Agile Software Projects (CM2020)

Course Notes

Felipe Balbi

November 30, 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.

Week 4

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.

2.301 User-centred design

User-Centered Design or UCD is a framework of processes where usability goals, user characteristics, environment, tasks and workflow of a product, service or process are given attention at each stage of the design process. The main concern is with what a given user wants to achieve with our system and the means by which they intend to achieve that.

UCD is a cyclic process. We don't specify some requirement and carry on with development without thought or care. Instead, our aim is to constantly refine our designs and developments in relation to feedback from the users; including processes such as regular stakeholder meetings, testing and feedback.

There are a number of stages involved in User-Centered Design process, we may start by identifying a want or need that a user has. Once we're happy with our ideas, we can begin designing through a range of fidelities from sketches to interactive prototypes.

What we're really doing here is a series of iterative steps. Based on user feedback, we refine our design also considering the broader implications in terms of universal design and in terms of accessible design.

During this process, there are some common mistakes we want to avoid:

Wrong Assumptions we can't make assumptions about what our users want, need or what their experience might be. Empirical methods will help guide the work and provide quality metrics for measuring success. *Test early and test often*

Planning too far into the future we are unable to consider everything from the start. Instead, we can follow a process that allows us to adapt to changes in our ecosystem.

We should try to begin with robust requirements:

- Context of use
  - Intenttions, goals

  - Stakeholders
  - Market forces
- Constraints and mitigating factors
  - Mandated constraints
  - Conventions and formal processes
  - Knowns and assumptions
- Scoping
  - The scope of the work
  - The scope of the product
  - Functional and data requirements
  - Minimum viable product

For each of our design choices, we should consider its impact in terms of:

- Aesthetics
- Usability and accessiblity
- Performance
- Operational
- Maintainability and support
- Security
- Cultural
- Political
- Legal
- Ethical
- Social

Projects may fail and we should remain conscious of project pitfalls:

- Risks
- Costs
- Documentation and training

Week 5

Key Concepts

- Identify interesting and valuable sources of information
- Make decisions based around research findings
- Consider the wider implications of building a system for purpose.

3.01 What is research?

We use research methods to explore the contextual implications for the design and development of our software applications.

The goal is to build an application that adds efficiency and utility to a given workflow, therefore we must define what **efficiency** and **utility** mean.

We may want to consider similar products and services that exist in the market in order to design a product or service that adds value to our customers.

Research is a systematic investigation into something unknown. During research, we aim to build knowledge or insights that relate to our area of application.

Research explores our understanding of a given space in relation to a number of factors. We tend to focus on things we can measure to determine their effects on other things.

We aim at isolating variables as best we can in order to avoid confounding factors that may impact our findings. We have to ensure that our research is robust and meaningful to our research agenda.

Research also has a **validity** associated with it. This can tell us whether or not we are measuring what we say we are measuring.

Another important topic is that of **reliability**. We want to ensure our results have meaning from the context of experimentation.

3.101 Understanding the differences between quantitative and qualitative research

There are two main types of research:

Quantitative explores measures, often relying on mathematical models and tools such as statistics. Tends to focus on **what** is happening, **who** is involved, **where** it's happening, **when** it's happening and so on

Qualitative focuses on dynamic realities such as emergent themes and behaviors. Often focuses on **how** and **why** things are happening

3.102 Introduction to research methods

- Read Sharp, H., J. Preece and Y. Rogers Interaction design: beyond human-computer interaction. (Indianapolis, IN: John Wiley and Sons Inc, 2019) chapters 8 and 9 pp.259–34.2

3.201 Identifying useful sources of information

Newspapers generally a good source of information, however most publications probably do have some intrinsic bias.

Books consider the publisher, publication method, reviews, suggests (e.g. from educators)

Social Media could be anything from very good to very bad

Peer Reviewed Sources High barrier to entry for publication. In general the process improves quality

Data even data can be biased. The method used for sampling may impose biases

3.202 Team activity and deliverable: doing prerequisite investigation

- Department for Education Realising the potential of technology in education: A strategy for education providers and the technology industry. (2019).

Week 6

Key Concepts

- Consider the wider implications of building a system for purpose.
- Identify interesting and valuable sources of information
- Make decisions based around research findings

3.301 How to analyse the competition and place your 'solution' in a space

A stakeholder is any person who has an interested in what we're doing. Our employer, for example, or those working on supporting roles of the business, team members, users and so on.

There's a vast number of people who would be considered as a *stakeholder* in a system, even for small systems; therefore we can apply STEEPLE analysis to try and understand our external environment.

STEEPLE is concerned with the following aspects:

Social changes in culture, demographics, and attitudes

Technological changes in technology that can affect our system

Economic financial implications of our decisions

Environmental concerned with environmental implications of our product

Political labour law, taxation obligations, etc

Legal legal implications concerning data and privacy, for example

Ethical besides legal ramifications, we want to ensure our approach is ethical

3.302 SWOT analysis

SWOT Analysis is a strategic planning technique to help an individual or organization identify strengths, weaknesses, opportunities, and threats related to business competition or project planning.

Strengths What's *special* or *unique* about the project?

Weaknesses What do we lack that may reflect on the project?

Opportunities What environmental or business qualities can we exploit to our advantage?

Threats What can cause trouble to the project?

Alternative models

- UNICEF SWOT and STEEPLE

Week 7

Key Concepts

- Understand user-centred design and what it means to employ said methods and strategies.
- Consider how user-centred design techniques can be used to iteratively improve software quality.
- Formalize your findings and develop them as a series of prototypes.

4.01 User-centred design in a nutshell

As mentioned before, User-centred design is an iterative set of steps which focus on involving the users in the design process, development and testing of our software (i.e. our *Software Development Lifecycle*).

A key concern in UCD is constantly re-evaluating the context of use. This helps us to specify (using e.g. UML) requirements which, in turn, help inform our designs.

Because our conceptual understanding of how a system works is, in general, lacking, UCD provides a process to try and remove biases during system design an development.

Ultimately, we want to produce better systems. We achieve that by taking into consideration feedback from our stakeholders. As we incorporate this feedback, the resulting product tends to better fit our users' requirements.

We also want our designs to fit the widest possible demographic. This is referred to as **Universal Design**.

The result of all this effort also has tangible socio-technical benefits. Through an iterative design process we get:

Minimal training requirements when users are involved in the design of the system, they won't need training afterwards

Minimal resistance to change the system embeds users' values, therefore they will be more willing to adopt the new system

Improved requirements because users are involved throughout the design and development, we have many chances to clarify requirements, which results in a much clearer set of requirements for the system

4.03 Design and prototyping

- Sharp, H., J. Preece and Y. Rogers Interaction design: beyond human-computer interaction. (Indianapolis, IN: John Wiley and Sons Inc, 2019) chapter 12 pp.421–466.

4.101 Why 'user-friendliness' is a false metric

User-friendliness is not a universal, measurable metric. Its meaning changes depending on context.

Our systems will, hopefully, be used by hundreds of different users from different backgrounds. How can we design a system that's *friendly* to this myriad of different users? In short, we can't.

We can say that user-friendliness is a **false metric**, considering that there's no universal truth for what user-friendliness means. What is friendly to one person could be unfriendly to another.

This is one reason why we can find so many options of SW doing roughly the same thing (IDEs, Operating Systems, Text Editor, etc).

Instead of looking at user-friendliness, we should aim at understanding **Usability**. Here, our primary concerns are:

Effectiveness can I complete a task?

Efficiency How quickly/easily can I complete a task?

Satisfaction Do I enjoy using the system?

Usability grew from the work on pioneers such as Jakob Nielsen, Donald Norman and Ben Shneiderman. The real revolution, however, came about with the increase in usage of systems.

Along the way a series of innovations paved the way for big paradigm shifts. For example, the early Xerox Alto and Windows operating systems gave us a Graphical User Interface (GUI) operating system while affordable broadband internet largely increased the number of internet users. More recently the original iPhone revolutionized the mobile phone market and MOOCs have changed the way education is delivered.

4.102 Introduction to user-centred design

- Sharp, H., J. Preece and Y. Rogers Interaction design: beyond human-computer interaction. (Indianapolis, IN: John Wiley and Sons Inc, 2019) chapter 12 pp.471-491.

4.201 Test early, test often, iterate

Testing is important in all Software Engineering. When we talk about *Iterative User Testing*, we might think of a test as way to **verify** the our systems behaves **as specified**.

We can combine our testing methodology with our usability definition. For example, from an effectiveness and efficiency point of view, we can collect metrics on whether a user completed a specific task and how long did it take:

Task	Completed?	Time
Task 1	Yes	00:23:17
Task 2	No	N/A
Task 3	Yes	00:11:46

User satisfaction testing is a bit more subjective. We probably want to collect more qualitative data regarding user satisfaction. We could watch people using our system to identify interaction patterns, or try to identify hesitation to counter user feedback.

Sometimes, observational data and metrics may paint a different pictures, we have to resolve these sorts of problems as well.

In order to *frame* our use cases, we may want to create *Personas*. They are an amalgamation of data that can help us think about what a typical user might look like. It should help us targeting users from a particular demographic.

4.202 Heuristics for design

- Sharp, H., J. Preece and Y. Rogers Interaction design: beyond human-computer interaction. (Indianapolis, IN: John Wiley and Sons Inc, 2019) chapter 12 pp.549-576.

Week 8

Key Concepts

- Understand user-centred design and what it means to employ said methods and strategies.
- Consider how user-centred design techniques can be used to iteratively improve software quality.
- Formalise your findings and develop them as a series of prototypes.

4.301 Why do we prototype?

At some point during our research phase, we will start forming more concrete ideas about our designs. It will become necessary to **test** some of our assumptions to make we're following the right path.

A protype helps us test these assumptions. It's essentially a smaller version of our application, with limitted functionality built solely to test a hypothesis. We could test a single feature, a certain algorithm, a UI Design, etc.

We could depend on some parts of the system to be fully working to test a certain prototype. Perhaps, a registration page prototype needs the backend authentication and registration fully working.

There are *levels of prototype* to consider:

Low-fidelity prototypes often paper-based mock-ups of areas of our application with placeholder content

Medium-fidelity prototypes an extension of low-fidelity prototypes. Often contain some more realistic data, images, etc. May also include limitted functionality

High-fidelity prototypes typically functional or semi-functional parts of our application. Generally closer to what the final product will look like

When it comes to testing assumptions, it's important to test frequently to avoid spending too much time pursuing a false premise. Besides, a series of small incremental changes is easier to reason about and, in general, improves the quality of the final product.

Given all this information about levels of prototyping, there some prototyping techniques to be discussed.

Sketching and Storyboarding primitive design tool used to generate ideas and concepts. Usually employed early in the design phase

Wireframes provide a standardised framework for design placeholders. Focused on *structure* and how pieces fit together

The drawbacks of prototypes are that they won't give us a good sense of what to expect in the final version. They may also cause confusion to some users who may expect a fully functional system, and they can consume valuable resources (time and monetary).

Week 9

Key Concepts

- Present research around your topic.
- Explore a series of iterative design cycles that culminates in a proposed design solution.
- Present your prototype to be assessed.

5.01 Main components of your proposal

A project proposal should include:

- Work done so far
- Project goals
- How we intend to achieve our goals

The proposal should be descriptive and informative while also trying to convince the audience that the proposal is sensible. It's as much about **process** as it is about **product**.

A good product comes about through systematic and thorough processes.

A useful technique is the "300 second talk", where we try to condense the project proposal document into (at most) a 300 second narrative. This helps maintain focus to key problem space, deliverables and challenges that are likely to emerge.

5.101 Assessment strategy, deliverables and expected outcomes

The expectation is a project similar in quality to what we will build during Level 6 Individual Project.

From a practical standpoint, the report should clearly identify our **Minimum Viable Product** or MVP and the **Delivery Stragegy**. In summary, the expectation is:

- A basic solution to a problem space identified
- Characteristics that suggest a value proposition
- Evidence that thought has gone into all stages of the development process

A good MVP is something that (sort of) works. We shouldn't focus too much on feature-rich applications. The user should always be involved.

From the proposal, it should be clear:

- What is being delivered
- Which set of processes are useful
- The timescale/dependencies relationship
- System specification
- Scope, limitations, and risks
- Requirements
- Where tool sits in the marketplace
- Evidence of prototyping and iteration
- Some simple tests/evaluation steps
- A strong rhetoric that informs and describes

Week 11

Key Concepts

- Project management.
- Group work.
- Iterative Development.

6.01 What is and what is NOT agile?

The Agile Manifesto is a set of *standards* for setting up projects with minimal bureaucracy.

The guidelines are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

It's similar to user-centred design practices. Agile and UCD can work well together.

Customer satisfaction is achieved by following a process of continuous improvement. Therefore, we build products by iterating and improving on our MVP. This has the added benefit of reducing friction with customers, who get a working version of our software after each iteration.

The conclusion is that collaboration benefits everyone. Customers get a working software in increments, which reduces need for training and disruptions, developers get to focus on small features at a time.