Development Process

What is Software Engineering?

“The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software” IEEE standard 610.12-1990.

Software Engineering focuses on: » the practicalities of developing and delivering useful software » professional software development in teams » producing software products

LCCS Specification

1.19 Identify features of both staged and iterative design and development processes

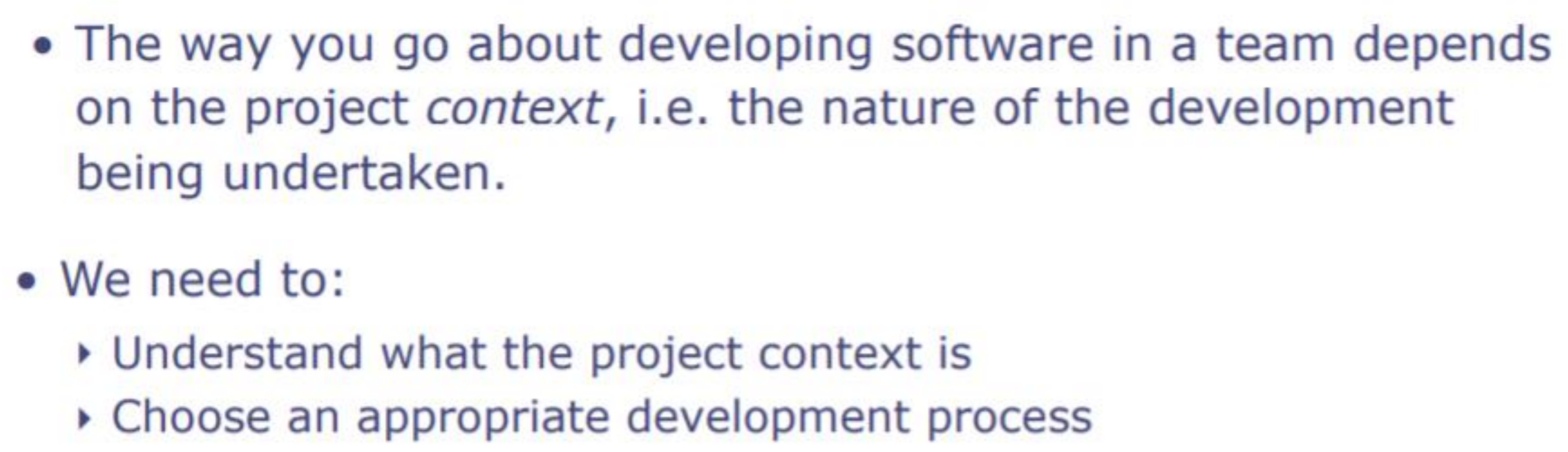
1.23 reflect and communicate on the design and development process

- Professional software development is hard.
  - » Software products are very complex.
  - » Teams are large.
  - » Teams are geographical distributed.
  - » Team members have varying levels of expertise.
  - » User requirements keep changing.
  - » Technology keeps changing.

- The discipline of Software Engineering has produced a range of processes, tools and techniques to make it easier to development, deliver and maintain software products.

Software Development Process

Context



- The way you go about developing software in a team depends on the project *context*, i.e. the nature of the development being undertaken.
- We need to:
  - Understand what the project context is
  - Choose an appropriate development process

Context

- Most projects are complex and it is difficult to predict the best solution in advance
- Some project are complicated but an expert could predict the best solution.
- Some project are obvious and it is easy to predict the solution - for example deploying software to your 100th customer.

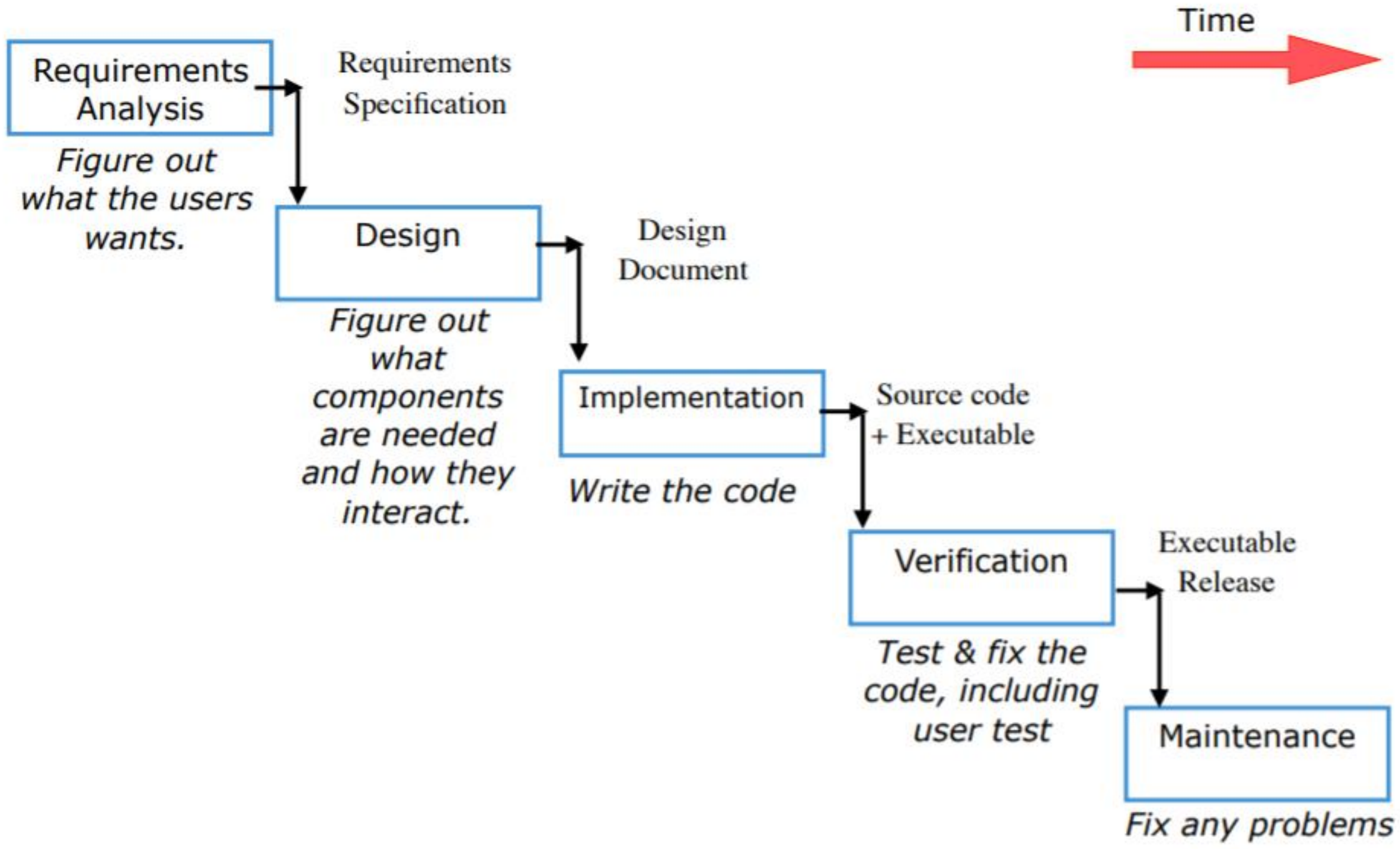
Process

There are staged/planned/waterfall design processes and iterative/agile processes

Waterfall Process

- It is one of several plan driven processes.
- Plan-driven development works well if you are applying it to problems that are well defined, where the work is predictable, and the results are unlikely to undergo any significant change.

- This is the traditional software development process.


Time
Requirements Analysis
Figure out what the users wants.
Requirements Specification
Design
Figure out what components are needed and how they interact.
Design Document
Implementation
Write the code
Source code + Executable
Verification
Test & fix the code, including user test
Executable Release
Maintenance
Fix any problems

Advantages

Simple and easy to understand and use

Clearly defined stages.

Well understood milestones.

Easy to arrange tasks.

Disadvantages

High amounts of risk and uncertainty.

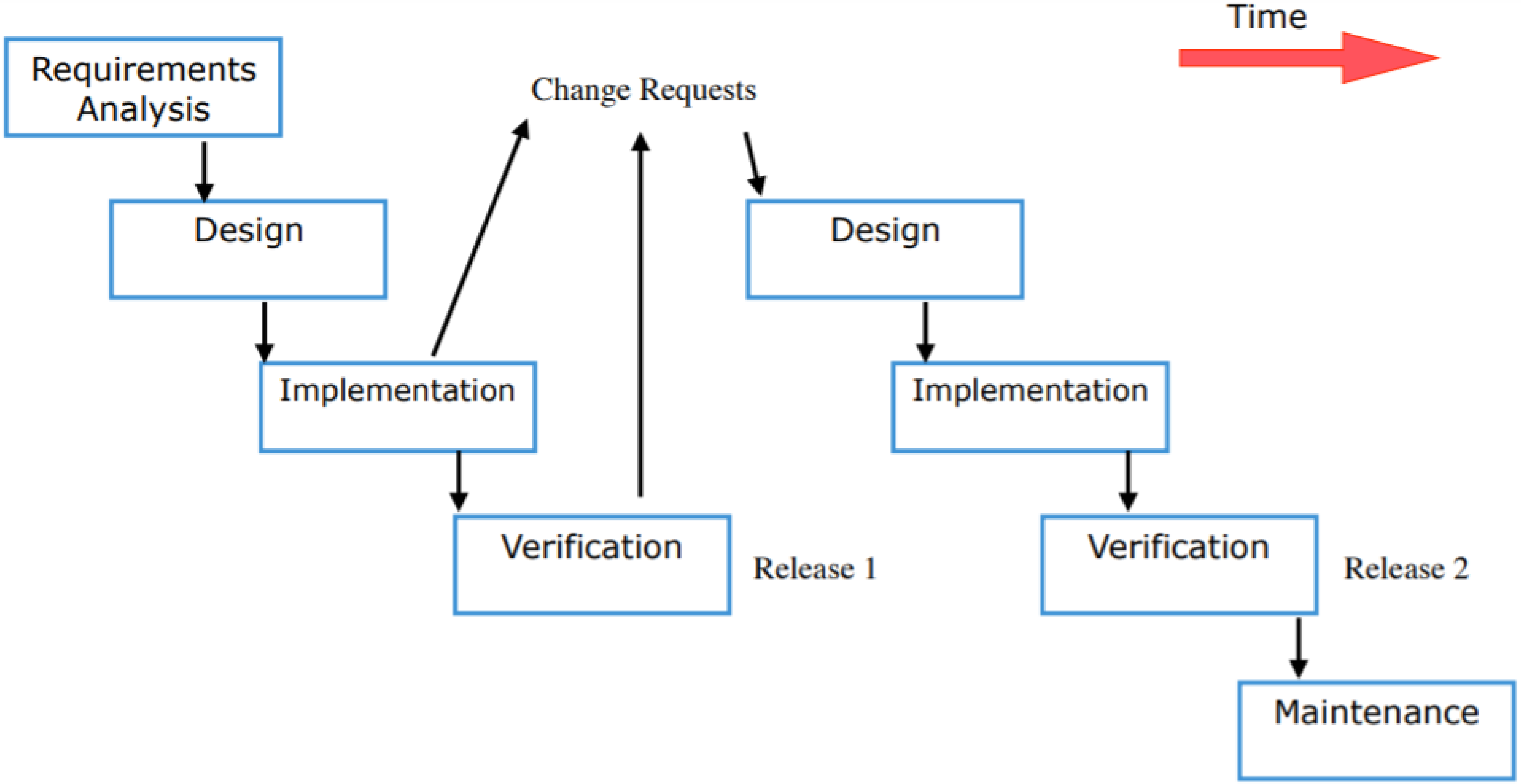
Poor model for long and ongoing projects.

It is difficult to measure progress within stages.

Cannot accommodate changing requirements.

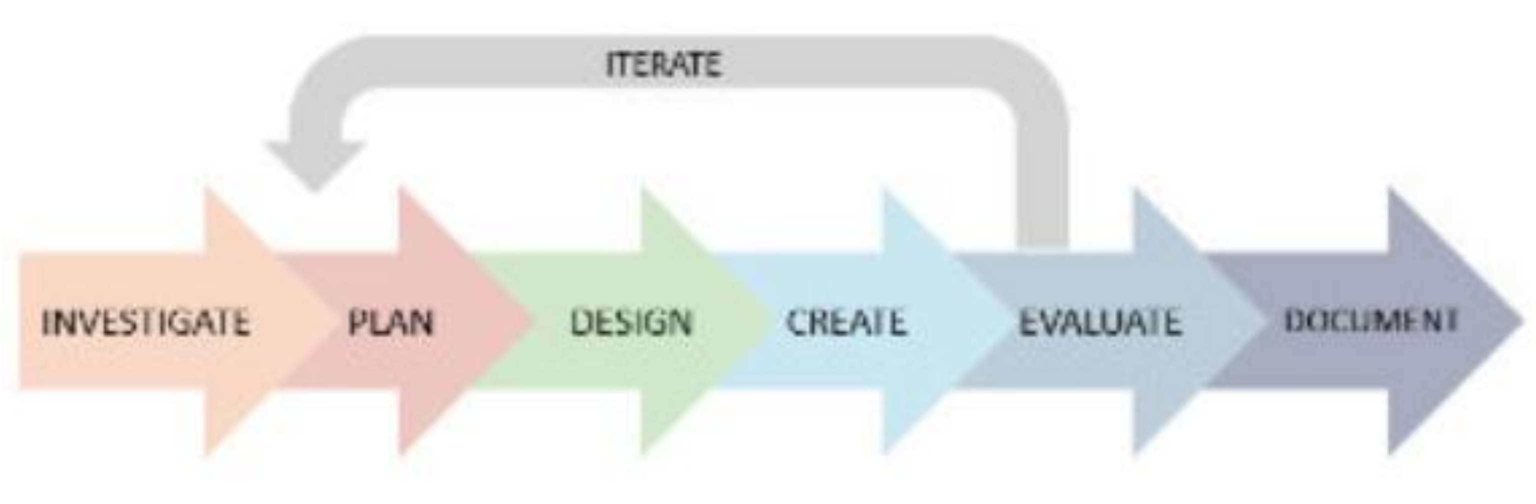
Iterative Process

- “Iterative development acknowledges that we will probably get some things wrong before we get them right and that we will do things poorly before we do them well.”
- Iterative development is planned re-work.
- Use multiple passes to improve what we are building.
- Use Change Request documents to record needed changes and pass the fixes on the to next stage.



Could be several internal releases before the customer release. Change Request review board decides when to include changes.

SW Dev Cycle used in LC Computer Science



Iterative Process

- Works pretty well.
- Problems tend to get thrown “over the wall”, either to the next stage or the next release.
- Leads to single discipline teams.
- Management often likes it because of the control points.

Advantages of Iterative Development

- Communication and Collaboration - frequent and close cooperation between software team and end users.
- User Feedback - Team can deliver early versions of the product for feedback. This means better customer satisfaction and end user experience
- Eliminate Issues early - Constant review helps eliminate issues early in the project
- Reduces the overall Project cost - Software testing is part of the review process. This means errors are caught early on and can be fixed before moving on to the next stage.

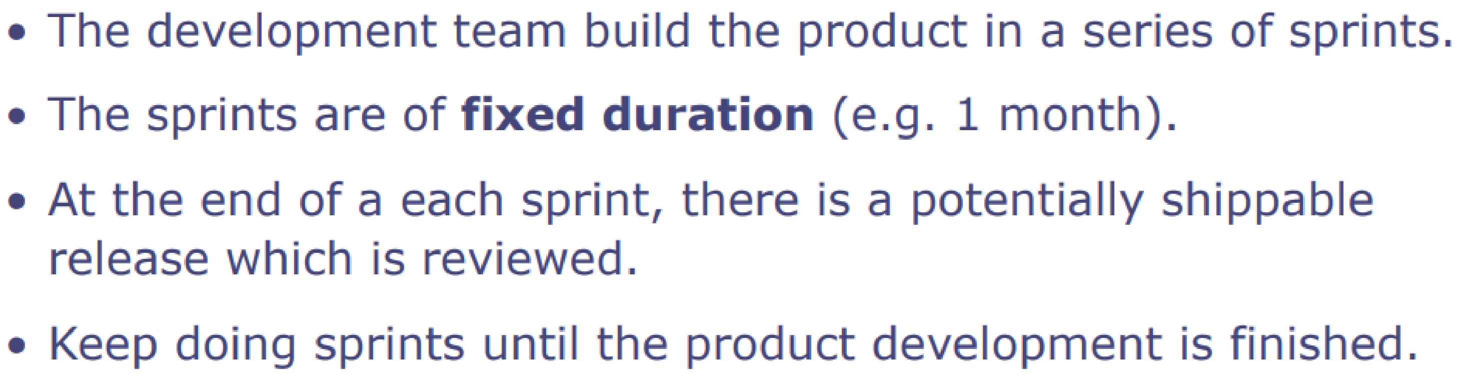
Agile Software Development

- Agile development is an “umbrella” term for several iterative and incremental software development methodologies.
- While each of the agile methodologies is unique in its specific approach, they all share a common vision and core values. They all fundamentally incorporate iteration and the continuous feedback that it provides to successively refine and deliver a software system.

https://www.youtube.com/watch?v=q_R9wQY4G5I&t=20s

Scrum

Scrum describes a set of meetings, tools and roles that work together to help teams structure and manage their work.



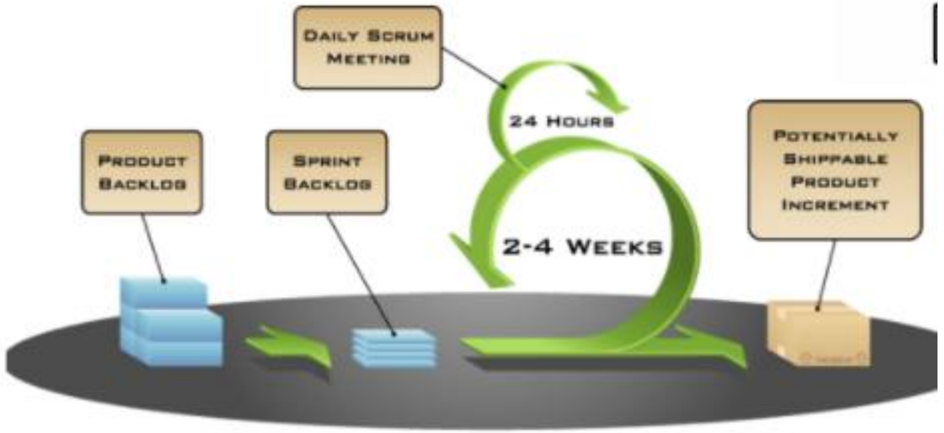
- The development team build the product in a series of sprints.
- The sprints are of **fixed duration** (e.g. 1 month).
- At the end of a each sprint, there is a potentially shippable release which is reviewed.
- Keep doing sprints until the product development is finished.

Agile - Scrum involves separating the project into **sprints**. The list of sprints (stages of projects) make up the **Product Backlog.**

The **sprint backlog** contains the tasks relating to the current sprint. Each sprint lasts for a period of time. At the end of the sprint, there is a **potentially shippable product** as that stage of the project should be tested and working. The next sprint (stage) will then be started.

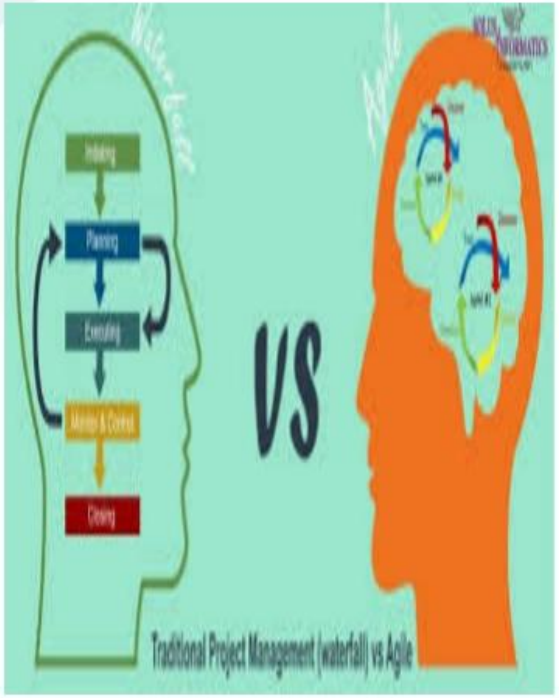
In a software development team, there would be a **daily scrum meeting** where each team member would update on their progress. Your Coursework is an individual project but your daily scrum meeting would be where you report on your daily progress, including problems.

https://www.youtube.com/watch?v=q_R9wQY4G5I&t=20s



Waterfall v Iterative (Agile)

•**Agile** is an incremental and iterative approach; **Waterfall** is a linear and sequential approach.
•**Agile** separates a project into sprints; **Waterfall** divides a project into phases.
•**Agile** helps complete many small projects; **Waterfall** helps complete one single project.
•Requirements are prepared everyday in **Agile**, while requirements are prepared once at the start in **Waterfall**.
•**Agile** allows requirement changes at any time; **Waterfall** avoids scope changes once the project starts.



Stages of Software Development Process

Software Development Process

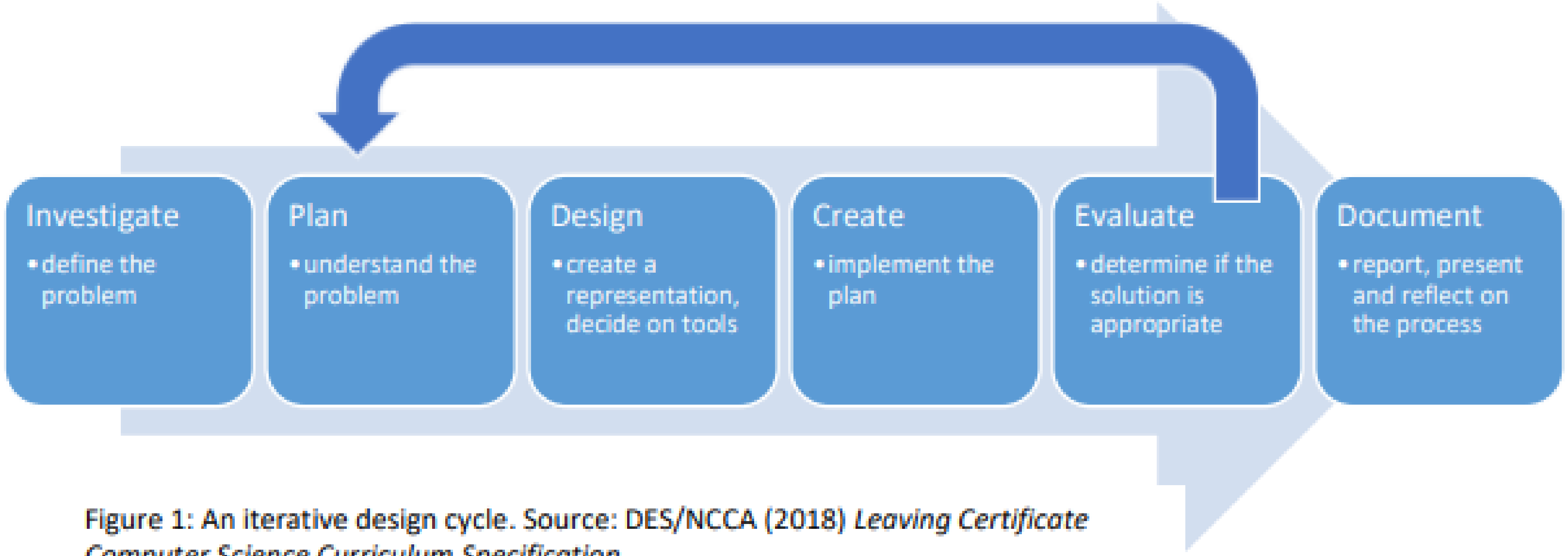
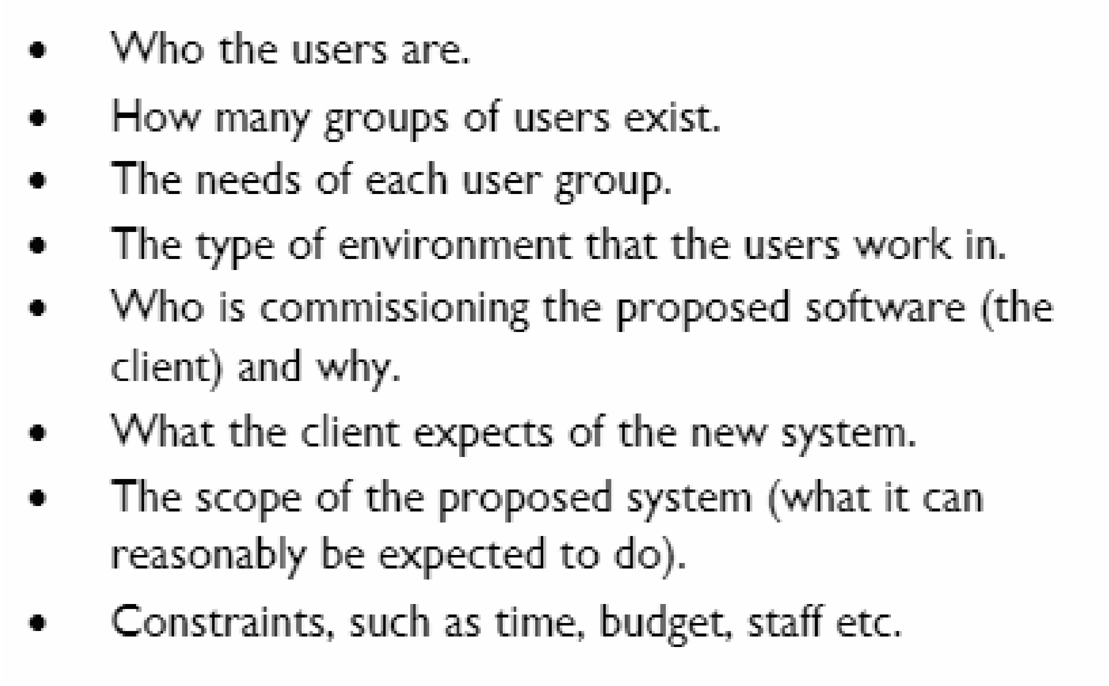

Investigate
• define the problem
Plan
• understand the problem
Design
• create a representation, decide on tools
Create
• implement the plan
Evaluate
• determine if the solution is appropriate
Document
• report, present and reflect on the process

Figure 1: An iterative design cycle. Source: DES/NCCA (2018) *Leaving Certificate Computer Science Curriculum Specification*.

1. Investigate - Define the Problem

Identify and fully analyse the problem



- Who the users are.
- How many groups of users exist.
- The needs of each user group.
- The type of environment that the users work in.
- Who is commissioning the proposed software (the client) and why.
- What the client expects of the new system.
- The scope of the proposed system (what it can reasonably be expected to do).
- Constraints, such as time, budget, staff etc.

Investigate - Define the Problem

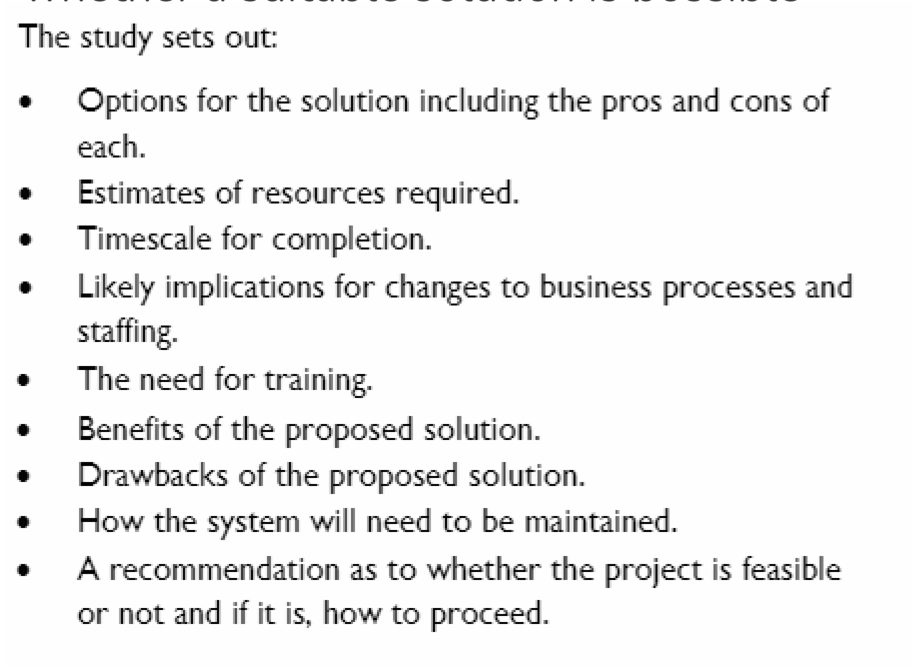
Gather information - interviews, questionnaires, observations, samples of existing documentation

Requirements Specification

- **User requirements**
- **Client requirements**
- **Hardware/Software requirements**

Investigate - Define the Problem

Carry out a **Feasibility study** - report written for client detailing whether a suitable solution is possible



The study sets out:

- Options for the solution including the pros and cons of each.
- Estimates of resources required.
- Timescale for completion.
- Likely implications for changes to business processes and staffing.
- The need for training.
- Benefits of the proposed solution.
- Drawbacks of the proposed solution.
- How the system will need to be maintained.
- A recommendation as to whether the project is feasible or not and if it is, how to proceed.

2. Plan - Understand the Problem

A plan is necessary to ensure that everyone knows **what** must happen, **when** and by **whom.**

The plan should cover the various stages of the project broken into tasks.

Plan - Understand the Problem

- Timescale
- Dependencies
- Hardware and Software requirements
- Staffing

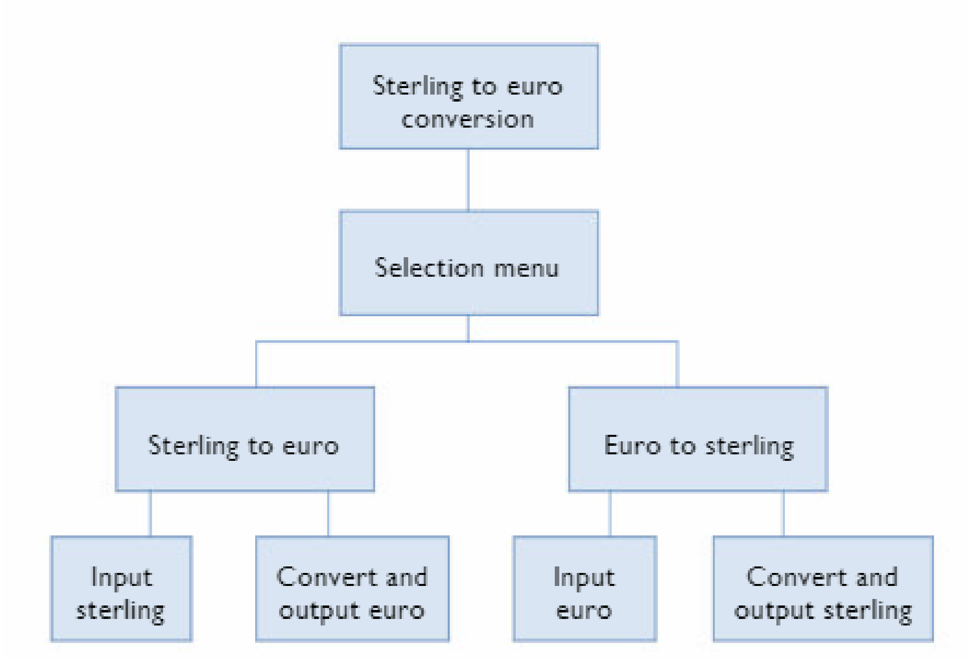
3. Design - Create a representation, decide on tools

Approaches to design

Top Down Design

This approach starts with the main system at the top and breaks it down into smaller units in a hierarchical fashion.

Each unit is further broken down into smaller units.

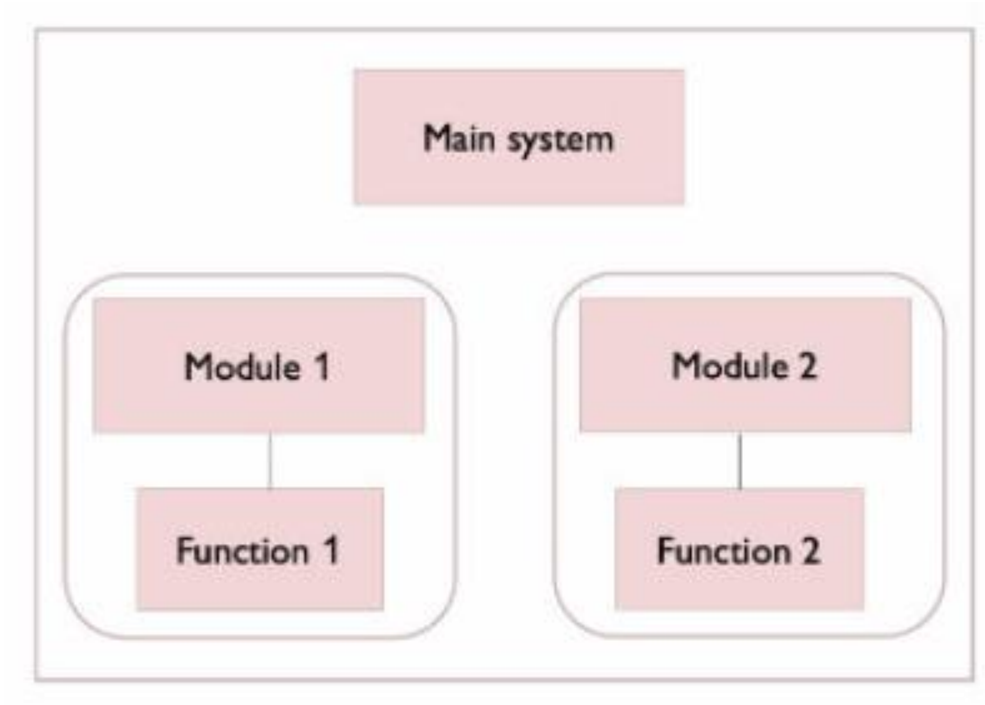


Approaches to design

Modular Design

System is divided into smaller units called modules.

Each module must represent a single self contained task or function that can be created independently of the other.



Approaches to design

Prototyping

A prototype is an early model of a product built to test a concept in advance and to get feedback from users.

Typically features only a few key aspects of the whole system.

Enables users to give feedback at an early stage - additional user requirements can be identified and problems recognised early on.

Often used in Engineering, Software Design.

Most valuable when designing user interfaces.

Approaches to design

<u>Flowcharts</u>

Represents the flow of operation of the system.

Clear graphical way of communicating

Difficult when complex logic is involved.

Design

Algorithms

All functionality should be described.

These can be shown using

Pseudocode
Normal Language
Flowcharts

Approaches to design

<u>Data and Variables</u>

All data should be listed along with their datatype.

Particularly important when dealing with databases

How will data be stored and backed up

Privacy concerns regarding data

4. Create - Implement the Plan

This stage involves implementing the plan/design.

Programmers create the system based on the work carried out in the previous stages.

The design might be modified due to issues that arise during implementation.

Unit Testing is carried out during this stage.

5. Evaluate - determine if the solution is appropriate

Testing to see if the solution works consistently.

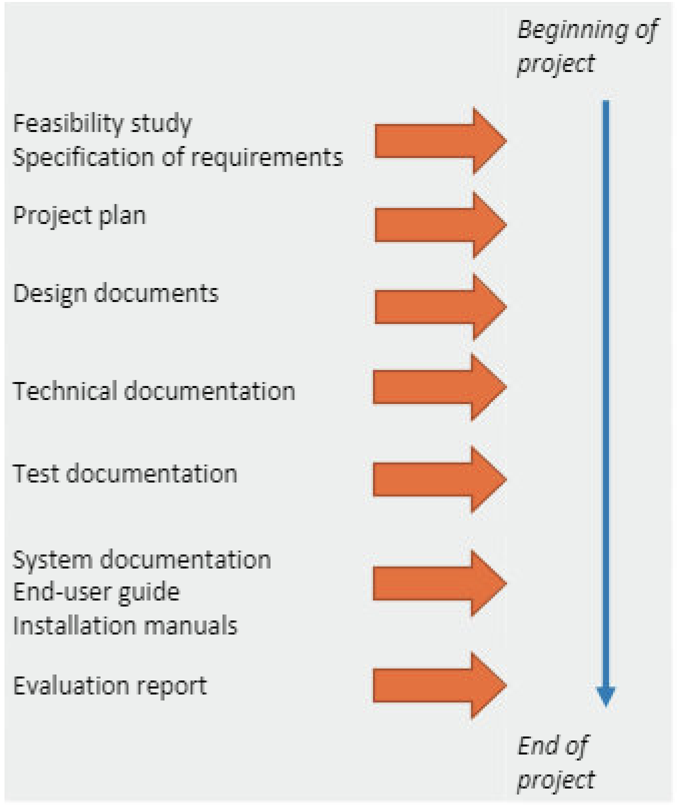
Testing to make sure that the software does not do anything it shouldn’t.

More on Testing separately…

6. Document - report, present and reflect on the process

Records of all meetings, correspondence, communications, actions agreed, changes to requirements and design should be stored.

Evaluation of the project


Beginning of project
Feasibility study
Specification of requirements
Project plan
Design documents
Technical documentation
Test documentation
System documentation
End-user guide
Installation manuals
Evaluation report
End of project