31269: Business Requirements Modeling

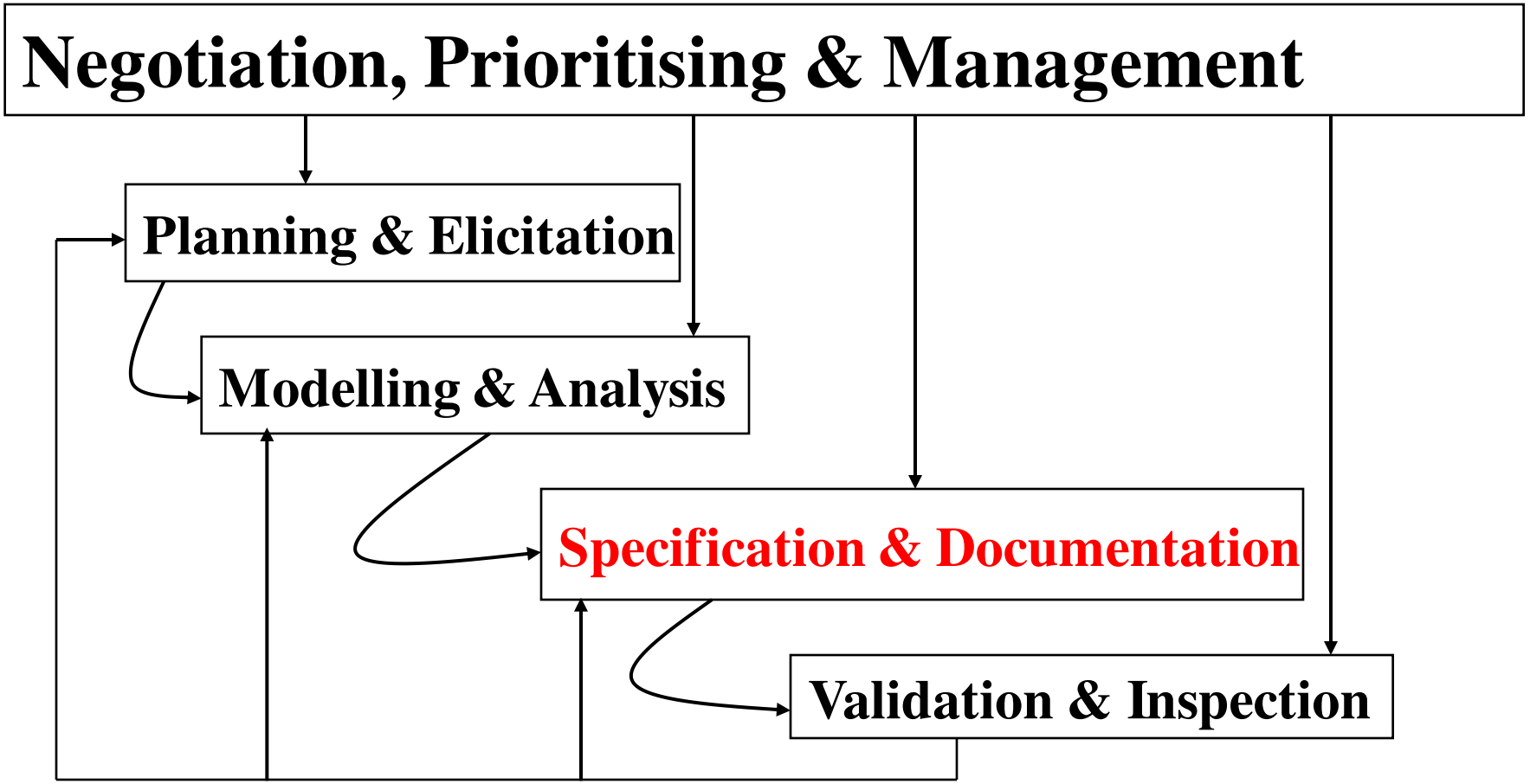
Week 6 Lecture - Software Requirements Specification and Documentation

- References
  - Mastering The Requirements Process Ch 16
  - BABOK Guide Version 2.0 Ch 6
- Acknowledgement
  - Professor Didar Zowghi's Lecture in Spring 2013

Objectives

- Understand the difference between agile and traditional software requirement specification (IEEE SRS).
- Discover how user stories can be used to **specify software requirements**
- Understand how user stories are estimated and prioritised.

Requirements Process



Requirements Analysis

- Last few weeks - How to analyse and model the elicited requirements?
  - Process Model (BPD), Data Model (ERD)

- This week - How do we analyse these process and data models and **document the software requirements?**
  - How do we use these business processes to identify the user stories?

Topics

- Software Requirements Specification (SRS)
- SRS content and structure
- Traditional IEEE SRS vs User Stories (Agile approach)

Software Requirements Specification (SRS) definition

Software Requirements Specification is a document that provides the detailed description of **what the system should do**.

- structured document setting out the **system services and capabilities in detail**
- sometimes called as functional specification
- often serves as a **contract** between the client and vendor

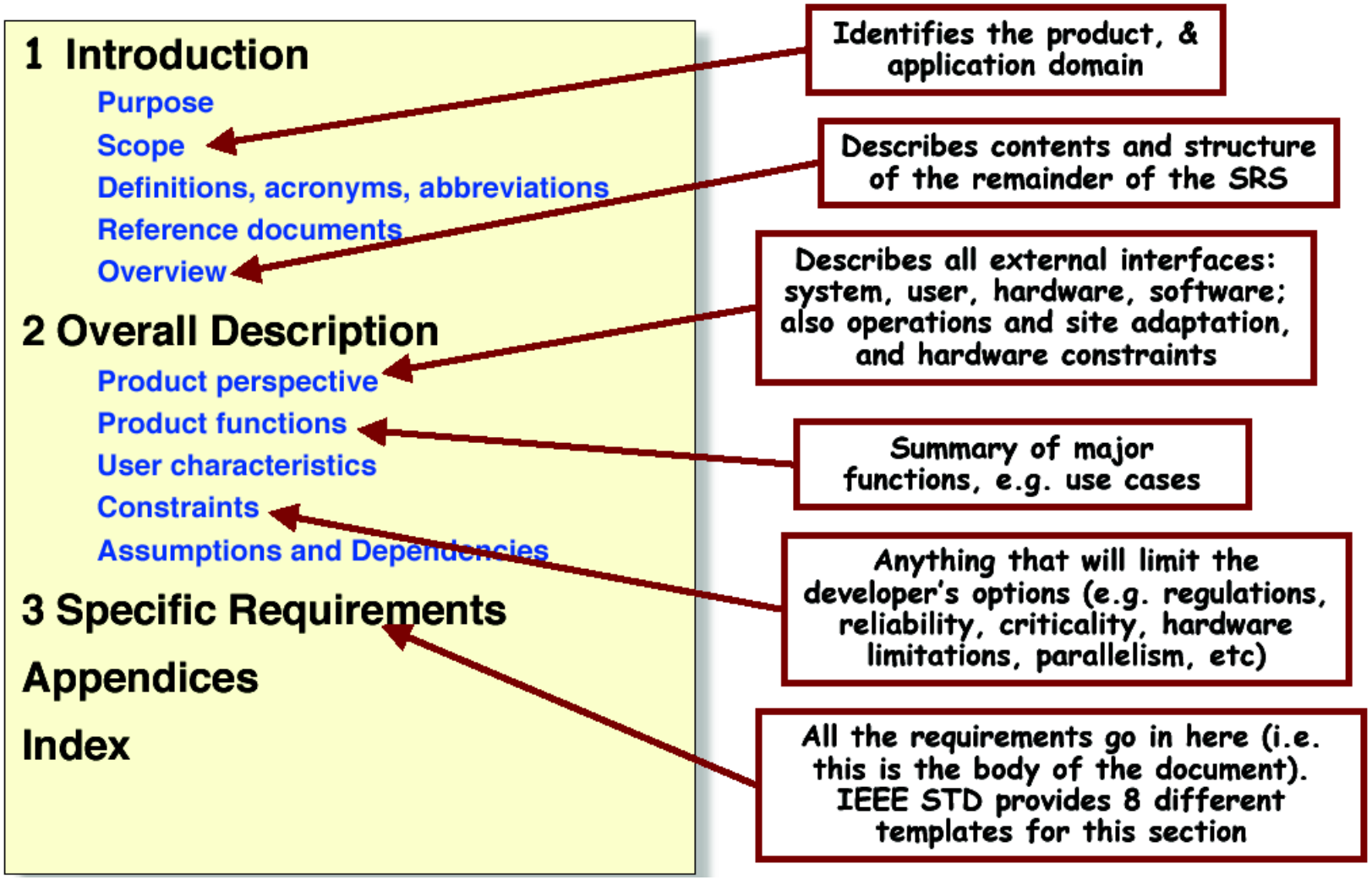
Purpose of Software Requirements Specifications (SRS)

- To analyse the elicited requirements
- Define what designers/developers have to build
- Verification against the delivered system
- Validating that they are indeed what stakeholders want
- Baseline for evaluating the software
- Support for testing (verification and validation)

SRS: Role and Purpose

- A legal document. A **contractual device** for judging the completion of the specified job.
- **Functional device** for improving the understanding of the customer's real needs (both in business and technical terms). So it is a **communication device** that conveys an understanding between different teams in the software development process.
- Used as basis for a **user manual.**
- Statement of commitment. **Validation device** for validating the requirements stated in a formal manner.
- Used to develop **test cases**.

IEEE Standard for SRS



Source: Adapted from IEEE-STD-830-1993

IEEE standard section 3 example

- 3.1 External Interface Requirements
  - 3.1.1 User Interfaces
  - 3.1.2 Hardware Interfaces
  - 3.1.3 Software Interfaces
  - 3.1.4 Communication Interfaces
- 3.2 Functional Requirements
  - *this section organized by mode, user class, feature, etc. For example:*
  - 3.2.1 Mode 1
    - *3.2.1.1 Functional Requirement 1.1*
    - …
  - 3.2.2 Mode 2
    - *3.2.1.1 Functional Requirement 1.1*
    - …
  - …
  - 3.2.2 Mode n
    - …
- 3.3 Performance Requirements
  - *Remember to state this in measurable terms!*
- 3.4 Design Constraints
  - *3.4.1 Standards compliance*
  - *3.4.2 Hardware limitations*
  - *etc.*
- 3.5 Software System Attributes
  - 3.5.1 Reliability
  - 3.5.2 Availability
  - 3.5.3 Security
  - 3.5.4 Maintainability
  - 3.5.5 Portability
- 3.6 Other Requirements

Source: Adapted from IEEE-STD-830-1993

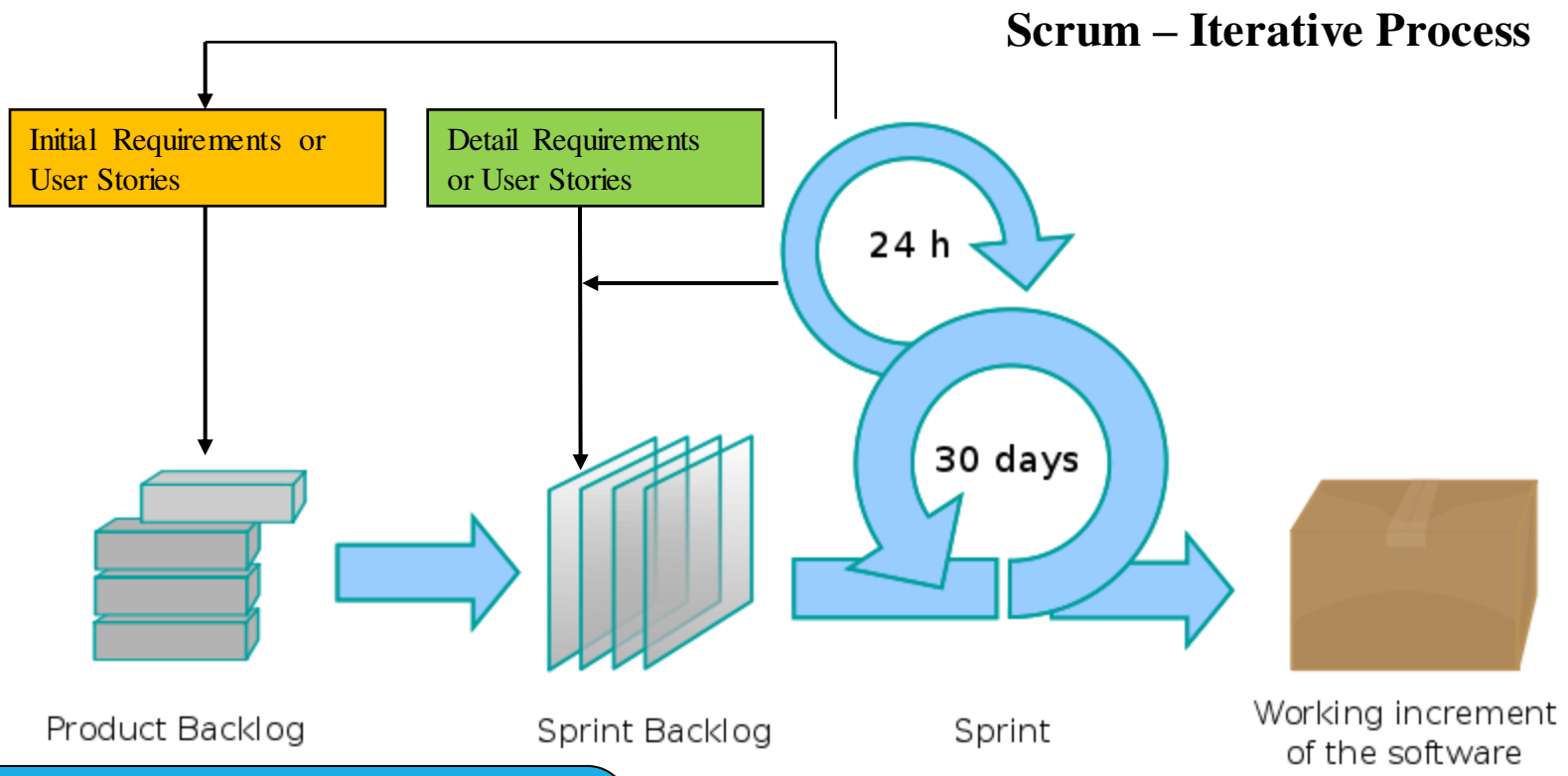
Agile Requirements Specification: Template adapted and used in this subject

Requirements Specification

- Traditional aspects
  - Upfront and detailed
  - Simple plain language requirements statements
  - Structured use cases (next week's lecture)

- Agile aspects
  - Iterative
  - User stories: card, conversation, confirmation
  - User story map
  - User story estimation and prioritisation
  - User story or agile card wall

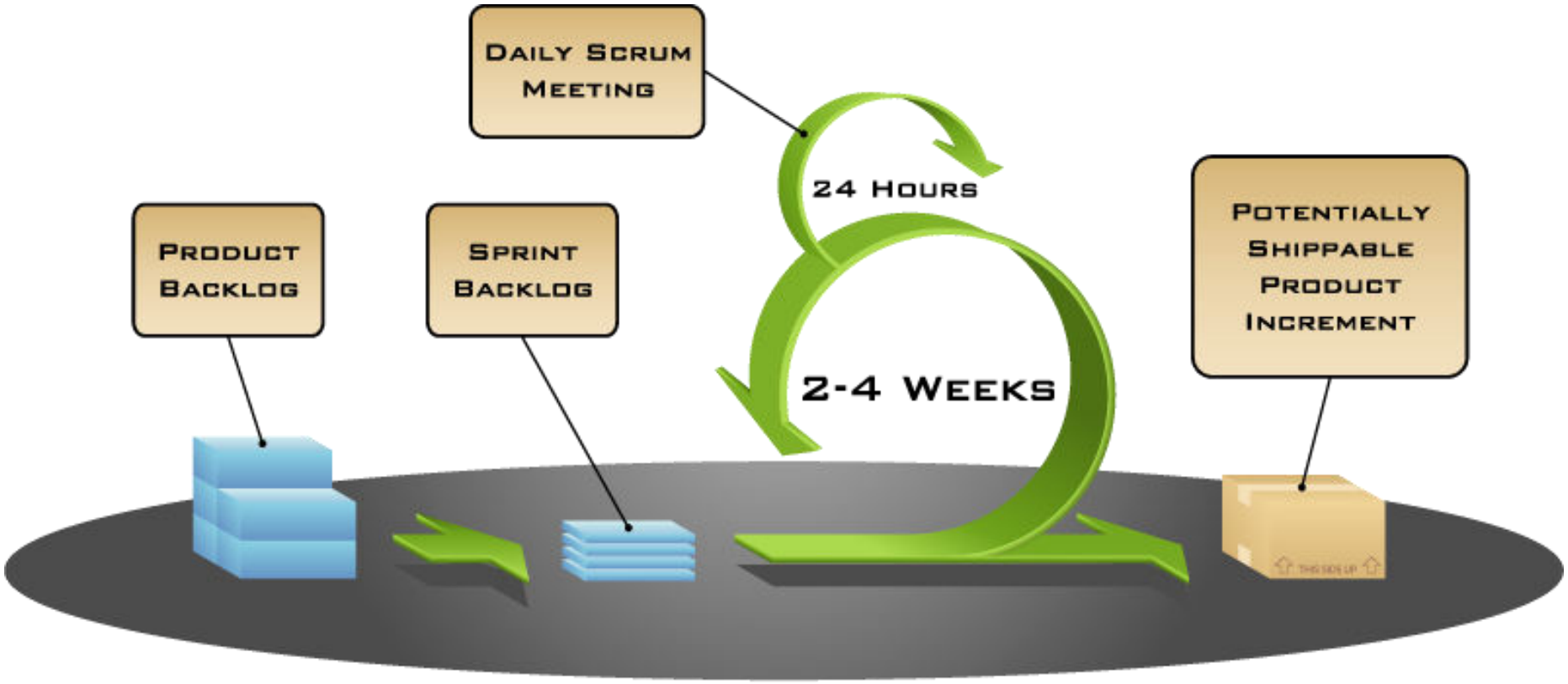
Agile System Development Process



Please watch the video available on uts online – "Intro to Agile Scrum in Under 10 Minutes"

Source: http://en.wikipedia.org/wiki/Scrum_(development)

SCRUM at a glance



Sprints

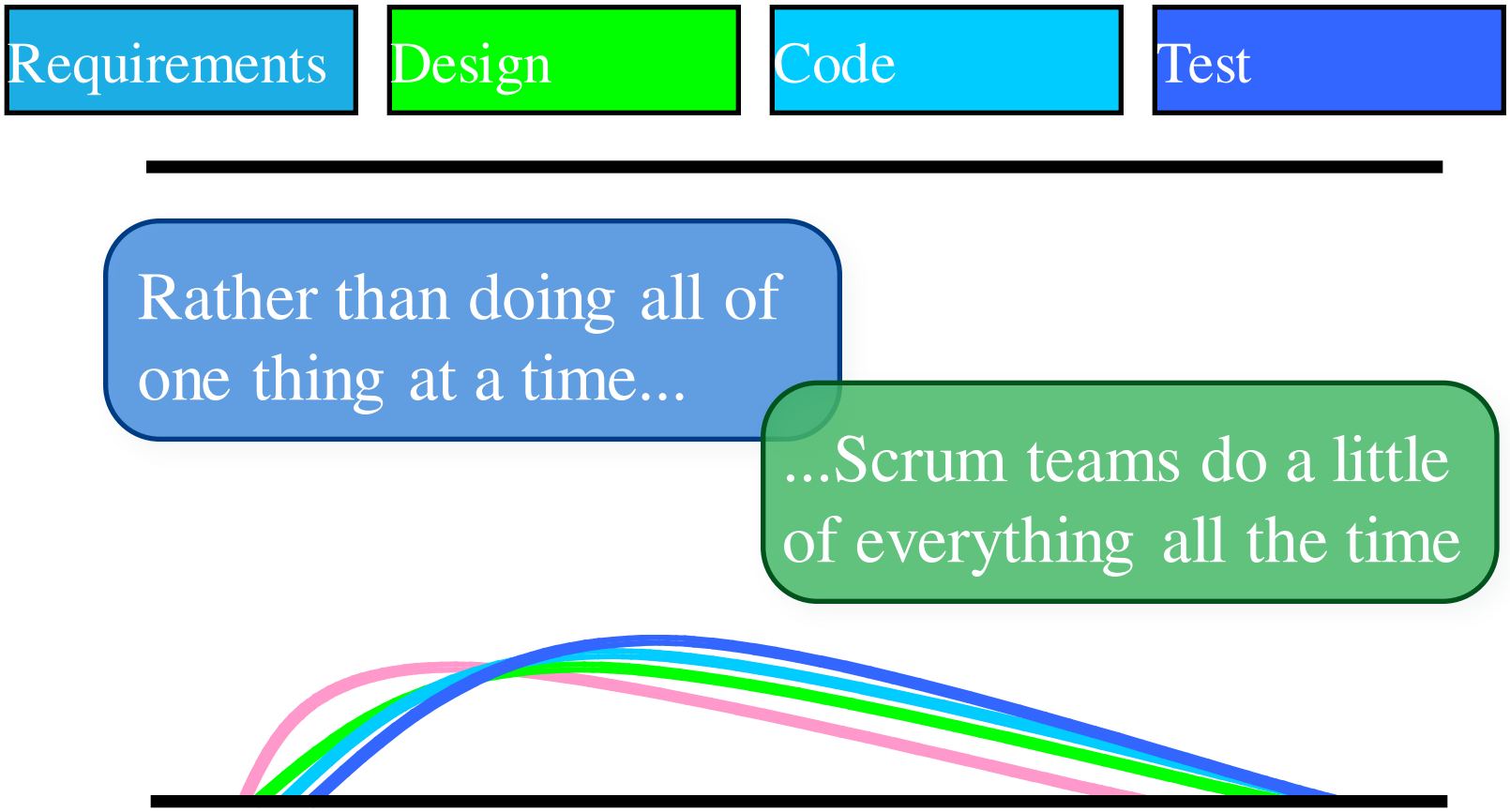
- A sprint (or iteration) is the basic unit of development in Scrum. Scrum projects make progress in a series of "sprints".
  - Analogous to Extreme Programming iterations
- Typical duration is 2–4 weeks or a calendar month at most
- A constant duration leads to a better rhythm.
- Product is designed, coded, and tested during the sprint.
- Each sprint starts with a sprint planning event, the aim of which is to define a sprint backlog, where the work for the sprint is identified and an estimated commitment for the sprint goal is made. Each sprint ends with a sprint review and a sprint retrospective, where the progress is reviewed and lessons for the next sprint are identified.

Iterative and incremental development

- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during development of earlier parts or versions of the system.
- Learning comes from both the development and use of the system, where possible key steps in the process start with a simple implementation of a subset of the software requirements and iteratively enhance the evolving versions until the full system is implemented.
- **Incremental** development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross-discipline work, from the requirements to the deployment.

(http://en.wikipedia.org/wiki/Iterative_and_incremental_development)

Sequential vs. overlapping development



Source: "The New New Product Development Game" by Takeuchi and Nonaka. *Harvard Business Review,* January 1986.

Scrum Framework

Roles

- Product owner
- ScrumMaster
- Team

Ceremonies

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

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

Scrum Framework

Roles

- Product owner
- ScrumMaster
- Team

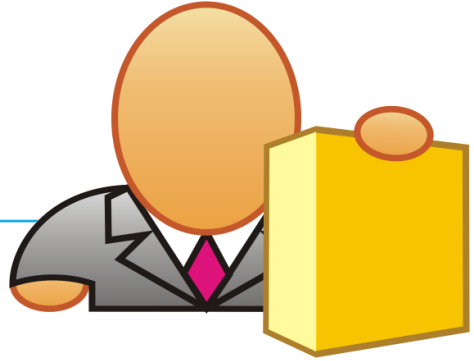
Ceremonies

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

Artifacts

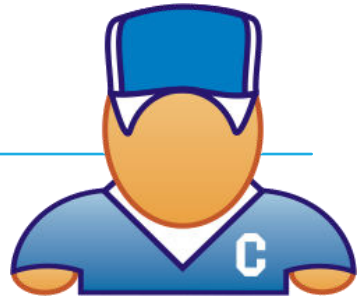
- Product backlog
- Sprint backlog
- Burndown charts

Product owner



- Represents the stakeholders and is the voice of the customer and is accountable for ensuring that the team delivers value to the business.
- Writes (or has the team write) customer-centric items (typically user stories), ranks and prioritizes them, and adds them to the product backlog.
- Defines the features of the product
- Decides on release date and content
- Responsible for the profitability of the product (ROI)
- Prioritizes features according to market value
- Adjusts features and priority every iteration, as needed
- Accepts or rejects work results

The ScrumMaster



- Is a Facilitator
- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enables close cooperation across all roles and functions
- Shields the team from external interferences

The Team

- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- The development team is responsible for delivering potentially shippable increments (PSIs) of product at the end of each sprint (the sprint goal).
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility
- Membership should change only between sprints

SRS: An agile aspect - User Story

- User stories are short, simple description of a **feature** told from the perspective of the person who desires the new capability, usually a user or customer of the system.
- User stories are short descriptions of **functionality–told from the perspective of a user**–that are valuable to either a user of the software or the customer of the software.

(http://www.gatherspace.com/static/use_case_example.html;
http://www.mountaingoatsoftware.com/agile/user-stories)

User Story Template

- User Stories (e.g. feature, epic, scenario) follow a simple template:
  - "As a **-role-**, I want -**goal/desire**- so that **-benefit-**"

User Story Template

- **As a** <type of user>, **I want**/need/can <some goal> **so that** <some reason>.

(http://www.mountaingoatsoftware.com/topics/user-stories)

User Story Examples:

- **As a** student, **I want** to download BRM subject lecture notes from the UTSOnline System **so that** I can prepare for the final exam.
- **As a** lecturer, **I want** to upload BRM subject lecture notes on the UTSOnline System **so that** I can deliver lecture.

User Story Examples

User Story Examples:

- **As a** passenger, **I want** to buy a ticket via Online Ticketing System **so that** I can travel from one city to another city in Australia.

- **As a** user closing the application, **I want** to be prompted to save anything that has changed since the last save **so that** I can preserve useful work and discard erroneous work.

User stories: Purpose

- User Story is not a whole requirement,
  - It serves as placeholders for conversations about the users' detailed needs. It is a starting point of **conversation and confirmation** between the analyst and stakeholders (e.g. product owner or process owner).
  - Based on the User Story, you have further conversations with the user or proxy user and identify the additional details. The Product Owner has further conversations with developers.
  - planned for project releases and iterations
  - written on a **story card, index cards or sticky notes**
  - stored in a shoe box, and arranged on walls or tables to **facilitate planning (releases and iterations)** and discussion
  - they strongly shift the focus from writing about features to discussing them. These **discussions are more important** than whatever text is written.

(http://www.gatherspace.com/static/use_case_example.html)

User Stories: Estimation and Prioritisation

- Each User story:
  - **estimated** in terms of time taken to complete a user story
  - **prioritised** (importance) for each release

User Stories: Estimation

- Common methods to **estimate the time to complete each user story** include:
  - T-shirt sizes (S, M, L, and too big)
  - Powers of 2 (1, 2, 4, 8)
  - The Fibonacci sequence (1, 2, 3, 5, 8, 13, etc.)
  - **Story Point (1 to 10) -** (method used in this subject)

User Stories: Importance/Prioritisation

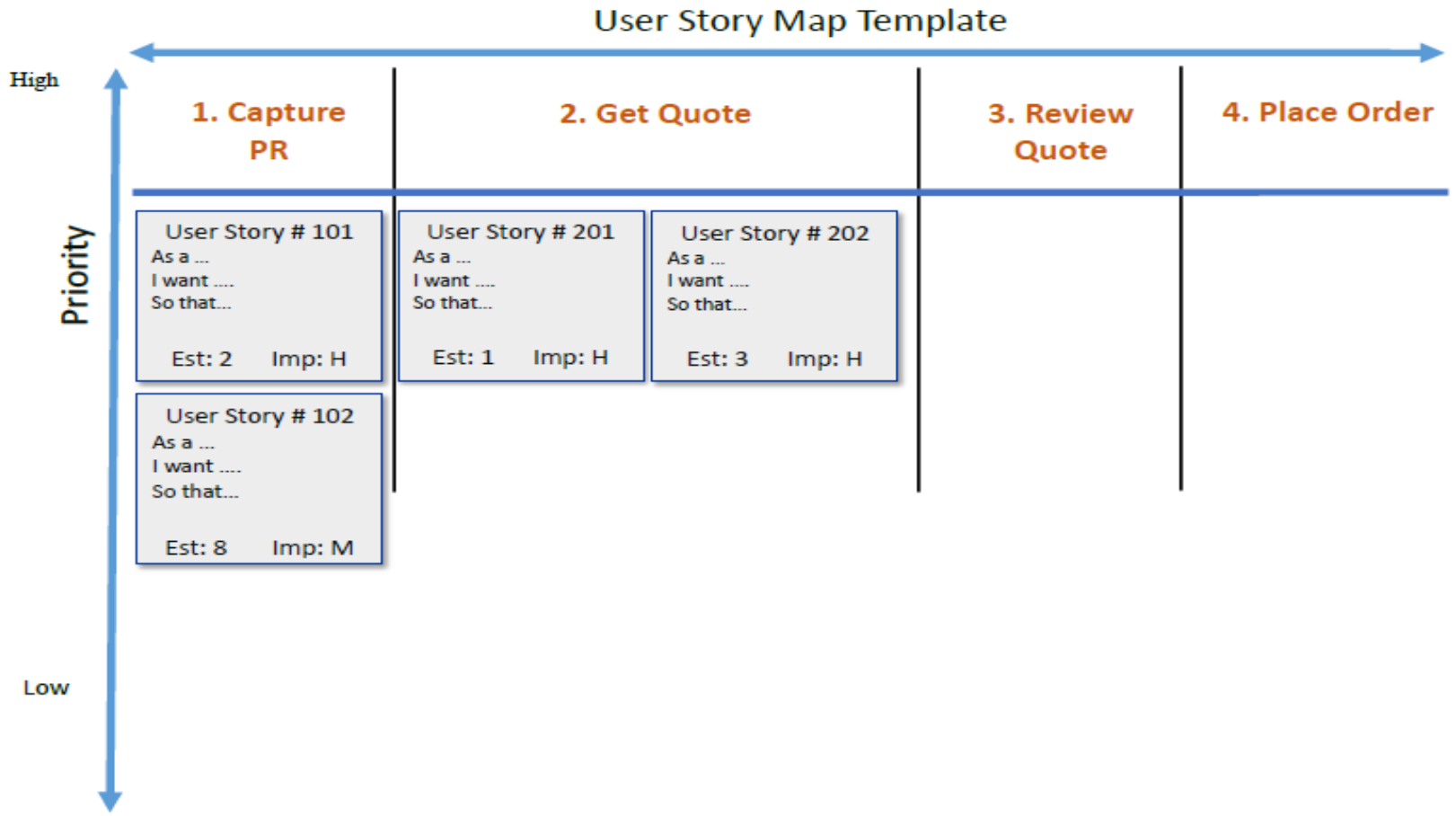
- **Requirement prioritisation** is used in software product management for determining which candidate requirements of a software product should be included in a certain release.
- Requirements (user stories) are also prioritized to minimise risk during development so that the most important or high risk requirements (user stories) are implemented first.
- Remember, the Product Owner prioritises the user stories and decides on which user stories will be included in a certain release in consultation with the development team. Please watch “Backlog Refinement Meeting” video.

http://en.wikipedia.org/wiki/Requirement_prioritization

User Stories: Importance and Prioritisation

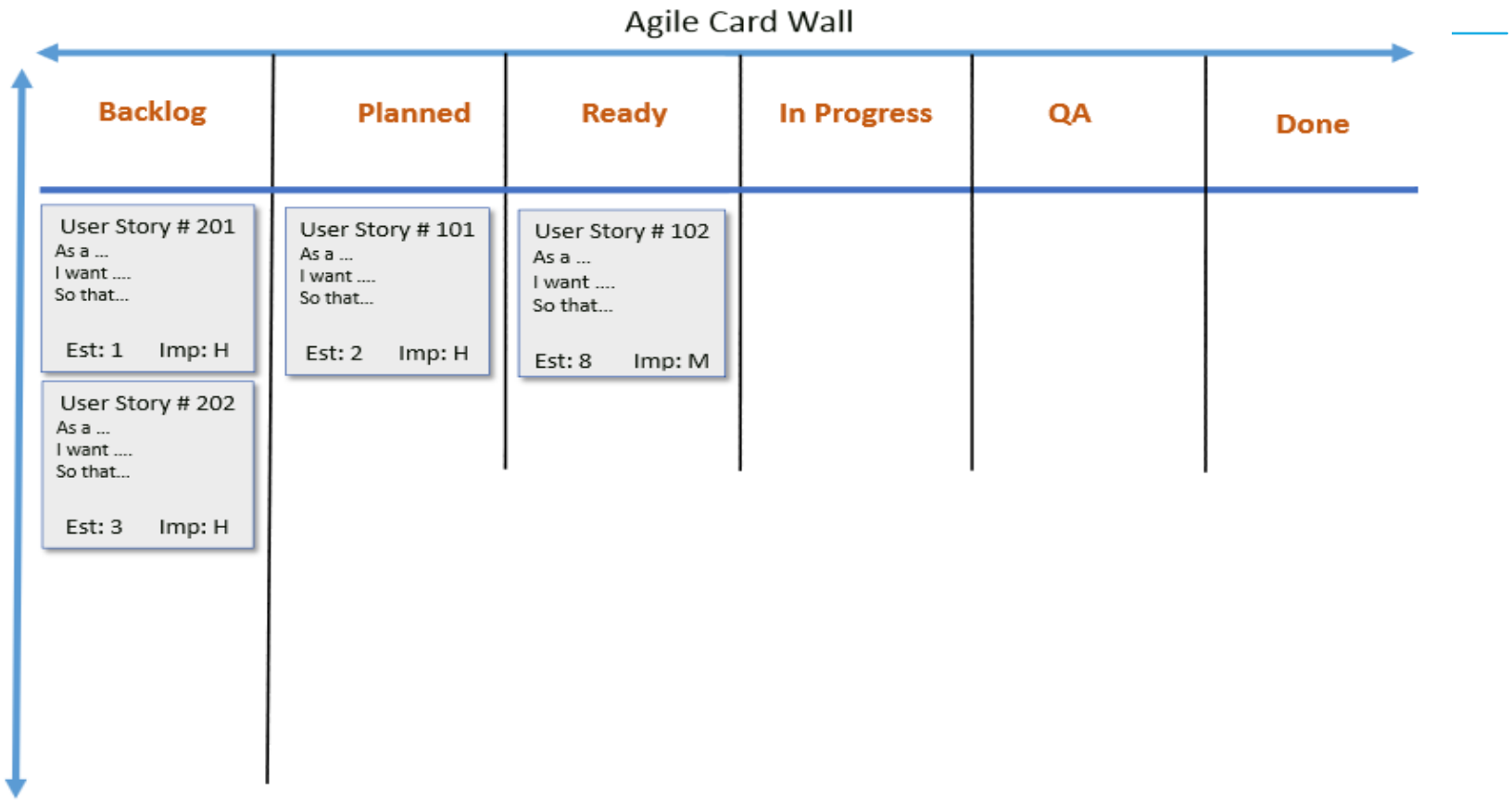
- Common methods to **prioritise (importance)** each user story include:
  - MoSCoW method (Must, Should, Could, Would/Won't)
    http://en.wikipedia.org/wiki/MoSCoW_method
  - Weighted Shortest Job First (WSJF)
    http://agile102.blogspot.com.au/2013/01/weighted-shortest-job-first-bit-of-safe.html
    http://www.scaledagileframework.com/wsjf/
  - **High, Medium, Low (HML) -** (method used in this subject)

User Story Map: Tutorial Case Study



- User Story Map is a way to organise user stories.
- Structure user stories into user or business workflow left to right, then decompose and prioritise them from top to bottom.

Agile Card Wall



- Agile Card Wall shows the progress of the team as well as what the team is currently working on.
- It is clearly visible to anybody who visits the team area, which includes stakeholders, project managers, team or anybody from the organisation.

Additional useful terms

- An **Epic or a feature** is a large user story. Because an epic is generally too large for an agile team to complete in one iteration, it is split into multiple smaller user stories before it is worked on.
- Epics will later be decomposed into smaller stories that fit more readily into a single iteration.
- **Theme** is a collection of related user stories.

Tools

- MS Office
- Rally
- IBM Jazz Hub
- JIRA
- Mingle

Agile Requirements Specification: Template adapted and used in this subject

Summary

- SRS has may roles and purposes
- SRS has different audiences
- Good SRS are hard to write: they have to conform to some quality attributes
- There are different structures one can use for writing SRS
- Traditional IEEE SRS are now replaced with User Stories on Agile projects.
- Each user story must be estimated and prioritised

Conclusion

- This Week's Workshop
  - **Quiz 4 – Data Modelling (3 marks)**
  - Tasks – Software Requirement Specification

- Next Lecture (week starting 29 April – after STUVAC)
  - Object Oriented Models with UML- Use Case Modelling
- Next Workshop
  - **Quiz 5 – Software Requirements Specification (3 marks)**
  - Tasks – Use Case Modelling

Reminder: Assignment 1 is due on **29/04/2019**