IFB295 – IT Project Management

Project Management

Lecture 1: Project Management, User Stories

Lecturer
Prakash Bhandari

a university for the real world®

Agenda

Unit Introduction

Aim, themes, delivery, assessments

Project Management

Terminology, approaches, paradigms

User Stories

The Scrum Framework, User stories

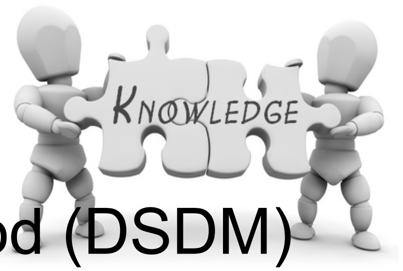
Project Management (PM)

“A **project** is a temporary endeavor, designed to produce a unique product, service or result with a defined beginning and end (usually time-constrained, and often constrained by funding or deliverables), undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value.”

“**Project management** is the process and activity of planning, organizing, motivating, and controlling resources, procedures and protocols to achieve specific goals in scientific or daily problems.”

Source: Wikipedia

Project Management Approaches in IFB295



- Incremental & Iterative Models (Agile)
  - Scrum (simple) Framework
  - Dynamic Systems Development Method (DSDM)
- Phased Models (Waterfall or Traditional)
  - PRINCE2 method

Agenda

Unit Introduction

Aim, themes, delivery, assessments

Project Management

Terminology, approaches, paradigms

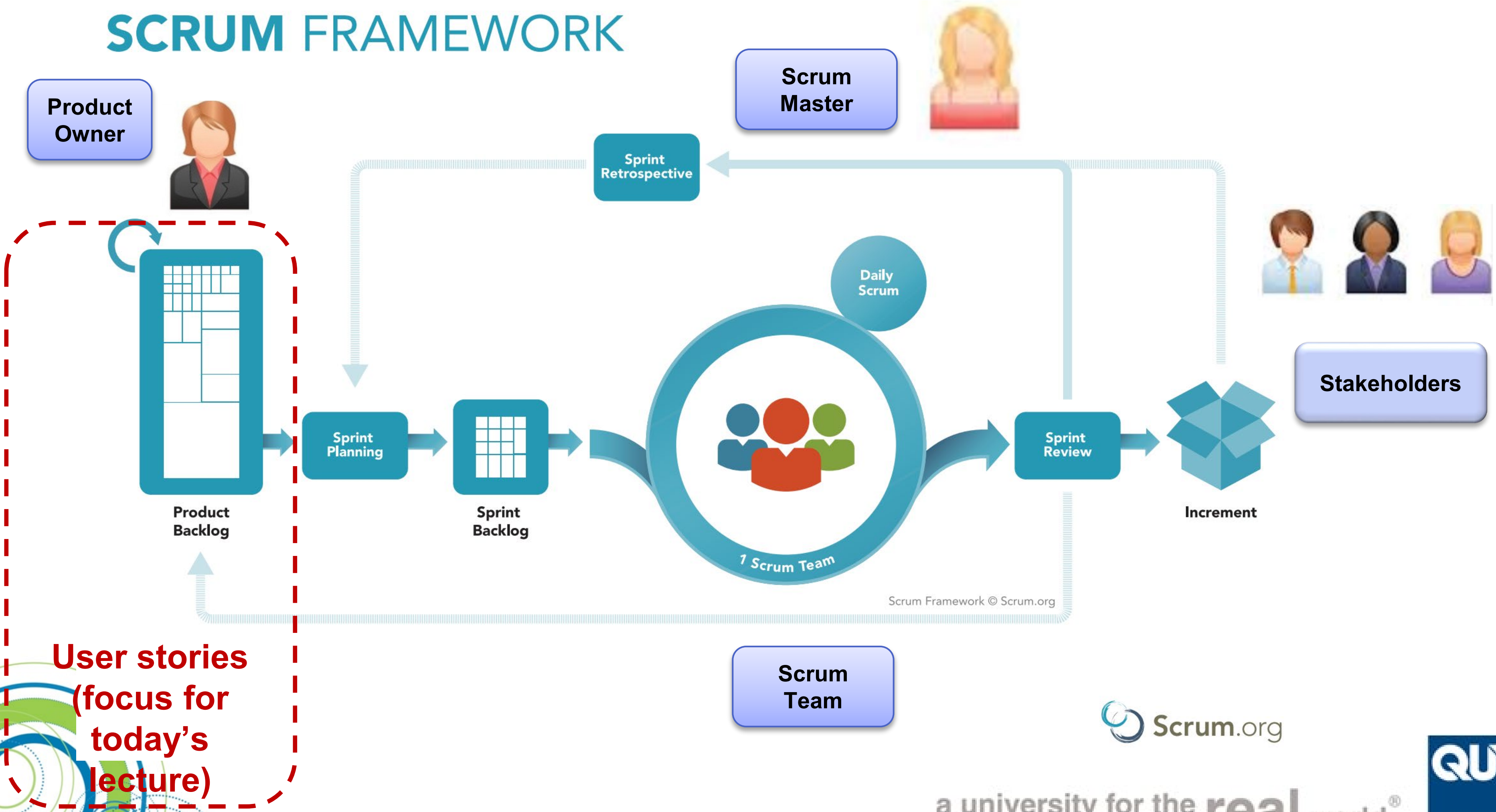
User Stories

The Scrum Framework, User stories

The Scrum Framework

- A commonly used Agile framework is SCRUM.
- Will be covered in more details in next week lecture.
- In this lecture, Scrum concepts namely Scrum Roles and User Stories are introduced so that you can work on the Week 2 Tutorial exercises.

Overview of Scrum Framework



User stories (focus for today's lecture)

Scrum Roles

- Scrum Team members plays one of the following roles

  - Product Owner/Clients

    Responsible for maximizing the value of the product being delivered by Development Team.

  - Developer

    Committed to deliver the product.

  - Scrum Master

    Will ensure that the team follows Scrum principles and guidelines

User Stories

- Short description of functionality.
  - textually small
  - short development time (1-3 days)
- From the user's (Clients) perspective.
- Provides value to the user or sponsor
  - consider both types of clients
- Must be testable.
- Provides enough information to make rough estimates.

Why User Stories?

- History of poor requirements capture
  - large out-dated documents
- Communication
  - track requirements
    - cards & BVC
  - up-to-date
    - conversation
  - understand user needs
    - conversation & confirmation

Story Structure

- As a [*role*], I want to [*do / see / change something*] so that [*outcome*].

e.g.

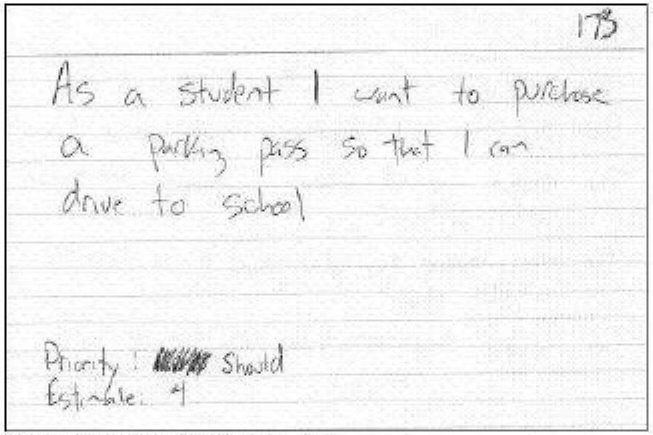
- As a *permanent employee* I want to be able to *see my leave balance* so that *I can plan my holidays*.

Story tools

- Commonly written on index cards
  - then stuck on walls
- Can be managed electronically
  - Trello
  - Microsoft Excel
  - Power Point slides
  - Jira - Atlassian
  - Microsoft Project etc.

Story cards

Front of Card

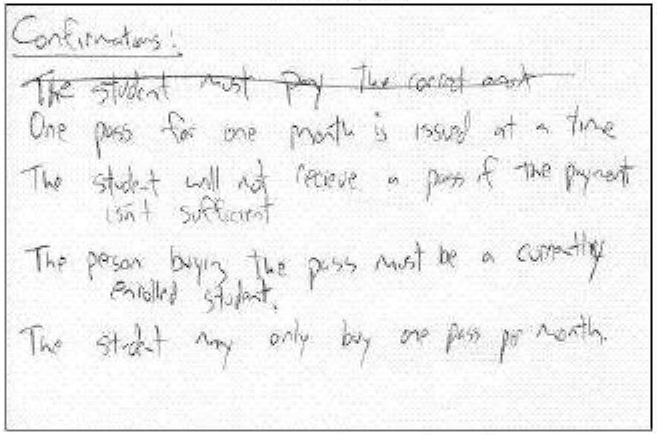


173

As a student I want to purchase a parking pass so that I can drive to school

Priority: ~~[illegible]~~ Should
Estimate: 4

Back of Card



Confirmations:

~~The student must pay the correct amount~~
One pass for one month is issued at a time
The student will not receive a pass if the payment isn't sufficient
The person buying the pass must be a currently enrolled student.
The student may only buy one pass per month.

Story Wall

Story Example

Story Ref #	Feature	Story Title	As a	I want to	so that
1	Upload Media	Upload Audio File	Uploader	upload an audio file	it is safely stored in a universally accessible location

Questions	Comments	Business Priority	Story Point Estimate	Status	Release	Iteration
Who can access it once it is uploaded?	*Safely* assumes cloud storage mechanism provides redundancy	High	2	Story written	1	1

Story

- *Card* – initial written description
  - often on index cards
- *Conversation* – between developers and customer representatives
  - customer driven design
- *Confirmation* – tests to determine when implementation complete
  - initial criteria written on story card
  - full tests in automated test suite

Conversations

- Details are discovered by talking with the customer representatives.
- Conversations occur whenever someone needs information.
  - not “once off”
- Story cards are a starting point for a conversation.
  - they don’t record the requirements
  - may record notes on card during conversation
- The User Acceptance Testing (UATs) become the requirements.
  - we’re done when we pass the tests

INVEST in Your Stories

- Independent
  - dependencies make planning, prioritisation and estimation difficult
- Negotiable
  - details are worked out in conversation
    - between developers and customers
  - too much detail limits the conversation and options
    - too easy to think all detail is in story
- Valuable
  - must provide value to customer
    - get customer to write stories

INVEST in Your Stories

- Estimable
  - at least to start with ballpark estimates
    - prioritisation and planning depends on this
  - problems: lack of domain knowledge or story too big
- Small
  - representing a few days in person effort
    - the smaller the stories, the more accurate the estimates
- Testable
  - need completion criteria (Acceptance criteria)
    - we don’t develop what we can’t test

Acceptance Criteria

- Use Given-When-Then template to write acceptance criteria for a User Story:

  (Given) some context
  (When) some action is carried out
  (Then) a particular set of observable consequences should obtain

- An example:

  - Given my bank account is in credit, and I made no withdrawals recently,
  - When I attempt to withdraw an amount less than my card's limit,
  - Then the withdrawal should complete without errors or warnings

Example Stories

- **As a *lecturer* I want to be able to *see a list of all students enrolled in my classes* so that *I can see class lists and numbers enrolled*.**

Acceptance Criteria

Given I have a "Show Classes" link displayed for the classes I lecture,

When I click on "Show Classes",

Then the system should display the list of classes together with class activity, class no, day, time, and room where classes are held for each class.

Example Stories

As a *coursework student* I want to be able to *see a list of all available offerings of my classes from which I can select classes to attend* so that *I can choose convenient times to be on campus*.

Acceptance Criteria

Given I have a “Available offerings” link displayed for each of my classes is displayed ,

When I click on “Available offerings”,

Then the system should display a list of offerings for that class together with “Register” button.

Which are Correct Examples of Stories?

- As a *marketer* I want the *Digital Workspace to look like a QUT site* so that *we project a consistent image*.
- As an *IT support* I want to be able to *post an outage / downtime notification* so that *I can inform people that the portal is out of service*.
- As a *web developer* I want to be able to *capture usage data in XML log files* so that *we can analyse patterns of usage*.

IFB295 Tutorial Tips

- Roles of the end-Users – Identify types of users for the given case study.
  - be specific
  - Clients of your team plays these roles
- Goals – For each role
  - general expectations of system functionality
- Depth First
  - focus on one aspect of the system at a time
    - like stories breed like rabbits
- Clarify workflows
  - use Power Point slides, Excel spreadsheet or post-it-notes

Story Brainstorming

- Choose one aspect of system
- Everyone starts writing stories on cards
  - ~10 minutes
  - stop when people start slowing down
  - avoids facilitator filtering
- Review stories
  - 3 stacks
    - keep
    - fix
    - throw away

Reviewing Story

- After each brainstorming session
  - each author reads out their stories
- Keep Stack
  - clear, in scope, meets INVEST principles
- Fix Stack
  - do not meet INVEST principles
  - in scope but not clear, too large, too small, …
- Throw Away Stack
  - duplicates, out of scope
  - do not provide value to user or customer
    - e.g. focuses on technical issues

User Role Modelling

- What types of people will use the system?
  - each will have different goals
- Don’t think of an anonymous user
  - oversimplification
- Identify different user roles
  - brainstorm initial set
  - group related roles
  - consolidate roles
  - refine roles
- Don’t get stuck on organisational roles

Examples Roles – QUT Digital Workspaces

Undergrad Students

Postgrad Students

Research Students

Full-Time Students

Part-Time Students

External Students

Academic Staff

Casual Academic Staff

Course Coordinators

Teaching Support Staff

Admin Staff

IT Support

Web Dev Team

Chancellery Staff

Examples Roles - Refined

Students	Academics	Admin Staff	IT Services	Chancellery
Coursework	Casual			
Research	Course Coordinator			
External	Teaching Support			

Story Writing Guidelines

- Start with goals
  - for each role identify the goals they have for using the system
- Write closed stories
  - allow the user to accomplish something useful
- Ignore the UI
- Write for one specific user
- Use active voice
- Focus on the next few iterations
  - small estimable stories
  - larger more general stories for more distant future

Epics

- Too big to implement in a reasonable timeframe
- Too big to estimate
- Compound story
  - split into separate stories
- Complex story
  - hard to split
  - providing end-to-end functionality
    - “slicing the cake” – Bill Wake
- Too general
  - useful as a starting point for ideas
    - come back to it later

Goal ≠ Story

- What the system needs to accomplish or support
- Larger and more complicated than a story
  - usually not estimable, small or testable
- Break down into smaller pieces
- Good prompt for story discovery

Activity ≠ Story

- What users can do with the application
- Not valuable in itself
- Usually part of another story
  - may need to create new story to support activity

Task ≠ Story

- Things the development team need to do
- No business value by themselves
- Usually merge with another story
  - may need to create new story to include task

Non-Functional Requirements

- Constraints on system behaviour
  - criteria to judge system effectiveness
- "ilities"
  - stability
  - reliability
  - usability
  - portability
  - scalability
  - maintainability
  - efficiency
  - …
- Need to be understood & captured
  - agile principles: communication and flexibility

Non-Functional Requirements

- Constraints – written like stories?
  - **As a *call centre operator* I want the system to *retrieve data in less than 1 second* so that *I can respond to customer queries with no delays*.**
- Infrastructure stories
  - **As a *web admin* I want *a web server set up by Dec. 1* so that *we can launch external beta testing*.**
- Stories don't work for everything
  - e.g. document an API as a contract

Preparation for week 2 - Scrum

- Ensure you are enrolled in a tutorial

- Read The Scrum Guide

http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100

Unit Plan – Next Week

No tutorial this week

Single-Source of Truth
QUT Blackboard