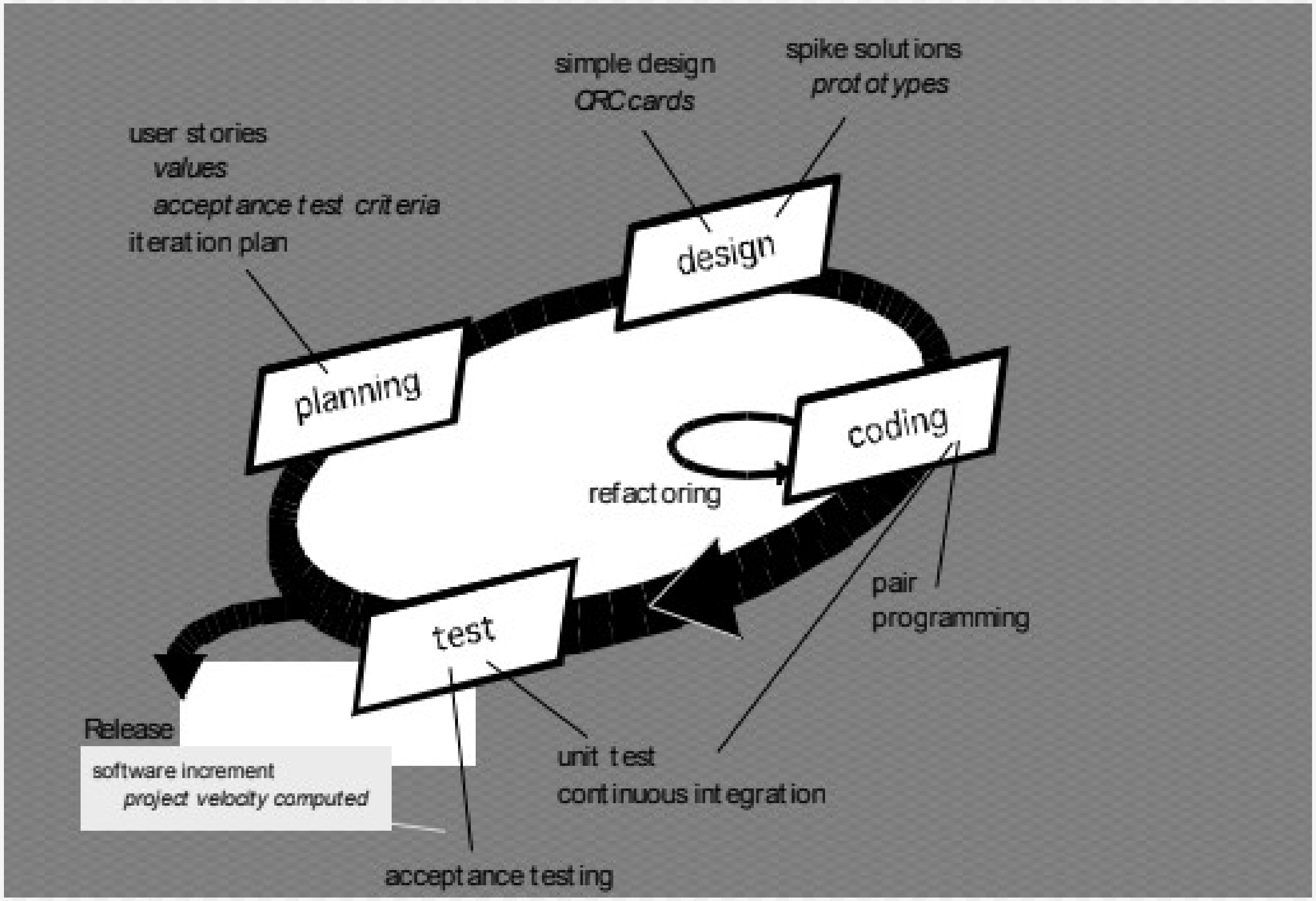
Extreme Programming (XP)

- The most widely used agile process, originally proposed by Kent Beck
- 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
  - After the first increment "project velocity" is used to help define subsequent delivery dates for other increments

Extreme Programming (XP)

- XP Design
  - Follows the KIS principle
  - Encourage the use of CRC cards (see Chapter 8)
  - For difficult design problems, suggests the creation of “spike solutions”—a design prototype
  - Encourages “refactoring”—an iterative refinement of the internal program design
- XP Coding
  - Recommends the construction of a unit test for a store *before* coding commences
  - Encourages “pair programming”
- XP Testing
  - All unit tests are executed daily
  - “Acceptance tests” are defined by the customer and excuted to assess customer visible functionality

Extreme Programming (XP)



These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

Scrum

- Originally proposed by Schwaber and Beedle
- Scrum—distinguishing features
  - Development work is partitioned into “packets”
  - Testing and documentation are on-going as the product is constructed
  - Work occurs in “sprints” and is derived from a “backlog” of existing requirements
  - Meetings are very short and sometimes conducted without chairs
  - “demos” are delivered to the customer with the time-box allocated

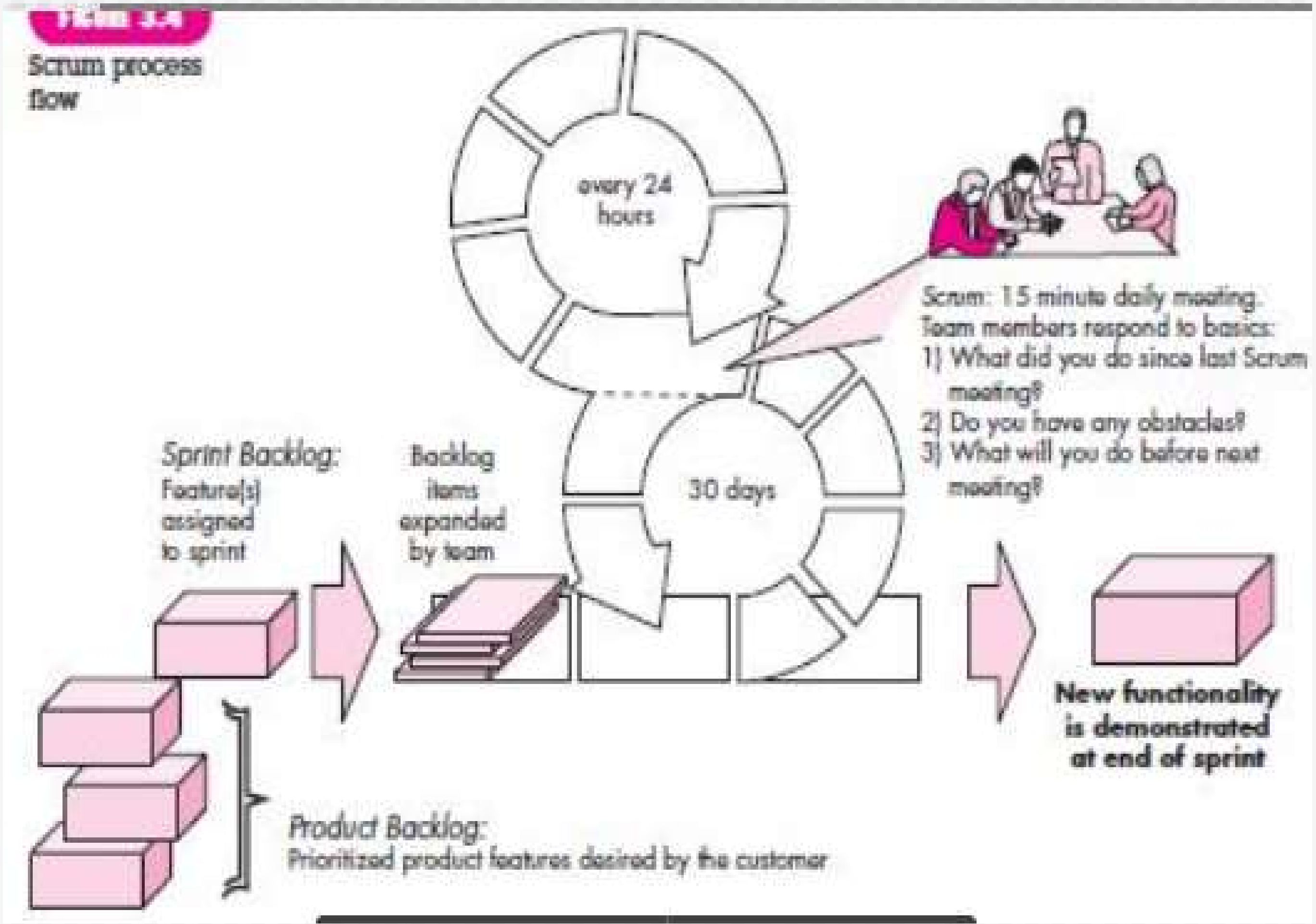
These slides are designed to accompany *Software Engineering: A Practitioner’s Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

Applications of SCRUM

- **Scrum** is a widely-**used**, agile product development strategy—a collection of values, team roles, and rituals (defined below) **used** in combination to create iterative work products.

- **Scrum** began in the software industry and has since spread to universities, Teaching and Learning, Research Industry, military, the automotive industry, and getting its place and adoption at every other industry for managing related projects in efficient way.

Scrum

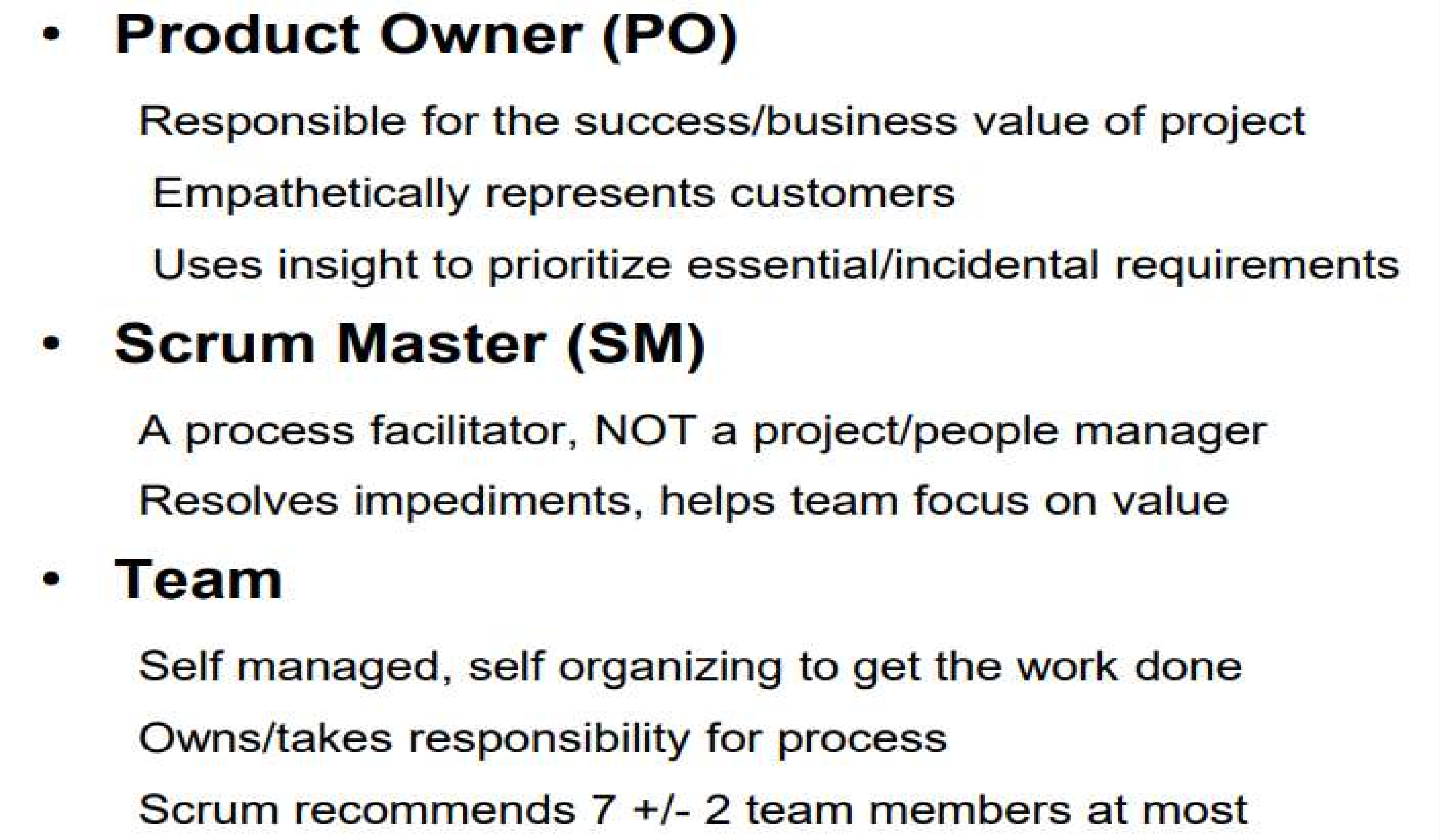


Scrum process flow

every 24 hours
30 days
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Scrum: 15 minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?

SCRUM Roles



- **Product Owner (PO)**

  Responsible for the success/business value of project

  Empathetically represents customers

  Uses insight to prioritize essential/incidental requirements

- **Scrum Master (SM)**

  A process facilitator, NOT a project/people manager

  Resolves impediments, helps team focus on value

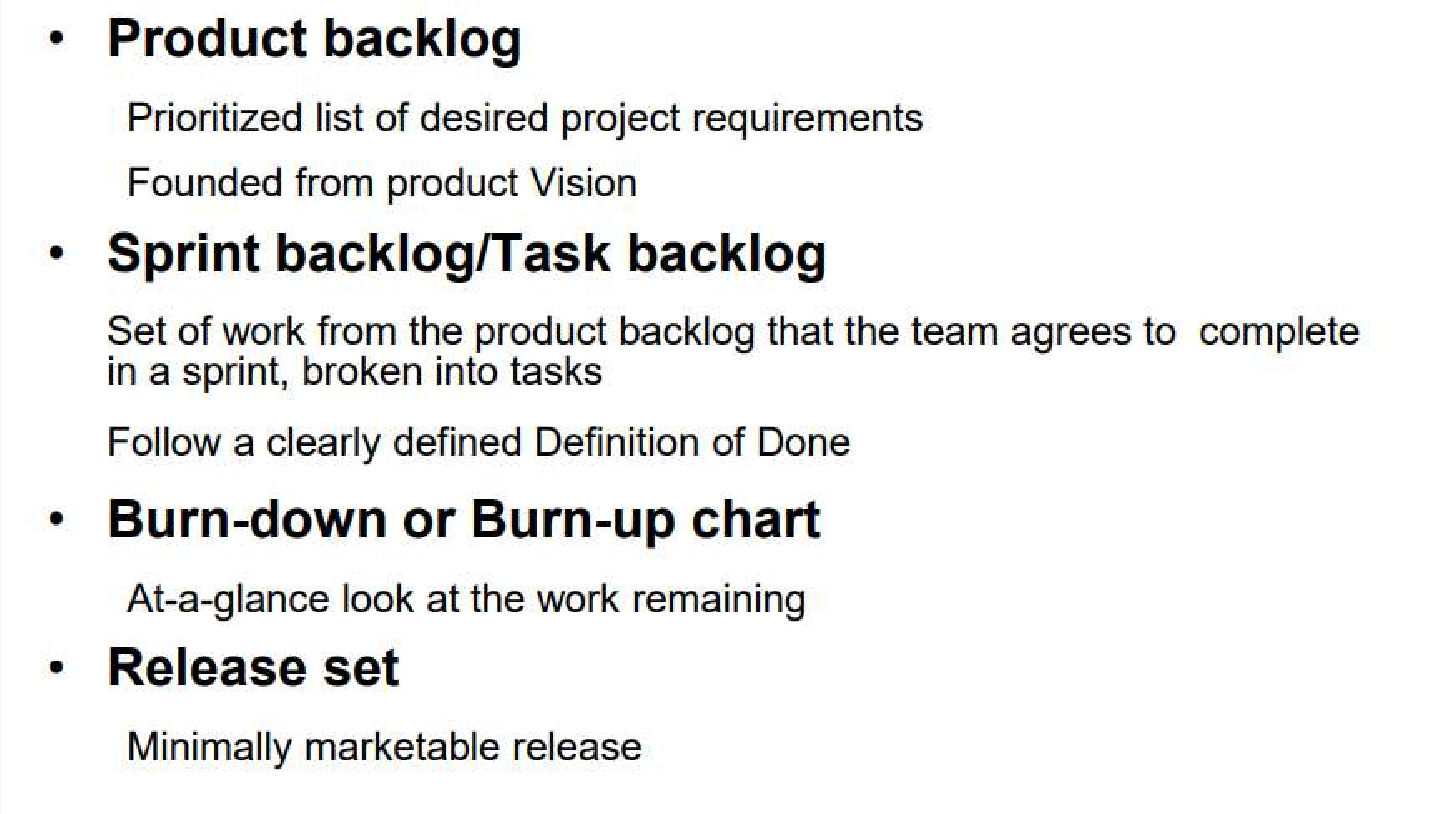
- **Team**

  Self managed, self organizing to get the work done

  Owns/takes responsibility for process

  Scrum recommends 7 +/- 2 team members at most

SCRUM Artifacts



- **Product backlog**

  Prioritized list of desired project requirements

  Founded from product Vision

- **Sprint backlog/Task backlog**

  Set of work from the product backlog that the team agrees to complete in a sprint, broken into tasks

  Follow a clearly defined Definition of Done

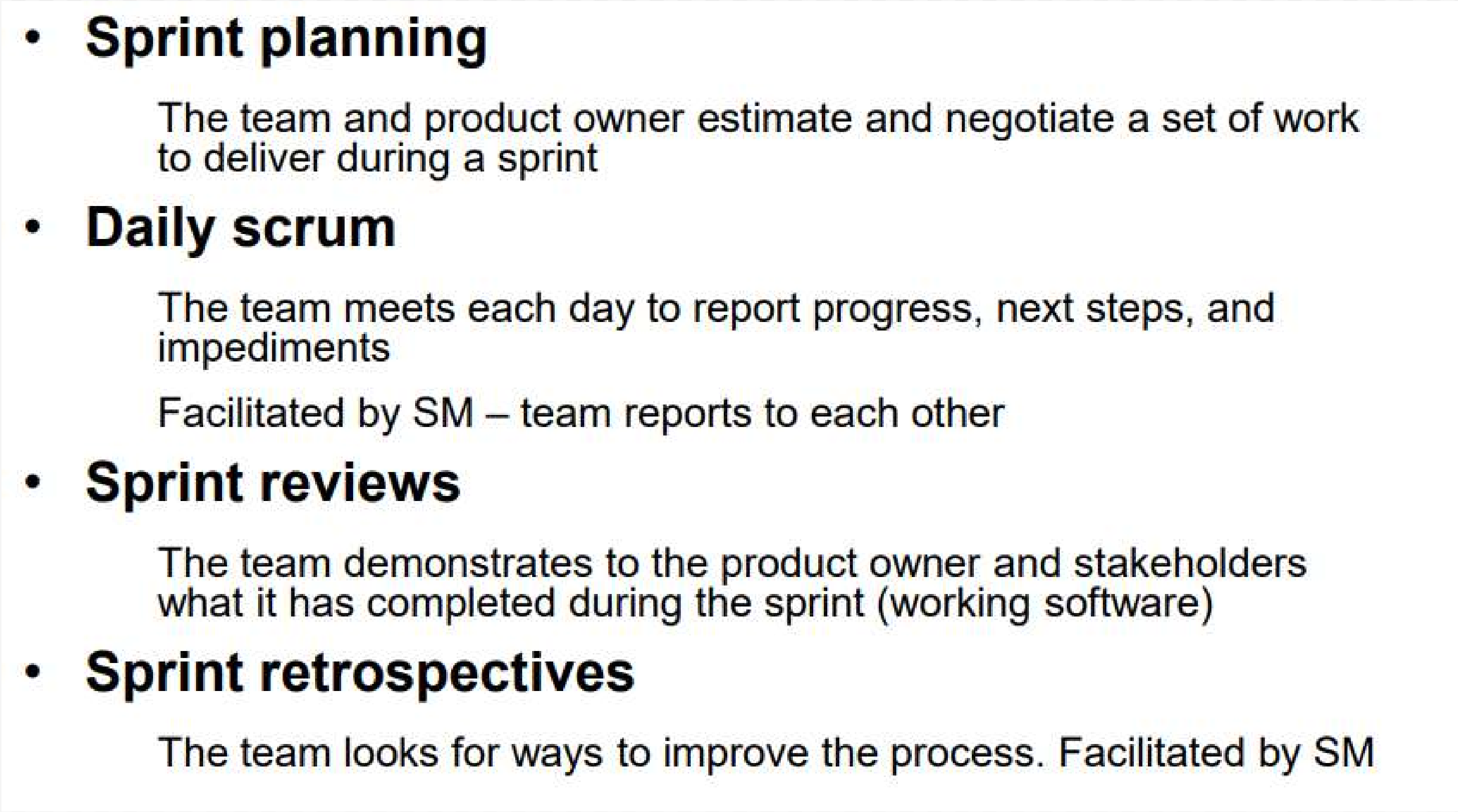
- **Burn-down or Burn-up chart**

  At-a-glance look at the work remaining

- **Release set**

  Minimally marketable release

SCRUM Ceremonies(Meetings)



- **Sprint planning**

  The team and product owner estimate and negotiate a set of work to deliver during a sprint

- **Daily scrum**

  The team meets each day to report progress, next steps, and impediments

  Facilitated by SM – team reports to each other

- **Sprint reviews**

  The team demonstrates to the product owner and stakeholders what it has completed during the sprint (working software)

- **Sprint retrospectives**

  The team looks for ways to improve the process. Facilitated by SM

User Stories

A concise, written description of a piece of functionality that will be valuable to a user (or owner) of the software. A user Story card has 3 parts.

1. Card - A written description of the user story for planning purposes and as a reminder

2. Conversation - A section for capturing further information about the user story and details of any conversations

3. Confirmation - A section to convey what tests will be carried out to confirm the user story is complete and working as expected

User Story Description

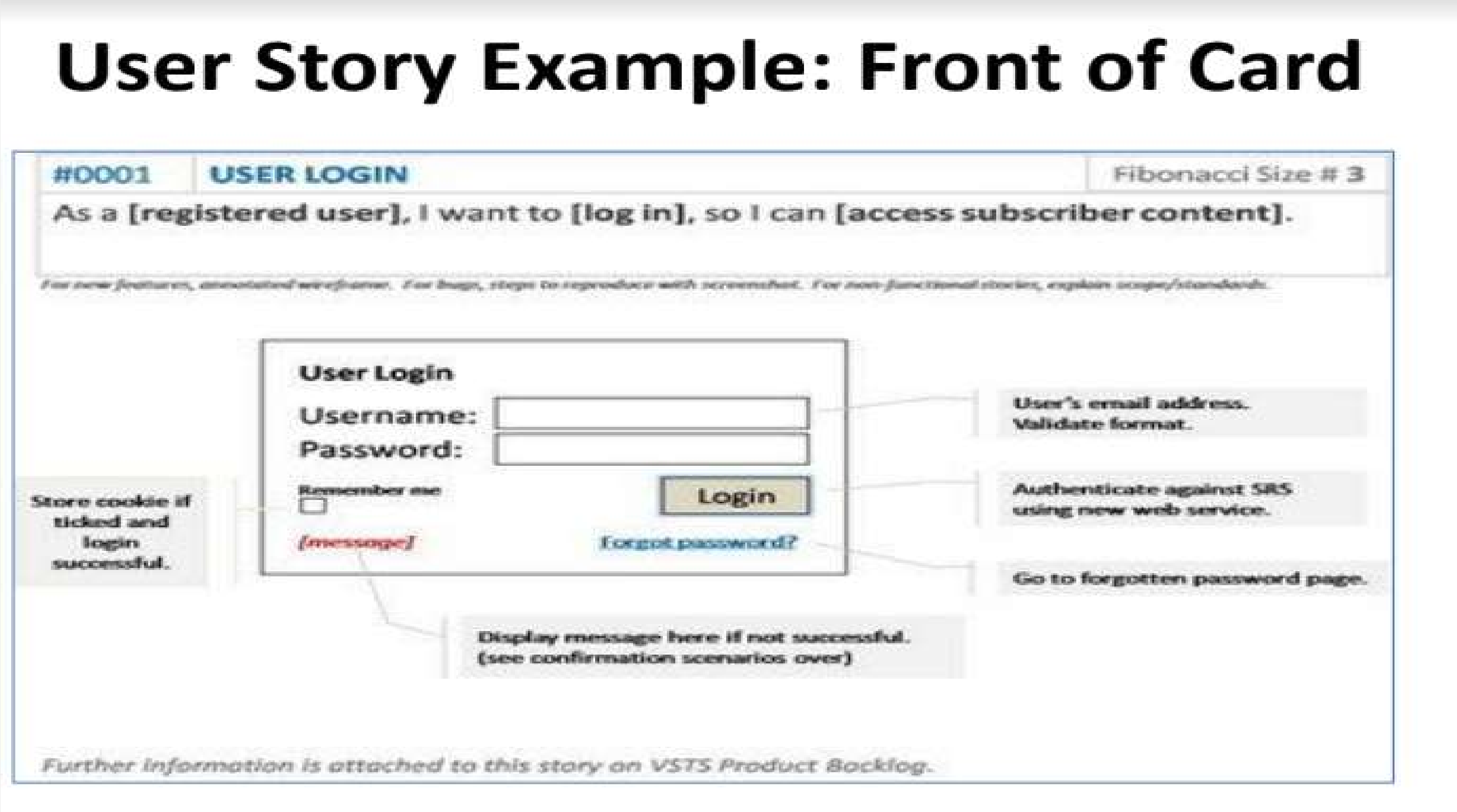
As a [user role] I want to [goal] so I can [reason]

For example:

- As a registered user I want to log in
so I can access subscriber-only content

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill, 2009) Slides copyright 2009 by Roger Pressman.

User Story Example: Front of Card



User Story Example: Front of Card

#0001 | USER LOGIN | Fibonacci Size # 3

As a **[registered user]**, I want to **[log in]**, so I can **[access subscriber content]**.

For new features, annotated wireframe. For bugs, steps to reproduce with screenshot. For non-functional stories, explain scope/standards.

User's email address. Validate format.

Authenticate against SRS using new web service.

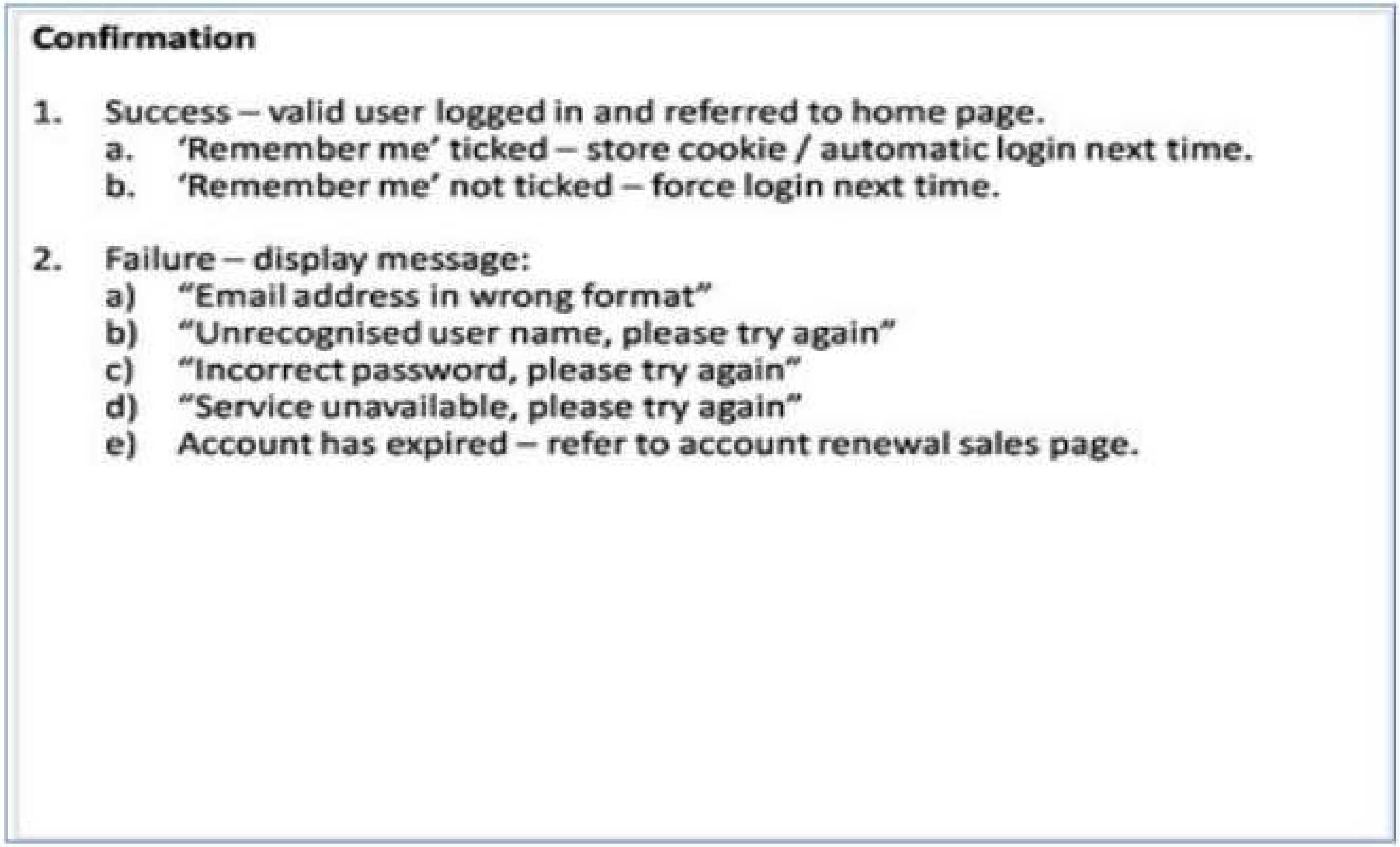
Go to forgotten password page.

Store cookie if ticked and login successful.

Display message here if not successful. (see confirmation scenarios over)

Further information is attached to this story on VSTS Product Backlog.

User Story Example: Back of Card



Confirmation

1. Success – valid user logged in and referred to home page.
   a. 'Remember me' ticked – store cookie / automatic login next time.
   b. 'Remember me' not ticked – force login next time.

2. Failure – display message:
   a) "Email address in wrong format"
   b) "Unrecognised user name, please try again"
   c) "Incorrect password, please try again"
   d) "Service unavailable, please try again"
   e) Account has expired – refer to account renewal sales page.