IFB295 Study Guide | 2022 Semester 2

Mark Walpole | Notes for IAB330 at the Queensland University of Technology

---

Table of Contents

IFB295: IT Project Management

In your information technology career, you will participate in and then lead project teams that are expected to deliver real benefits to stakeholders. This unit builds on your previous studies of earlier units to define a high-level solution by using a range of approaches of project management methodologies and frameworks. You will enhance your learning of these approaches (Agile, PRINCE2 etc.) by practicing it collaboratively with other students. To be successful in a complex environment, you need to employ appropriate project management strategies, tools and techniques for a given context. This unit provides you with the necessary knowledge and skills to enable you to effectively manage your project in the IFB398 Capstone Project (Phase 1) and IFB399 Capstone Project (Phase 2) and to be prepared for a project environment in industry.

Week 1: Project Management, User Stories, Scrum, and Teamwork

Project Management

Project management is the act of planning, organising and creating procedures and protocols to ensure that a project achieves it's goals, is delivered on time and is of a high quality.

Terminology

- Methodology
    - A methodology is a set of methods, principles and rules that follow concepts such as paradigm, theoretical model, phases and techniques.
- Framework
    - A framework is a basic structure underlying a system, concept, or text
- Artifact
    - A by-product produced during the development. This could be a use case diagram, class diagram or other design documents that help describe the design and architecture

Scrum

Scrum is a framework designed to help teams work together. Scrum encourages development teams to learn through different experiences, self-organise themselves while working on a problem, and reflect on their progress, wins, and losses in an attempt to continuously improve themselves.

Scrum Roles and Responsibilities

- Product Owner
    - This is the owner of the product. They are responsible for optimising value, ordering the product backlog in priority and product incrementing
- Scrum Team
    - The scrum team is the team working on the project. They are responsible for self-origanisation, cross-functional, ordering and choosing which items appear in the sprint

backlog
- Scrum Master
    - The scrum master is the leader of the scrum team. They are responsible for coaching, training, and supporting the scrum team.

User Stories

User stories are short informal descriptions explaining a software feature from the perspective of an end user. It's purpose is to demonstrate how the software or a feature will provide value to the customer. User stories also help define the acceptance criteria of a particular feature.

For example:

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

INVEST

All user stories should follow the INVEST acronym.

- Independent
    - User stories should be independent as dependencies can make planning, prioritisation and estimation difficult
- Negotiable
    - User story details should be worked out in a conversation between the customer and the developer
- Valuable
    - The user story must provide some sort of value to the customer. User stories should be written with the customer to ensure this
- Estimable
    - The feature or product the user story is crafted around should be estimable. These estimates are then used in the initial prioritisastion and planning of the project
- Small
    - User stories should be short and sweet. They must only represent a few days in person effort as this allows them to be more precise and accurate
- Testable
    - User stories should be able to be crafted into acceptance criteria. We can't develop things we can't test

Acceptance Criteria

Acceptance criteria is the criteria that must be met before we can say a feature is completed. They should follow a Given-When-Then template such that:

- Given some context
- When some action is carried out
- Then a particular set of observable consequences

For example:

> 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 numbers, day and time, and room where the class is held

Epics

Epics are large user stories that are too big to implement in a reasonable timeframe. These types of user stories are split into smaller separate stories to help spread and breakdown the workload.

Week 2: Why Agile? Scrum framework, principles. and roles

Agile Development?

Agile development focuses on repeating the waterfall development cycle for each feature working in 'sprints' that focus more on iterative and incremental development. Iterative and incremental development is the process of starting and finishing small chunks of the given problem at any one time. One of the major benefits of this is that the customer is constantly involved in the decision making and can slowly watch the solution be developed.

Developing in this iterative and incremental manner allows us to easily break down the problem and helps us avoid making sub-optimal design choices early on based on limited knowledge and understanding. Another benefit to this design process is that we have sections of working code at the end of each iteration allowing us to show managers and customers. This allows the development team know they are doing the right thing and gain active feedback.

Scrum

The scrum framework has 3 main principles:

1. Optimise value: Incrementally deliver value each sprint (every 30 days or less)
2. Optimise productivity: Foster self-organising teams
3. Optimnise predictability: Use empirical process control. This means that things are done in a highly visible manner e.g. Burndown charts

Product Backlog

A product backlog is a backlog of items that consist of the user stories, an estimated effort, and a priority order. For example:

Id	Priority	Backlog Item	Effort
412	1	Allow a guest to make a reservation	4
414	2	Allow a guest to cancel a reservation	2
415	3	Allow a guest to change the dates of a reservation	2
417	4	Allow an employee to run RevPAR reports	8
...			

Product Backlog Item (PBI)

A product backlog item is an item or task that goes into the product backlog. These contain important information about the PBI such as a description, reproduction steps (in the case of a bug), the acceptance criteria, and any discussions currently occuring about it such as updates or changes.

Sprint Ceremonies

A sprint is a defined period of time where a set number of PBI's are estimated, assigned and then expected to be completed. A sprint can be done in 1-4 week periods but once a time period is chosen it's unlikely to change. Each sprint consists of 4 main events:

1. Sprint planning (1-4 hours)

2. Daily scrum meetings (15 minutes)
3. Sprint review (1-4 hours)
4. Sprint retrospective (1-3 hours)

Sprint Planning

Sprint planning occurs at the start of each sprint. It's in this meeting that the total effort for the sprint is calculated (the total amount of working hours the team can provide) and which PBI's are assigned to the sprint. The total PBI's effort cannot exceed the maximum sprint effort. The PBI's brought into the sprint are now reviewed to make sure all the information makes sense to the team and that all required information is there to complete it.

Daily Scrum

Daily scrums are generally the first thing to happen each day and last for around 15 minutes. In these meetings the development team and the scrum master get together and each member describes:

1. What they did yesterday
2. What they plan to do today
3. If anything is blocking or preventing them from meeting their sprint goal
4. Any clarifications needed for them to go about their day

Sprint Review

A sprint review happens on the last day of the current sprint. In this meeting there are a variety of things mentioned:

1. The completed PBI's are walked through and explained (demonstrate the new working features or things that got done)
2. Any PBI's that weren't completed are talked about (why didn't they get finished)
3. How many times code was deployed to production
4. If all the completed PBI's comply with the definition of done (a document describing what needs to occur for a PBI to be marked as complete)
5. The Burndown chart is looked at and that information is taken and used in how the next sprint is planned e.g. how many items the team can take on

Sprint Retro

The sprint retrospective is a meeting where the development team reflect on the previous sprint and talk about:

- What went well
- What didn't go well
- What can be improved
- What lessons were learnt

Product Backlog Item Prioritisation

The product owner will generally set an order in which the PBI's must be completed in. When setting priorities it's good to follow the acronym MoSCoW:

- Must have: The customer would expect all of these items to be implemented
- Should have: The customer would expect most of these items to be implemented
- Could have: If there is capacity within the project and resources available these will be implemented
- Won't have: It is unlikely that these items will be implemented

Story Estimation

Each PBI (or story point) is initially estimated. This is done for a variety of reasons such as:

- To get an initial feel for the project cost
- Determine if the project is still feasible and whether development should continue
- Reduce, discard and re-prioritise PBI's based on cost

Each PBI is estimated in effort (not units of time). This means, for example, in a team of five people, suppose each member spends 10 hours a week on the project. This would mean that, with each person spending 2 hours a day, a total effort of 10 hours a day, or 50 hours a week, would be the estimated sprint effort. It's important to be consistent when labelling a PBI's effort as all PBI's with that estimated effort should require the same amount of effort. This allows all PBI's with the same effort to be exchangeable with each other.

There's many ways of labelling an estimate for a PBI:

- Powers of 2 (2, 4, 8, 16)
- T-shirt sizes (S, M, L, XL)
- Fibonacci scale
- and more

Planning poker is a game to help estimate the effort for each PBI. In this game each player has a set of cards (1, 2, 4, 8, 16, 32), one team member reads out the user story and each team member must place a card upside down with their estimation of the required effort. Once all cards are place they are revealed and the most common estimate becomes the effort. Everyone must then explain why their estimate is like such and whether they agree with the others or not.

Week 3: Project cost, Velocity, Release planning, and Sprint planning meetings

Top-Down Guesstimate

By looking at the team size for each sprint we can get a better estimate of how much work can be completed in the sprint. By also looking at the project as a whole we can estimate how many sprints will be needed to completed giving us an estimate of time and human resources cost.

Bottom-Up Mapping

Bottom-up mapping is done by looking at a PBI that is one story point (or effort) and discussing everything that needs to be done to implement the feature. This process should be repeated for a few similar stories for each size grouping to bring out some consistency. With this information a conversion factor can be calculated: conversion factor = development time / story points.

Estimate Comparisons

Once the top-down and bottom-up estimates are done they are then compared to each other. If they are not similar then a discussion takes place mentioning:

- Any assumptions
- Why the differences
- Re-estimate of effort

Team Velocity

At the end of a sprint the team will count the number of story points completed. It can be assumed that, assuming the same conditions apply next sprint, that the team will complete roughly the same amount of story points in the next sprint. This velocity can be used to convert story points into calendar effort.

The initial velocity estimate should be 1/3 to 1/2 of the available time. For example, if there are 6 members working in 2 week sprints with 10 total working days in each sprint (60 days). Using the initial velocity at 1/3 results in 20 ideal working days per sprint. Subsequent sprints would then use the sprint velocity calculated from prior sprints for forward planning.

Release Planning

Release planning is when it is determined when the deliverable is given to the product owner i.e at the end of which sprints will there be a release. User stories are grouped based on functionality and the smallest set of stories that delivers immediate business value are identified, these will make up the initial release.

Release planning issues:

- Stakeholder key dates:
  - What happens if the release plan doesn't correspond to the stakeholders important dates
- Balancing business value vs technical risks
- External dependencies
  - Risks of delay
- Resource requirements
  - Availability

Sprint Planning

There are two main parts to sprint planning. During the first part a meeting occurs with the product owner where the product owner will:

- Select stories from the backlog
- Set sprint goals
- Priorities stories from highest to lowest
- Revisit release plan (are there new priorities?)

The velocity for the sprint is calculated and the product owner decides what to do about incomplete stories.

In the second part a meeting with the team occurs where they break down each story into a set of tasks estimating the effort for each, this becomes the sprint backlog.

Task Break Down

Here are the things that occur when breaking down a task:

- Read out the story

- Brainstorm the tasks required to implement the story
    - These should be kept short (half a day or less)
    - Each task is written out on an index card or some other medium
- All stories have acceptance criteria and a "verify story is complete" task
- The tasks are then reviewed making sure everything is doable and the list seems complete
- A comparison of tasks and stories is done to ensure nothing was forgotten or missed

During the Sprint

During the sprint the highest priority tasks should be completed first with each developer only taking on one task at a time with stories being completed throughout the sprint. Once a story is complete (following the definition of done) the developer would then pick the next most important task and work from there. If a new task or bug is discovered during the sprint then a card is created for it and reviewed in the sprint retrospective, it is not worked on in the current sprint.

Halfway through the sprint a review is done ensuring that all tasks can be finished. If not, the sprint plan is quickly redone discussing which stories are to be dropped and a confirmation is done with the product owner about the new story priorities.

Week 4: Agile Manifesto, DSDM Philosophy and Principles, Success Factors and Risks

Manifesto for Agile Software Development

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.
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.

Taken from https://agilemanifesto.org/principles.html

Dynamic Software Design Methodology (DSDM)

DSDM is an agile methodology which primarily focuses on on the full project lifecycle. The main philosophy behind DSDM is to bring business value from each project. This is achieved when all stakeholders:

- Understand and buy into the business vision and objectives
- Are empowered to make decisions within their area of expertise
- Collaborate to deliver a fit for purpose business solution
- Collaborate to deliver to agreed timescales
- Accept that change is inevitable in projects and understand that a solution will grow over time

We use DSDM because:

- It has a broader focus than most Agile approaches as it puts the focus on projects more than the development and delivery of it
- Has a long track record of success
- Proven to be scalable being able to be implemented in small and large complex organisations
- It's equally as effective for IT and non-IT related projects

For DSDM to work all principles need to be followed, if not then this because a project risk in itself and must be addressed. These principles include:

1. Focusing on the business needs:
   - Understand the business priorities
   - Establish a sound business case
2. Being able to deliver on time:
   - Timebox the work
   - Focus on the agreed business priorities
   - Always hit deadlines

    - Build confidence through predictable delivery
3. Collaboration between team members:
    - Involve the right stakeholders at the right time
    - Build a one-team culture
4. To never compromise quality:
    - Set a level of achievable quality
    - Ensure the quality does not become a variable
    - Build in quality by constant review
    - Design and document appropriately and clearly
5. Build incrementally from solid foundations:
    - Carry out appropriate analysis
    - Create a solid foundation first
    - Formally re-assess priorities with each delivered increment
6. Develop iteratively:
    - Build products using an iterative approach
    - Incorporate business feedback into each iteration
    - Accept most details will emerge in the later stages of development
    - Embrace change
7. Always communicate clearly:
    - Encourage informal communication between members
    - Run daily stand-ups
    - Keep documentation lean and timely
    - Always aim for honesty and transparency
8. Demonstrate control:
    - Make plans and progress viable to all
    - Manage proactively
    - Use appropriate level of formality for tracking and reporting
    - Measure progress through delivery products
    - Continuously evaluate the projects viability

Preparing for Success

- Always adhere and follow the principles making as little deviations as possible
- Understand the constraints the project is under:
    - Consider the variables
    - Think about the people
    - Consider the principles
- Deal with problems as they arise:

    - Changing the deadline is not an option as time is fixed
    - Adding resources is not an option as resources and cost is fixed
    - Quality is not negotiable, everything must be held to a standard
    - Avoid adding "Must Haves" after a baseline is set and agreed upon
- Practice Agile project management
- Asses the risks before the project begins e.g.
    - Missing deadlines
    - Assuming unknown or volatile requirements are clear and fixed
    - Delivery of wrong solution
    - User acceptance testing late in the development
    - Not complying to the principles
    - Non-availability of business roles
    - Expecting a perfect 100% solution
    - Swapping resources in and out during development