Agile Software Projects (CM2020)

Course Notes

Felipe Balbi

October 15, 2020

Contents

Week 1

Key Concepts

- Describe events and sequences of actions in a coherent manner.
- Manage risk.
- Manage assets and resources.

1.01 Module introduction

During this course we will study processes involved in Engineering Software.

1.07 The basics of interaction design

The following reading provides a good introduction to interaction design:

Sharp, H., J. Preece and Y. Rogers Interaction design: beyond human-computer interaction. (Indianapolis, IN: John Wiley and Sons Inc, 2019) Chapter 1 What is Interaction Design, pp.1–34.

Available here.

1.101 What is a project? What do we mean by 'manage'?

Software Project refers to a deliverable component, something we use.

Software needs to be designed considering the different manifestations of computers. For example, computers may have different CPUs, memory size, IP addresses, languages, etc.

Considering the origins of what we consider the modern internet were built and implement in the 1960s, how is the same technology still working today?

We can think of the core standards taht enable internet connectivity as being successful in delivering what is referred to as a *Minimum Viable Product*, MVP for short.

The Internet is a great example of a successful project, both in terms of doing what it was supposed to do and supporting scalability to a massive degree.

One way to ensure a project is successful by its completion, is to think about our end-users or stakeholders. A simple way to do this is to consider what our users might be trying to achieve while using our software.

It's important to consider how we will define our goals for our intended systems and plan our actions according to our intent. It's easy to take a small idea and grow it beyond the scope of what resources we have available, whether that be our budget or timescale or technical capacity.

This can be a simplified way to think about our project as a trade off between three different agendas:

Functionality The features provided would dictate our design;

Resources A project with an infinite number of posibilities or permutations, one would need an infinite number of resources to build it;

Time Projects must have a start and end date. A plan is built around this time budget to ensure the project is delivered at the correct date.

Week 2

Key Concepts

- Describe events and sequences of actions in a coherent manner.
- Manage risk.
- Manage assets and resources.

1.301 Tracking progress, Gantt charts, managing resources and time

How can we track our progress as we go? We would like to ascertain where about we are in our process and know what we need to do to complete certain steps.

Our aims and objectives provide a framing for our goals as they give us an opportunity to define things at a high level. However, we have lots of small elements we're working on all the time.

We need to ensure all these small elements are tracked against the bigger goals. Moreover, our smaller goals is what we're really interested in while progress tracking because we work on small increments to achieve a longer goal.

To track progress, there are many tools one might use:

- Content Management Systems
- Version Control Systems
- Divide by process
- Divide by projects/deliverables
- Holding regular meetings
- Integrated tools use as VSCode, Trello, Basecamp, Slack (or a combination)

While the specific tool to be used is a free choice, one thing to remember is that *Failure to plan is planning to fail.*

We must be vigilant against **feature creep**, where the requirements and scope increase over time. We should also consider dependent processes and the impact that certain steps will have on other moving parts.

Gantt Charts are a good way to think about a linear breakdown of a project. They include milestones that define important points in a project's lifetime. Usually, tasks in Gantt Charts are linked to their relevant resources and dependencies.

Within a Gantt Chart, we can also embed a **critical path**, which is a route from start to finish that explores the manifestation of a project.

A Gantt Chart looks like the table below:

Step		H:M:D	H:M:D	H:M:D	H:M:D	H:M:D
Step 1	Sub Step 1.1	🟩				
	Sub Step 1.2	🟩				
	Sub Step 1.3	🟩				
	Sub Step 1.4	🟩				
	Sub Step 1.5		🟩			
Step 2	Sub Step 2.1		🟩			
	Sub Step 2.2		🟩			
	Sub Step 2.3			🟨		
	Sub Step 2.4			🟨		
	Sub Step 2.5			🟥		
	Sub Step 2.6				🟥	
Step 3	Sub Step 3.1					🟥
	Sub Step 3.2					🟥

In terms of the critical path, we should consider the project in relation to:

- All the activities necessary to complete the project
- Timescaled steps for each state
- Relationship between states
- Critical endpoints where a certain state **must** be reached

The critical path is, of course, an estimate. Unexpected events may occur that change the path of the project or prevent completion of a milestone. One thing we can do, is build a contingency plan in our model, which could mean taking a different approach if something happens that prevents us from completing the original path.

We should also consider adding what's called **Float Time** to the critical path. This will help us build robust timelines where variants can occur. Ideally, we want to start on a process as soon as we can, so if we complete a task sooner than expected, we can move on to other tasks.

We might want to allocate an individual to track the process to ensure accountability.

1.305 Version control systems

Identify each of the commands that you saw in the interactive plugin. You should find and read about each one of these commands at the following location:

`https://git-scm.com/docs`

1.401 The process of interaction design

Read Sharp, H., J. Preece and Y. Rogers Interaction design: beyond human-computer interaction. (Indianapolis, IN: John Wiley and Sons Inc, 2019) pp.37–55.

Week 3

Key Concepts

- Consider the wider implications of building a system for purpose.
- Explore formal specifications from both a functional and technical perspective.
- Identify key stakeholders, challenges, risks and innovations.

2.01 Introduction to requirements and specification

Requirements vary depending on the target audience. While an average person may require around 2000 Calories per day, a competitive power lifter may require substantially more. When thinking about requirements, we need to consider the broad spectrum of possibilities.

In the case of systems, we generally describe actions in the form of use cases. We assume a user wants to achieve something with our product and consider the steps necessary to achieve that goal. We also think about which steps or tasks are more important than other, which can be deferred or cancelled.

We need to consider all these details in order to provoke further considerations.

One way to define requirements is as a need or desired outcome for or from a system. If we're designing a payment system, it may be required to process transactions or accept certain payment methods.

Requirements can be specified using formal documentation with a discernible outcome. In order to write these, we must first a good formal understanding of what we're trying to achieve, only then can we specify requirements in a measurable manner.

The process that underpins this is referred to as Specification. One way to create specification is through modelling techniques such as UML, or Unified Modelling Language.

UML offers several benefits for specifying systems. For example, we get a visual representation of the relationships between entities or objects within our system.

UML has several types of diagrams, for example Use Case Diagrams and Class Diagrams, which are the most commonly used.

2.101 Purpose of gathering requirements

Gathering requirements helps us understand the scope of the project. Moreover, requirements help us with:

- specifying things that can be measured and tracked

- keeping all stakeholders in the loop
- formalising contracts and legal obligations
- defining complex relationships and systems in a meaninful way

In terms of specifications, there are two main categories:

Functional concerned with the ability to perform certain actions

Technical concerned with the manifestation of a system to achieve said outcomes

Both types of specifications describe a set of actions, intentions, systems and outcomes.

2.102 Introduction to requirements gathering

Read Sharp, H., J. Preece and Y. Rogers Interaction design: beyond human-computer interaction. (Indianapolis, IN: John Wiley and Sons Inc, 2019) chapter 11 pp.385–418.

2.201 Modelling requirements

Use Cases define a set of actions necessary to achieve a goal. This goal is describe as a **main success scenario**. A user wants to achieve something and follows a set of steps.

There are many ways to describe use cases, but in general they representations with a series of systems and how users engage with those parts of the systems at different levels.

Class Diagrams, on the other hand, describe classes or objects. They capture the classes' attributes and/or operations.

These are very good for defining relationships between objects in a system and some explicit structure. They can also describe depedencies and abstraction.