DRAFT - ICT Project Guidance

Delivery:  
User Story Development & Management

Version:

0.1

Author:

Sky Sigal, Solution Architect

## Purpose

The purpose of this document is to provide core guidance on the developing and managing Work Items on Boards managed within either a Scrum (iterative) or Kanban (continuous) context, to deliver of value to business sponsors and stakeholders.

## Synopsis

User Stories are distinct from contractual Requirements developed before procurement of services or products. User Stories are work item management artefacts developed during the delivery and maintenance phase, for defining and prioritising work items.

## Contents

[Purpose 1](#_Toc156390714)

[Synopsis 1](#_Toc156390715)

[Contents 2](#_Toc156390716)

[Context 3](#_Toc156390717)

[Methodologies 3](#_Toc156390718)

[Kanban 3](#_Toc156390719)

[Scrum 4](#_Toc156390720)

[Work Items 4](#_Toc156390721)

[User Stories 4](#_Toc156390722)

[User Story Acceptance Criteria 5](#_Toc156390723)

[Ready 6](#_Toc156390724)

[Comparison to User Cases 7](#_Toc156390725)

[Appendices 8](#_Toc156390726)

[Appendix A - Document Information 8](#_Toc156390727)

[Versions 8](#_Toc156390728)

[Images 8](#_Toc156390729)

[Tables 8](#_Toc156390730)

[References 8](#_Toc156390731)

[Review Distribution 8](#_Toc156390732)

[Audience 8](#_Toc156390733)

[Structure 8](#_Toc156390734)

[Diagrams 9](#_Toc156390735)

[Terms 9](#_Toc156390736)

## Context

## Methodologies

Agile recognises the volatility of product development, relying on self-organising teams to respond to evolving demand.

Kanban and Scrum are two widespread methodologies of managing work items in accordance with the above principles and ideals.

At its essence, Scrum requires high control over what is in scope, kanban lets you get into a continuous flow.

### Kanban

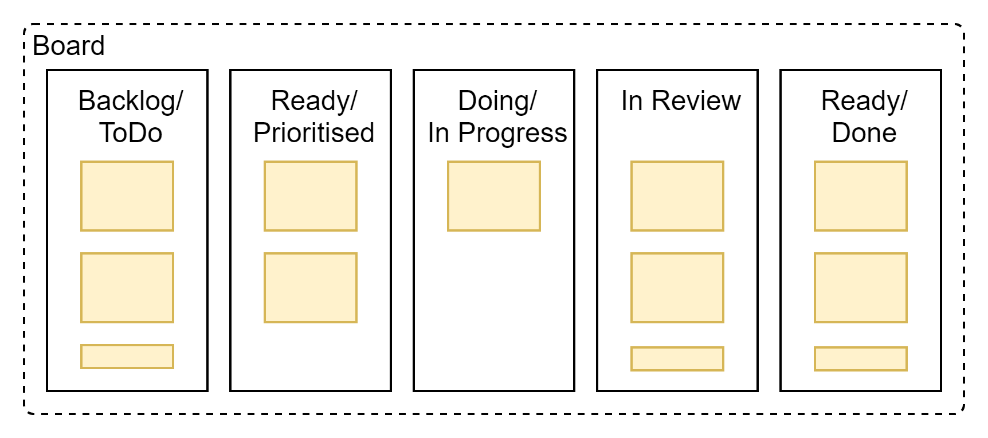


Figure : Kanban Board Example

Kanban (“bill board” in Japanese) is a visual system used to prioritise and manage the allocation and completion of work items in a continuous manner.

It emphasises maximising efficiency by measuring and highlighting lead time, cycle (completion) time and optionally caping Work In Progress (WIP) to remain sensitive to inefficiencies.

### Scrum

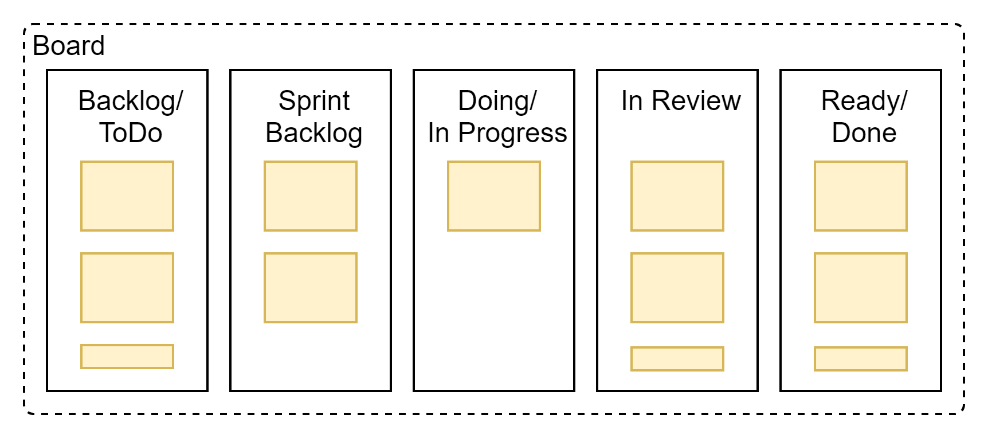


Figure : Scrum Board example

Scrum is a system – not necessarily visual -- of committed to complete increments of deliverable work in short fixed length sprints (e.g., 2 weeks), with an emphasis on maximising efficiency by quickly establishing learning loops via the use of defined:

* Roles (Product Owner as consumer advocate, Scrum Master as coach, Development Team)
* Ceremonies (Sprint Planning, scrums/standups, sprint reviews, sprint retrospectives)
* Metrics (team velocity, sprint burndown, workload distribution)

Note:  
Scrum’s may make tasks as visual as kanban, using a Scrum Board. More or less the only difference between a Kanban board and a scrum board is how Scrum uses the term Sprint backlog, and Kanban either dispenses with it, or makes a column called Prioritised/Ready [To Do].

## Work Items

Work Items managed by either approach are generally developed following Agile practices of creating User Stories, categorised or sub-set to Epics and/or Features.

Epics are *areas* of work, that work will continue within for a long while, certainly longer than a single sprint.

Features are system capabilities, generally accessible to authorised Users (Diagnostics, Error management, Session Management, Commenting, etc.).

### User Stories

User Stories are work items.

Depending on the software system used to manage them, the default fields of a User Story vary.

On Confluence, for example – the default fields are as follows:

* **Title**, used to recognise and identify the User Story,
* **Description**, used to write the user story using the project agreed format (e.g., “As a <role sme> I want/desire <outcome>…So that…<provide value>”[[1]](#footnote-2)).
* **Acceptance Criteria**, used to describe the fit criteria.   
  In mature environments[[2]](#footnote-3), functional behaviour is defined using a Gherkin DSL[[3]](#footnote-4) format (“Given…When…Then…”) to facilitate the test being automated using standard industry testing tools.
* **Details**, a group of metadata (e.g., *Assignee, Epic Link, Priority, Due Date*) used to categorise and manage the work item.

#### User Story Acceptance Criteria

While User Story descriptions are of course important, Acceptance Criteria[[4]](#footnote-5) are even more so: User Stories are incomplete without them[[5]](#footnote-6).

##### Testing Behaviour

User initiated activities should be demonstrated using Behaviour Acceptance Tests, defined using a Gherkin syntax.

For example, for the following user story:

As an <Authorised Teacher>  
I desire <being able to see a queryable list of my Learners>  
When I select the View on my Dashboard.

The following is a acceptable simple gherkin based acceptance test:

Given a view of my classroom  
When I select Learners  
Then I should see a single page of Learners

##### Testing Outcomes

Gherkin statements are slightly more difficult to develop when the user story does not describe an event, but describe an outcome.

For example, the following User Story:

“As the product owner   
I desire the system   
to authenticate users using the corporate IdP”.

may require acceptance tests to define examples of user behaviour relying on the condition being in place:

“Given <username> & <password>   
When I try to identify myself,   
Then I am, using the corporate IdP”.

##### Quality Requirement Alignment

It is possible -- but not practical -- that the User Story Acceptance Criteria define tests to indicate that the area worked contributes to meeting quality requirements.

Instead, Security, Performance, Maintainability, Usability, and Accessibility are best measured by independent ongoing testing, preferably automated in compilation/packaging/deployment pipelines, associated to Feature cards for each of those Qualities.

##### Functional Requirement Alignment

It is also possible – but again not practical – for User Story Acceptance Criteria to include tests to evidence how the work contributes to meeting Functional Requirements.

Instead, it is best to categorise what Functional requirement the User Story contribute towards, for better traceability and reporting.

#### Ready

To be considered “Ready” for pick up and being worked on, the totality of the a User Story (its description and acceptance criteria) are used to assess it against INVEST qualities:

* Independent: The user story should be independent of all others. Because they are not connected, they can be worked on in any order.
* Negotiable: A user story should be flexible enough to allow for negotiation between the customer and product owner.
* Valuable: What value does the user story bring? If you cannot find any value, the story should not be completed.
* Estimable: You should be able to estimate how long a user story will take so that you can effectively manage your time.
* Small: The story must be small enough to be completed within a single sprint.
* Testable: You must be able to test your user story in line with quality assurance standards.

#### Comparison to User Cases

User Stories are less formal than Use Cases used in other delivery methodology.

The lack of formality can lead to incompleteness and/or vagueness.

Appendices

Appendix A - Document Information

### Versions

* 1. Initial Draft

### Images

[Figure 1: Kanban Board Example 3](#_Toc156385231)

[Figure 2: Scrum Board example 4](#_Toc156385232)

### Tables

### References

**There are no sources in the current document.**

### Review Distribution

The document was distributed for review as below:

|  |  |
| --- | --- |
| Identity | Notes |
| Russell Campbell, Project Manager |  |
| Liam Hannigan, Scrum Master |  |
| Sandy Britain, Enterprise Architect |  |
| Rodney Snell, Team Lead |  |

### Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Structure

Where possible, the document structure is guided by either ISO-\* standards or best practice.

### Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Terms

Refer to the project’s Glossary.

##### Backlog

: a prioritized list of tasks to be done. It prioritizes features, fixes, requirements, and [user stories](https://www.wrike.com/agile-guide/user-stories-guide/), with the most important items listed at the top.

##### Backlog Grooming

: a recurring scrum activity when a Product Owner (PO) and some or all the rest of the team review tasks to prioritise them for future sprints.   
It is typically done between sprints, when a set of User Stories are proposed and accepted for being moved on to the Sprint Backlog ready for the next sprint.  
Common tasks include eliminating tasks that are no longer relevant, breaking down large tasks, talking through tasks to smooth out ambiguities, estimate and or assign story points, reprioritising based on new information.

##### Done

: the state of a Work Item that was selected from the product backlog (PB) to be part of a sprint backlog (SPB) an then was progressed through Doing, Tested to this state of Done.

##### Epic

: an ongoing area of work.

##### Feature

: a capability group.

##### IT

: acronym for Information, using Technology to automate and facilitate its management.

##### ICT

: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. IT is a subset of ICT.

##### INVEST

: good User Stores are Independent, Valuable, Estimable, Small, Testable (INVEST).

##### Kanban

: A methodology to adhere to the Agile philosophical point of view on delivery.

##### Kanban Board

: an agile workboard that lets team members visualize progress and holdups.

##### PB

: acronym for *Product Backlog*.

##### PO

: acronym for *Product Owner*.

##### Product Backlog

(PB): the *prioritized* list of tasks to deliver.

##### Product Owner (PO)

: is responsible for the Product Backlog, and therefore User Stories within it.

##### Ready

: a definition of a User Story ready to be worked on. This means it has Acceptance Criteria and an estimate of how long it will take to be accomplished.

##### Scrum Board

: Project visible Board of Work Items. It has the same objectives as a Kanban Board, with slightly different column titles.

##### Sprint Backlog

: a selected subset of the PB that is to be delivered in the next sprint.

##### Sprint Planning

: a Scrum ceremony to develop User Stories, prioritise them, and select and accept a subset of ready Work Items from the Product Backlog (PB) to be added to the Sprint Backlog.

##### Sprint Retrospective

: …

* Roles (Product Owner as consumer advocate, Scrum Master as coach, Development Team)
* Ceremonies (Sprint Planning, scrums/standups, sprint reviews, sprint retrospectives)
* Metrics (team velocity, sprint burndown, workload distribution)

##### Standup

: a daily *Scrum* activity for team members to state what they accomplished last time, what they are working on, what tasks (not person) they are blocked by.

##### Product Roadmap

: along with Requirements, comes first, before the Product Backlog. It lays down the strategy, vision and direction, and Features delivered in large chunks of Now, Next and Later, or committing at best to Quarters (although they slip).

##### User Story

: an Agile Work Item. When a User Story has been shaped up to be described in a well-known manner (“As a …”, etc.), with associated Acceptance Criteria, and Metadata (sizing, etc.) it is in a state of “Ready” to be worked on.

1. Rarely, the value is put first (“To help [customers], as a [support specialist], I desire [xyz…]”) [↑](#footnote-ref-2)
2. Best practice is to follow Test Driven Design (TDD) patterns and develop automated tests before working on the service itself. [↑](#footnote-ref-3)
3. [Gherkin Domain Specific Language (DSL) for BDD Testing - HMTMCSE](https://www.hmtmcse.com/sqa/java-selenium-cucumber/cucumber/gherkin) [↑](#footnote-ref-4)
4. *Acceptance Criteria* (AC), are groups of individual *Acceptance Criterions.* [↑](#footnote-ref-5)
5. They are also the hardest part to get right. [↑](#footnote-ref-6)