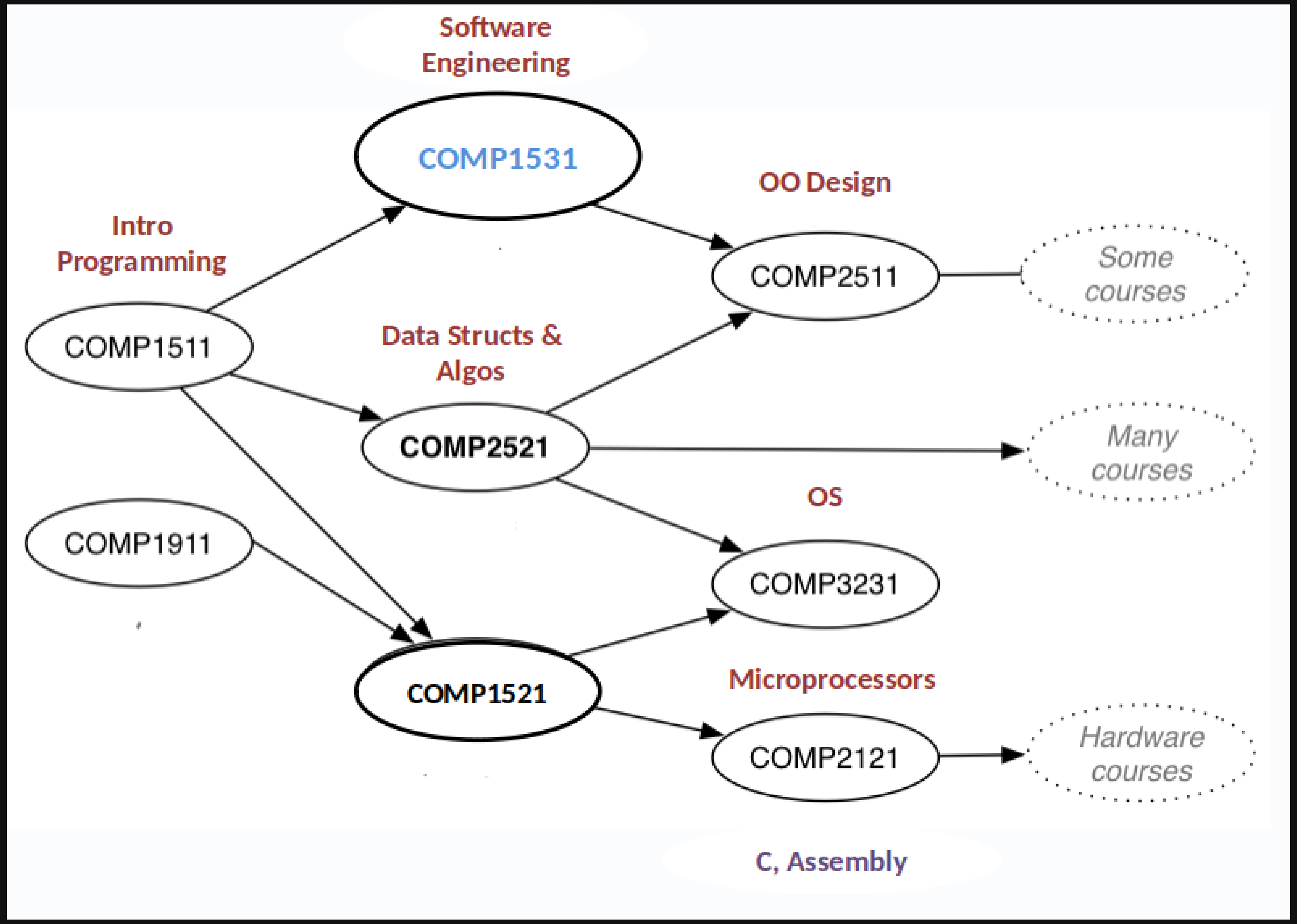
COMP1531

1.1 - Introduction to Software Engineering

Why should this course be important to you?

Relevance to your program



What we will assume

That you are at least a mediocre C programmer

- Control structures
- Data types
- Abstraction
- Testing

Overview

Software Engineering

Web & HTTP

Teamwork & Management

Python

Teaching Strategies

- **Lectures**:
  - 2 x 2 hours per week
  - Lecture recordings on Webcms3 & Echo360
- **Tutorials**:
  - 1 hour per week
  - Discussion & Share thoughts
- **Labs**:
  - 2 hours per week
  - Check in with your tutor
  - Either:
    - Complete and mark activities
    - Review milestone for project

Other Help

- **Help Sessions:**
  - Help sessions beginning in week 2
  - Tutors and Lab Assistants will attend

What is gitlab?

https://gitlab.cse.unsw.edu.au/

Gitlab is an online tool that we use (based on "git") to distribute tutes, labs, and projects

Lab Marks

- A+ => 2/2
- A => 1.75/2
- B => 1.5/2
- C => 1/2
- D => 0.5/2

Assessment

Item	Weighting	Notes
Class Mark	20%	2% per lab
Project	50%	3 milestones
Exam	30%	Hurdle No sympathy supps

- Labs need to be submitted by Sunday that week
- Labs need to be demonstrated in your labs (that week or next)

- Week 1/2: Talk about Project
- Week 10: Talk about exam

Getting Help

- Step 0: Your team
- Step 1: Piazza forum
  - Look for answers before posting
  - You were invited "z5555555@unsw.edu.au"
- Step 3: Help Sessions
- Step 2: Emailing Tutor / Assistant Tutor
- Step 3: Lecturer cs1531@cse.unsw.edu.au

System

- Any operating system is fine for this course.
- Windows may require a bit of configuration for some items (but this is a lot easier now with Windows Subsystem for Linux)
- You could do this course only on the CSE machines, so don't stress about your computer

Software Engineering

- What's the difference between **Computer Science** and **Software Engineering?**

Software Engineering

- What's the difference between **Computer Science** and **Software Engineering?**
  - At UNSW, Software Engineering is an extension of Computer Science, where we give extra focus to how software systems are built, how to manage projects, and how to test software to provide quality assurance.
- Do you need to be a **Software Engineering student** to be employed as a **Software Engineer?**

Software Engineering

- COMP1511: Learning programming by writing code to solve problems
- COMP1531: Learning Software Engineering by using programming in the context of the software development lifecycle (SDLC)

Software Engineering

Applying engineering methodologies to our current programming capabilities.

IEEE definition: "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software."

We're going to build thought-out, testable, scalable software that is meets set out requirements and is easily maintained

Being a software engineer

- Is there a perfect software engineer?
  - No, because a good software engineer is like being a good Doctor. It's principles that inform you, more than specific methods

That was "What" - but "Why"?

What happens in a world without
good software engineering
principles being used?

That was "What" - but "Why"?

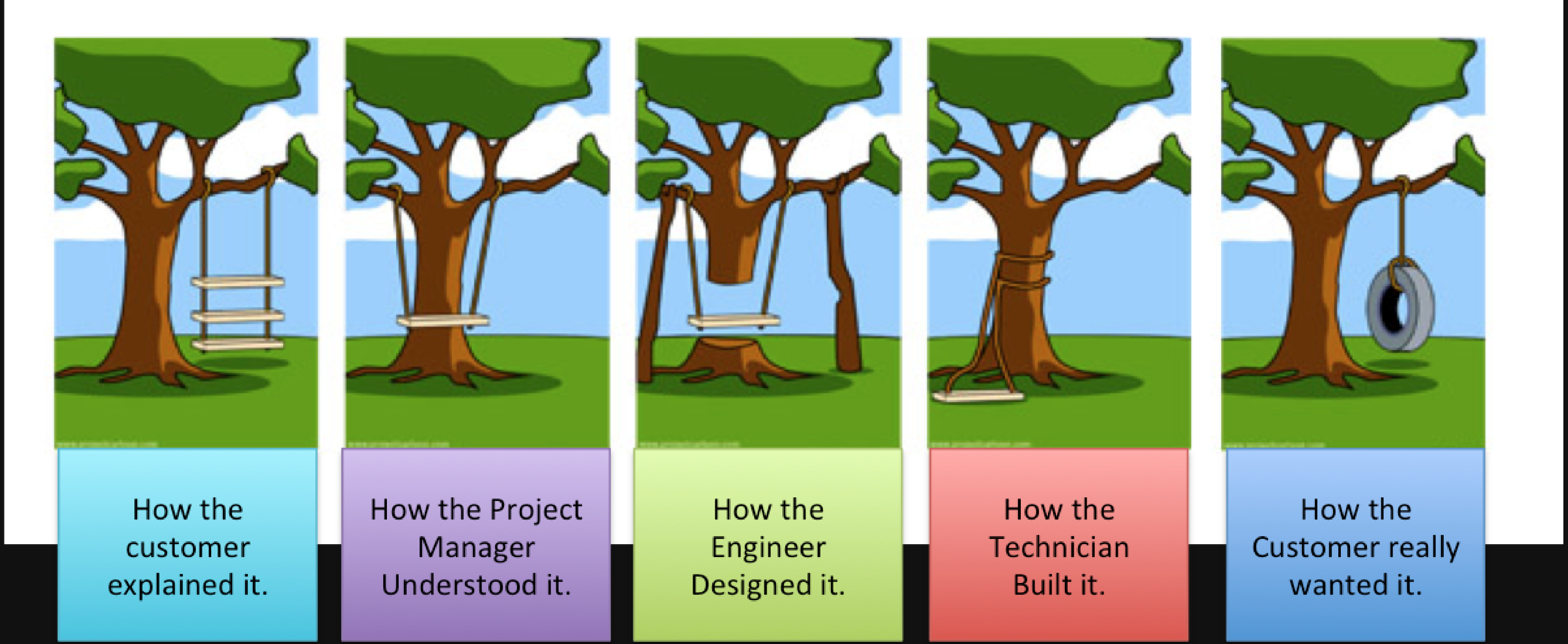
Software engineering fundamentally exists to allow **businesses** and **organisations** to de-risk their business goals compared to just hacking away.

- More predictability about time and budget
- Minimise errors and increase reliability

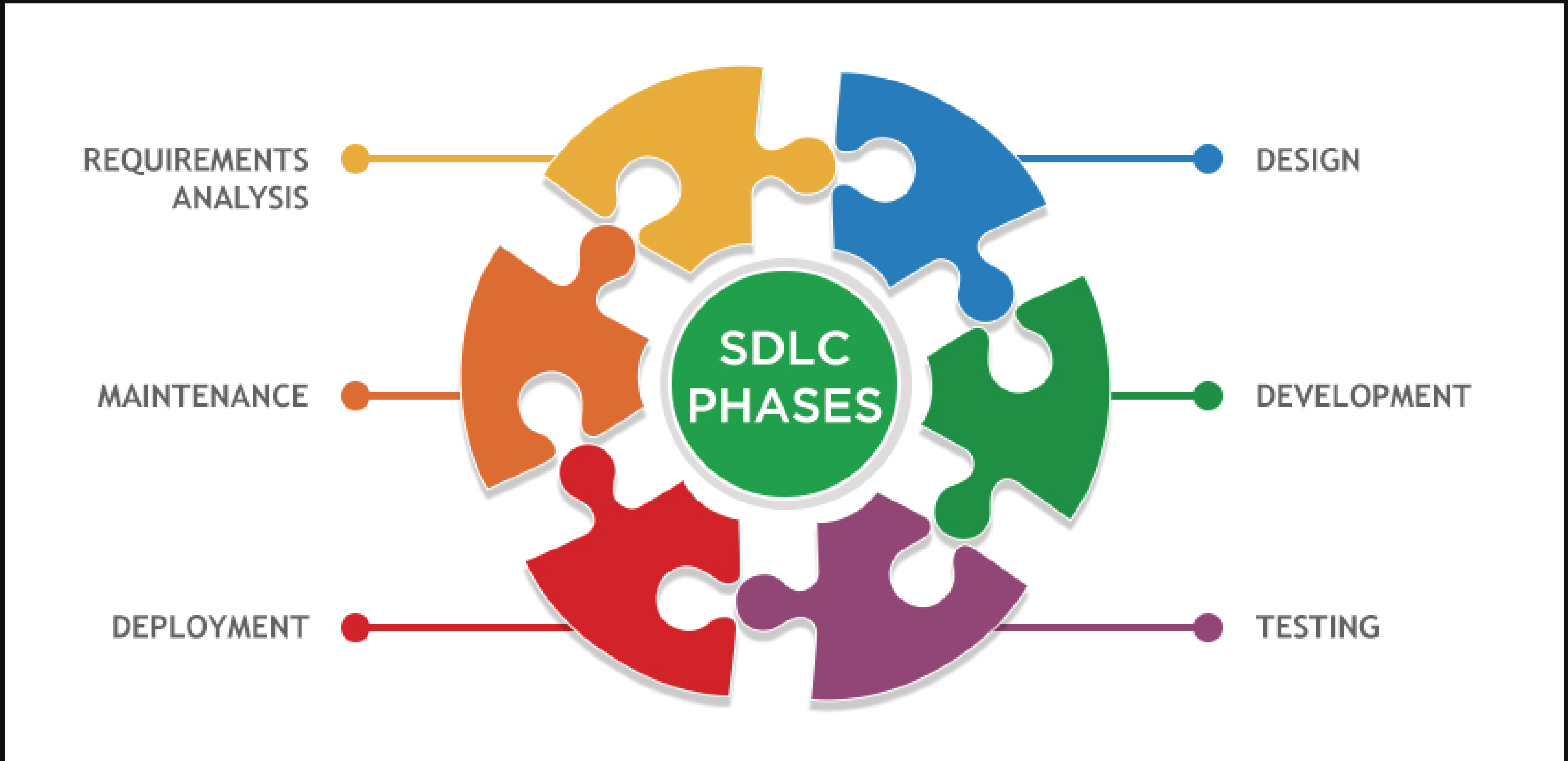
Software engineering adds small overheads through the software development process to provide higher assurances overall.

Bad things can happen

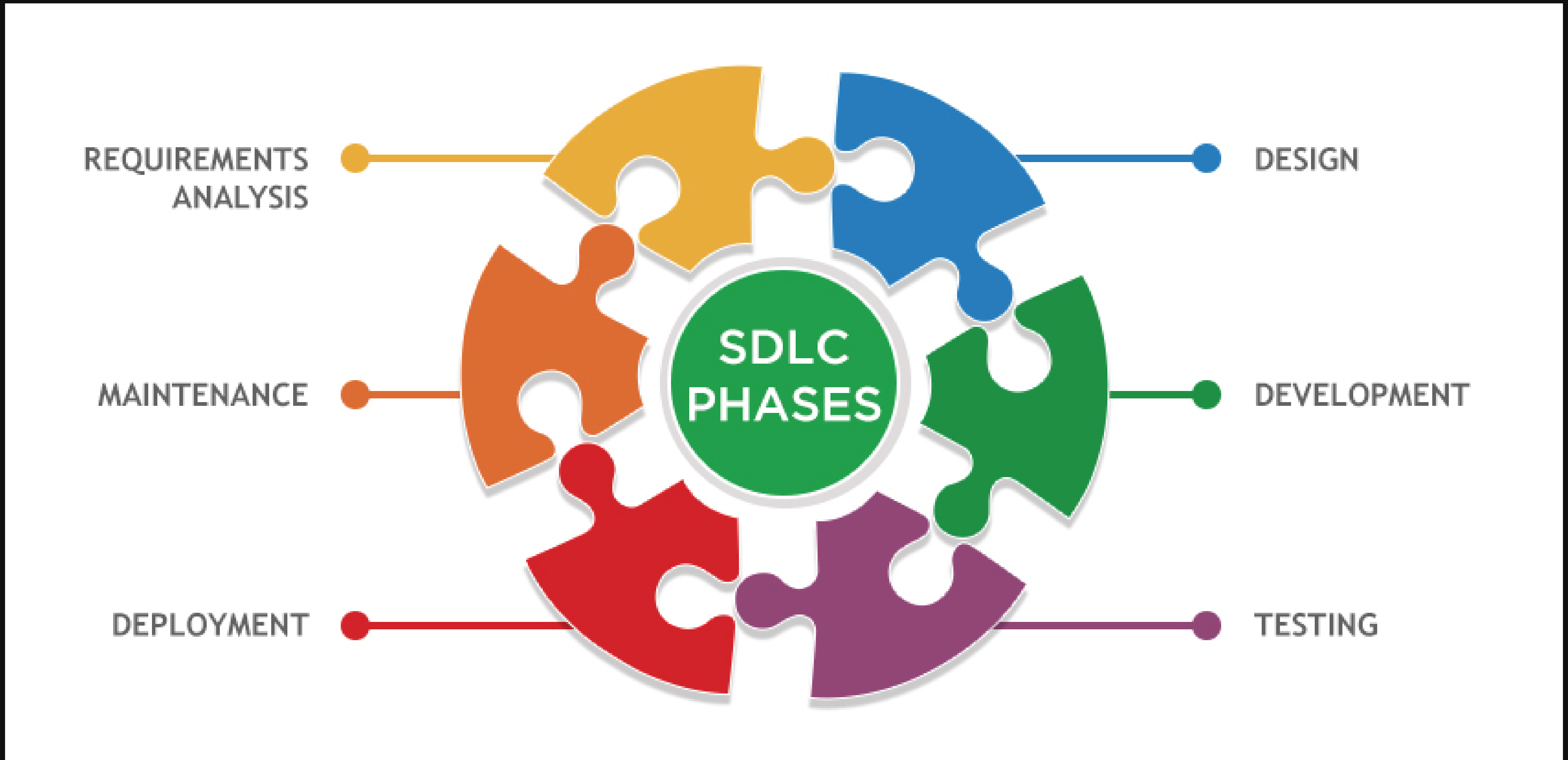
That was "What" - but "Why"?


How the customer explained it.
How the Project Manager Understood it.
How the Engineer Designed it.
How the Technician Built it.
How the Customer really wanted it.

What's the pipeline?



Software Development Life Cycle (SDLC)



SDLC

1. Requirements Analysis

- Analyse and understand problem domain
- Determine functional and non-functional components
- Generate userstories / use cases

SDLC

2. Design

- Producing software architecture/blue-prints
- System diagrams and schematics
- Modelling of data flows

SDLC

3.Development

- Choose a programming language and write the code

4. Testing

- Use unit or behaviour tests to test your software

SDLC

5. Deployment

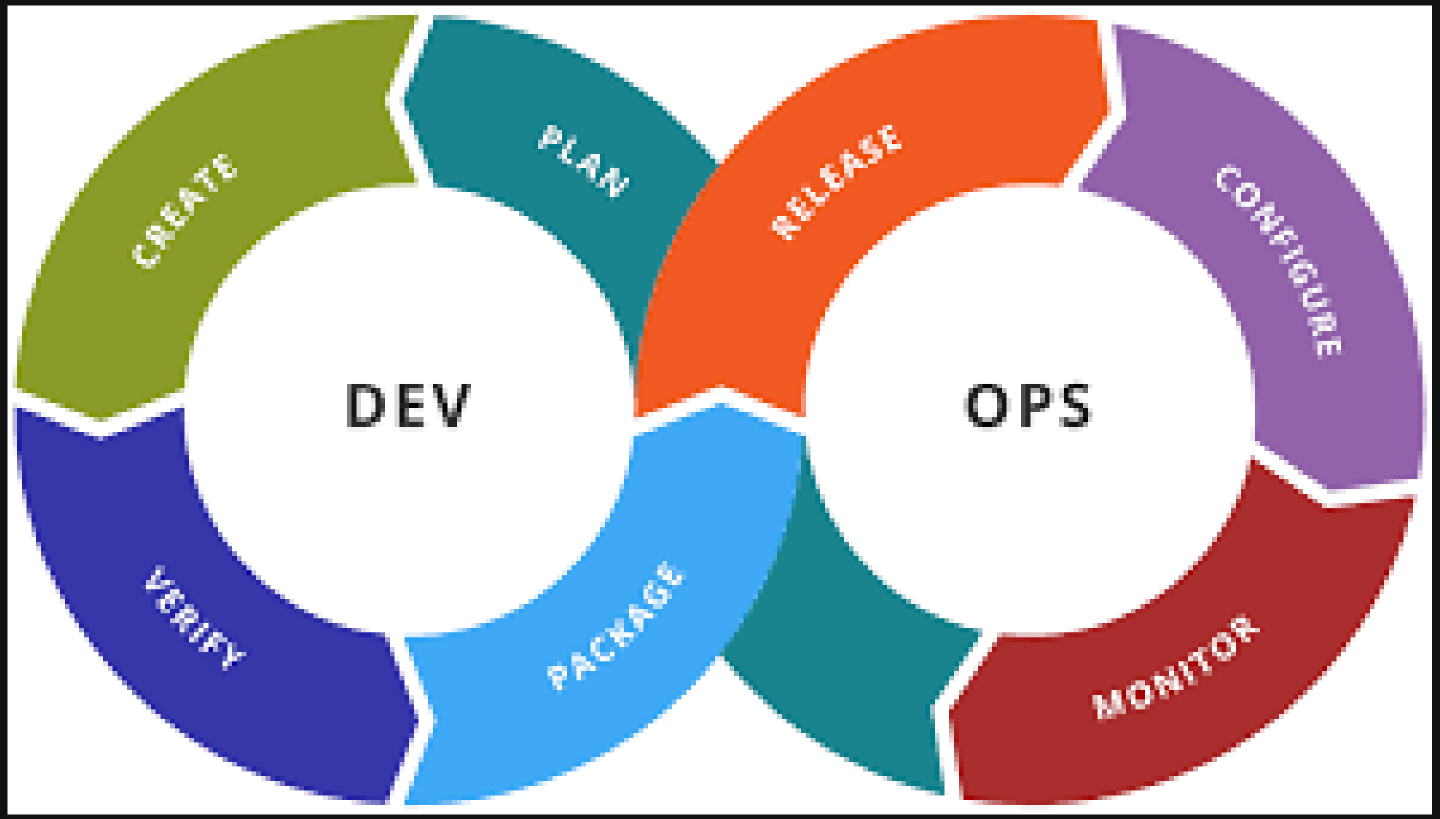
- "Ship it"

6. Maintenance

- Monitor

DevOps

- Modern philosophy of blending the development and operations teams
- Historically was just a merger of "System IT" and "Developers"



Exercise

- Let's think about how we could apply the SDLC to a simple product.