COMP1531

1.2 - SDLC - Intro

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

That was "What" - but "Why"?

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

That was "What" - but "Why"?

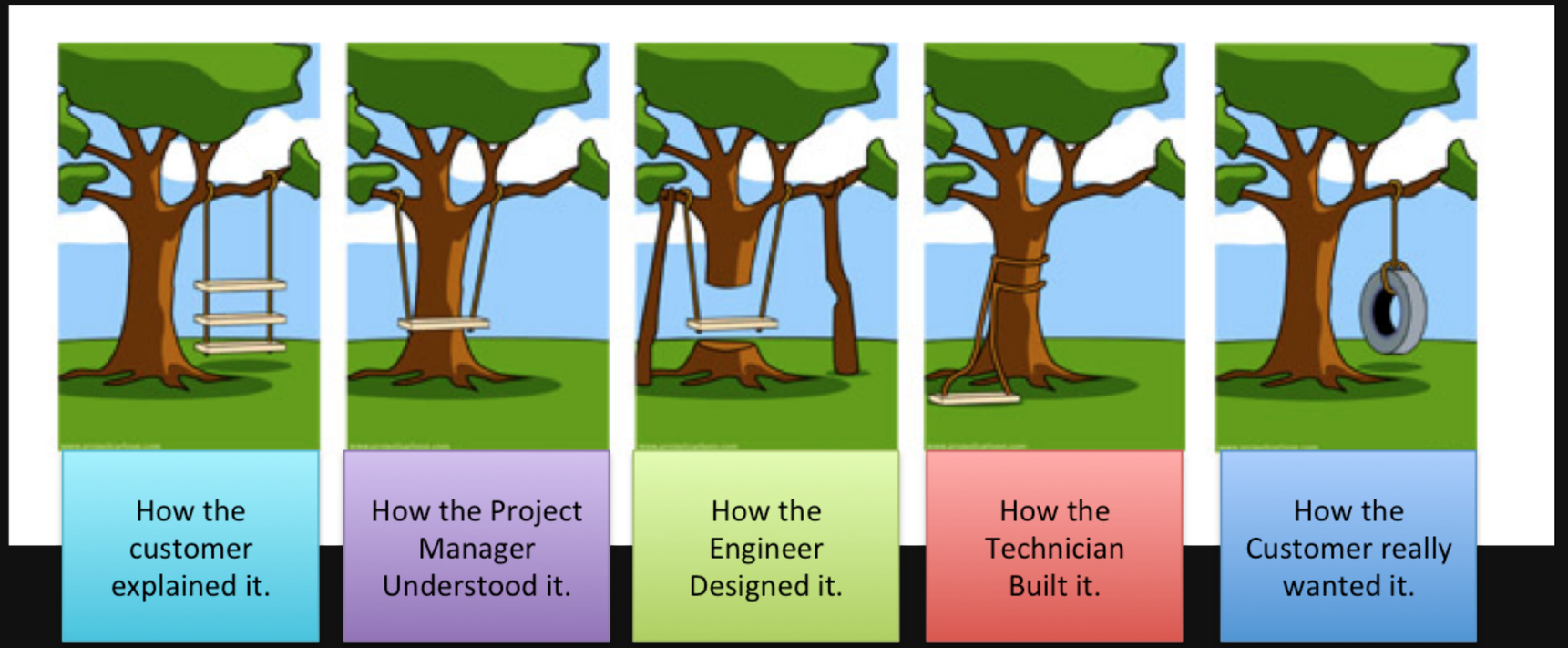
Software engineering fundamentally exists to allow **businesses** and **organisations** to de-risk their business goals compared to just hacking away.

- 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"?



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
- Automating testing for every code change

SDLC

5. Deployment

- Make the software available for use by the users

6. Maintenance

- Monitor the system, track issues, interview users to find more requirements

DevOps

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