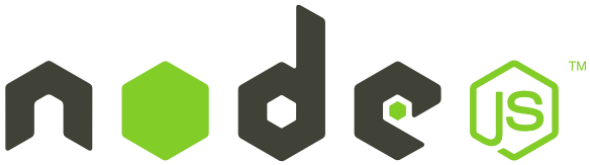
SOFTWARE DEVELOPMENT PROCESS – CT 216 SOFTWARE ENGINEERING I

Dr. Enda Barrett

OÉ Gaillimh
NUI Galway

Group Projects



- Web based real time application i.e. You will build a **web application**
  - frontend (**HTML, CSS, JavaScript**)
  - backend in **Node.js\Express** and a data storage component i.e. **MongoDB**

- No basic knowledge assumed

Group project dates

- Please update your project groupings by **Thursday 21st September at 17:00**
- Email me the following
  - Team/Group name ("The coders"),
  - Names of each member,
  - Come up with an idea and mail it to me
    - real time event app
    - instant messenger
    - social media tool

Software requirements specification

- Each group will be required to complete a Software Requirements Specification (SRS)
- This will set out what you intend to achieve and the design of your project
- A sample will be provided
- Clear division of labour (who is doing what)
- How can we break it down?
- If you opt out you will get 0!

Web applications

- Clear separation of concerns
  - Frontend view code or UI (CSS, HTML)
    - Look and feel, structuring content
  - Frontend dynamic content (JavaScript, Angular.js, Blaze, React)
    - GET/POST methods, handling/updating data
  - Backend server side code (Meteor.js, Node.js)
    - Returning data, developing APIs
  - DB component (MongoDB)
    - Schemas, queries for document retrieval

Project deliverables

- 40% in total for the project
  - 10% SRS (group)
  - 5% MCQ exam on the project technologies (individual)
  - 25% for the project (group)

- Last year the students opted to change some of the percentage allocated for certain assignments…

Today's learning outcomes (1st hour)

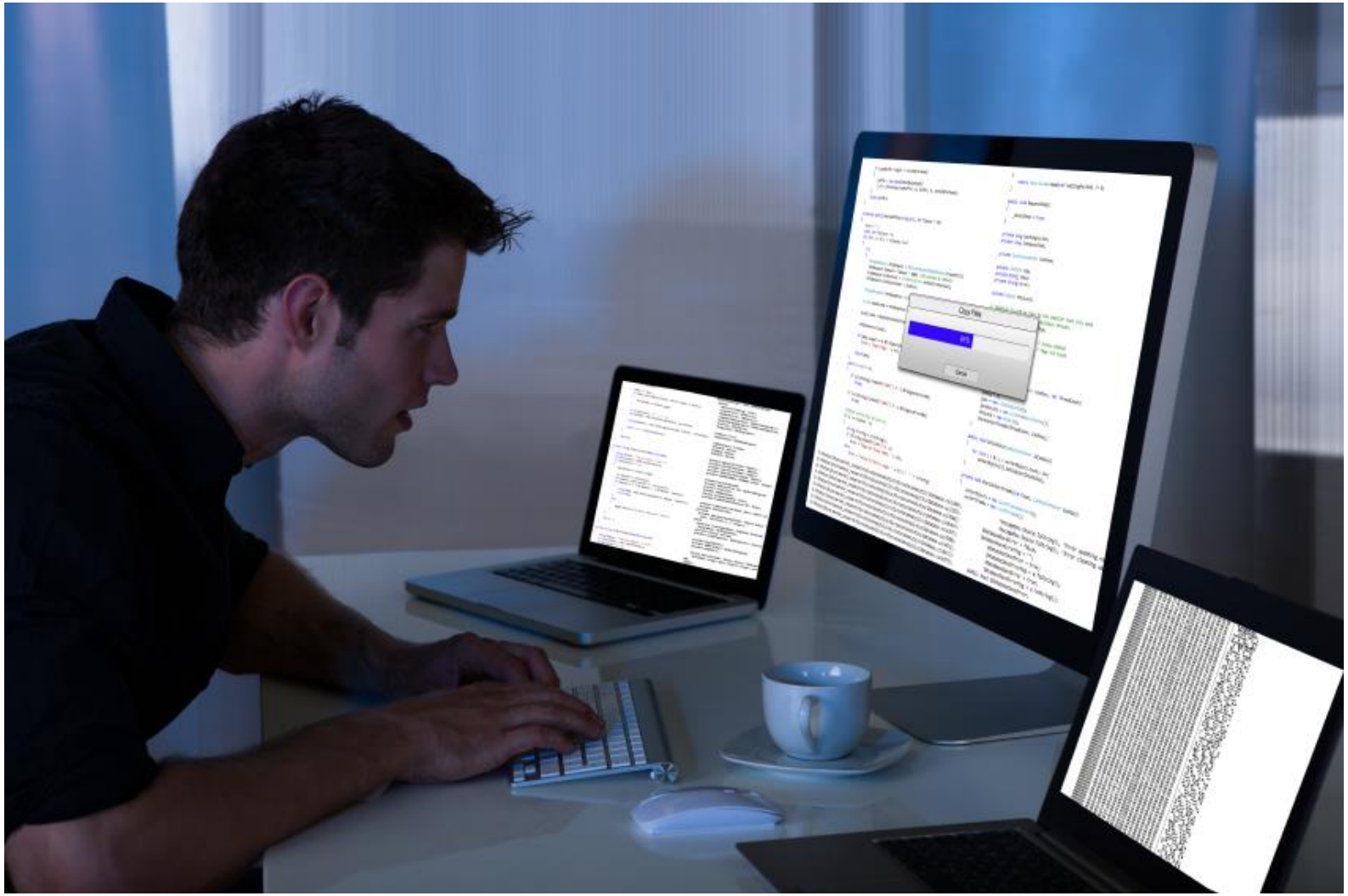
- Software processes
  - Introduce software process models
- Describe a generic process model
- What makes software engineering difficult?
- Myths around software engineering

Black holes stand at the very edge of scientific theory. Most scientists believe they exist, although many of of their theories break down under the extreme conditions within. But Professor Cornelius Van Bockstein of of the University of Ushuaia says he knows what you would find inside, and challenges the traditional idea that gravity would cause you death by "spaghettification".

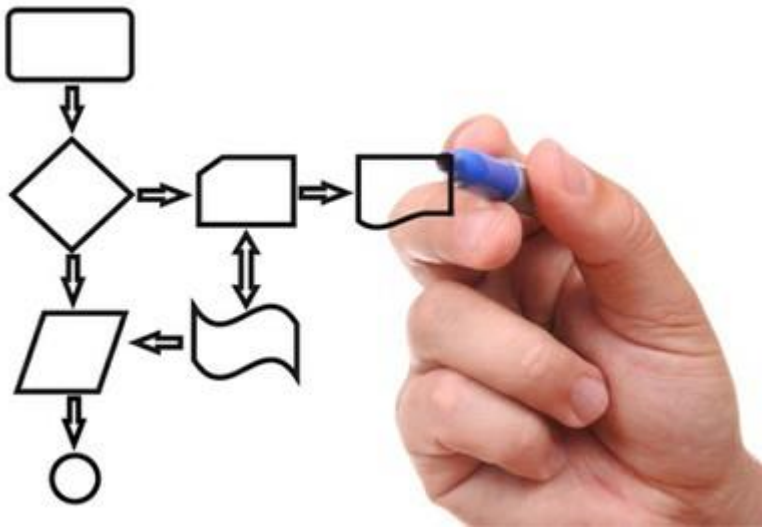
Count the F's in the text above
(Count them only ONCE!)

https://pollev.com/endabarrett504

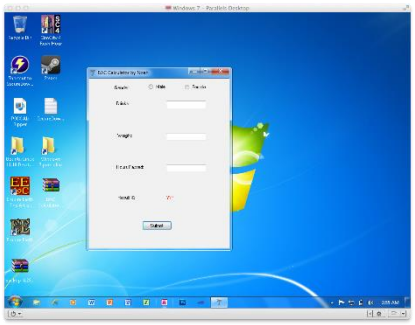
Who is like this?



Is there anyone like this?

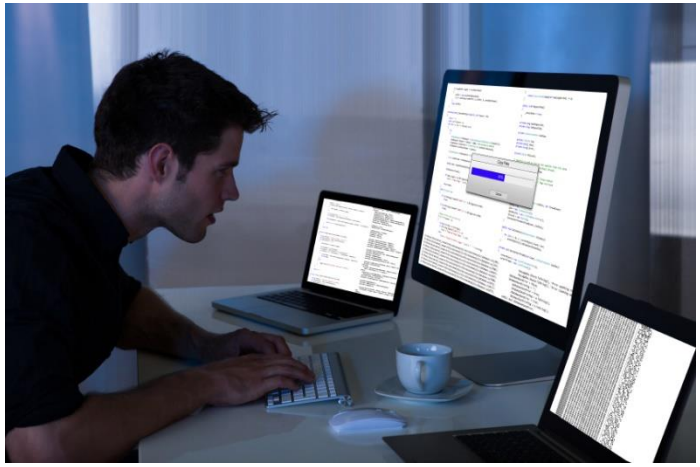


Recap: Software Dev. is complex and varied

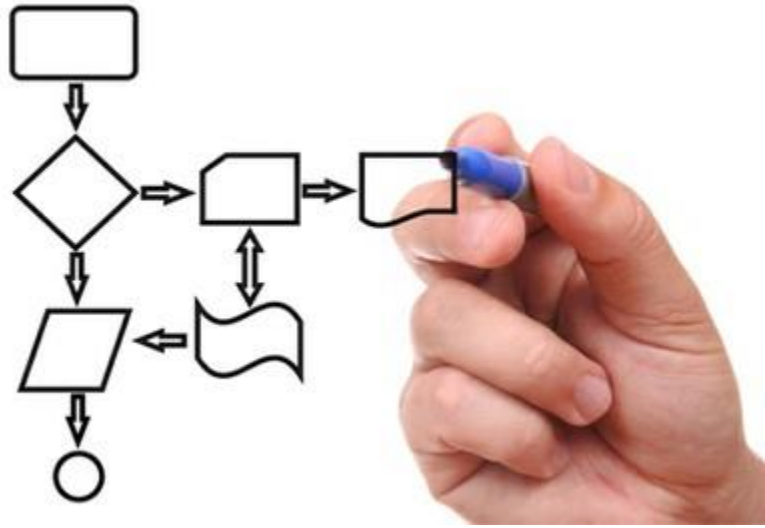


- Ada
- 3 levels of redundancy
- Different dev teams

Difference between these two?



vs



Bad Process

Bad Engineer

Good Process

Good Engineer

Building a house

- Plan
  - Sketch the layout/structure
  - Determine how the components will fit

- Construction
  - Laying foundations/block laying/engineer testing

- Deployment
  - Delivered to the customer who provides feedbac
    list

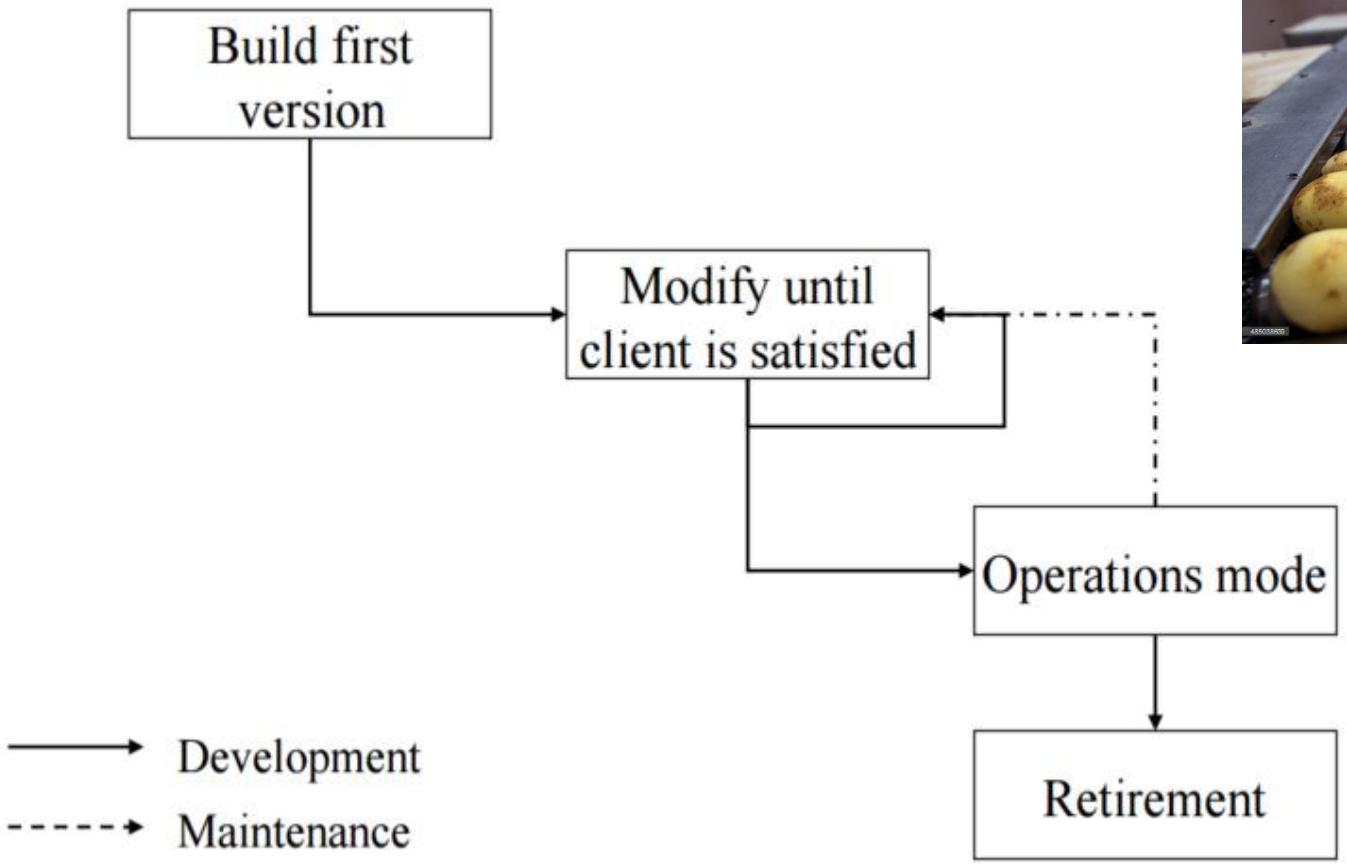
The software process

- A structured set of activities required to develop a software system
- Four fundamental process activities
  - Specification
  - Development
  - Validation
  - Evolution
- The foundation of software engineering is in the **process**
- **Goal:** To efficiently and predictably deliver a product that meets the requirements

Motivating case

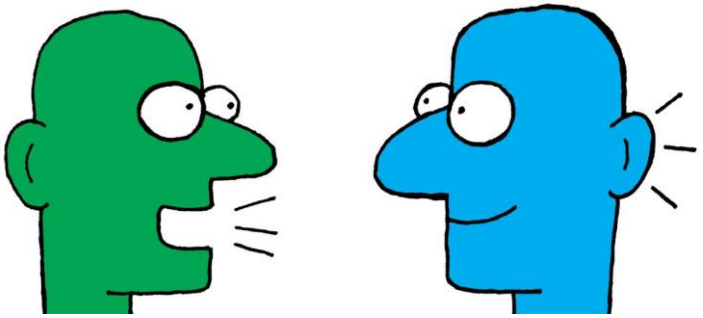
- You've been hired by a local independent retailer to build their potato peeling system

Build and fix model…worst approach



Software Process Model

- 1) Software Specification
  - Talk to the customer
  - Understand the problems
  - Talk to any relevant stakeholders



- 2) Software Development
  - Map out the tasks
  - Design the software
  - Develop the solution

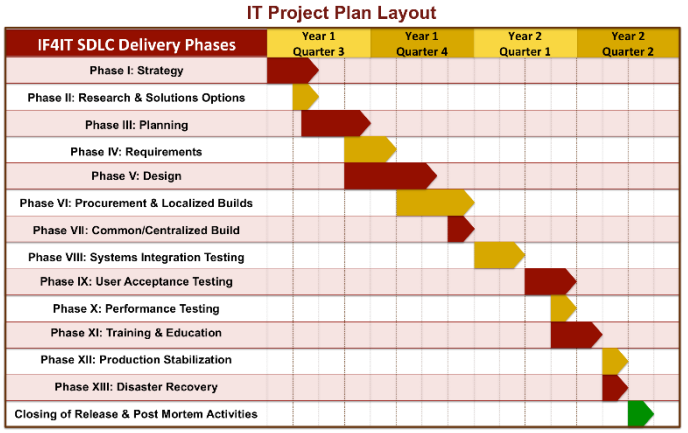


Figure: Example Layout of SDLC Phases as Key IT Project Milestones

Software Process Model…

- 3) Software validation
  - Does it meet requirements
  - Is it what the customer wanted

- 4) Software evolution (maintenance)
  - Modified to adapt
  - Changes in requirements
  - Customer & Market conditions

Model from start to finish

Examples of process models

- A workflow model
  - Shows the sequence of activities in the process including inputs, outputs and dependencies
- A dataflow or activity model
  - Represents the process as a set of activities where each carries out some data transformation
- A role/action model
  - People based: represents the roles and activities of people involved in the process.

Paradigms of software development

- Most software process models are of the following form
  - The waterfall approach
    - Separate process phases, requirements, design, implementation etc
  - Iterative development
    - Initial system developed, then customer redefines and iteratively the final system is built
  - Component Based Software Engineering (CBSE)
    - Parts of the system already built are integrated

Software Engineering Practice

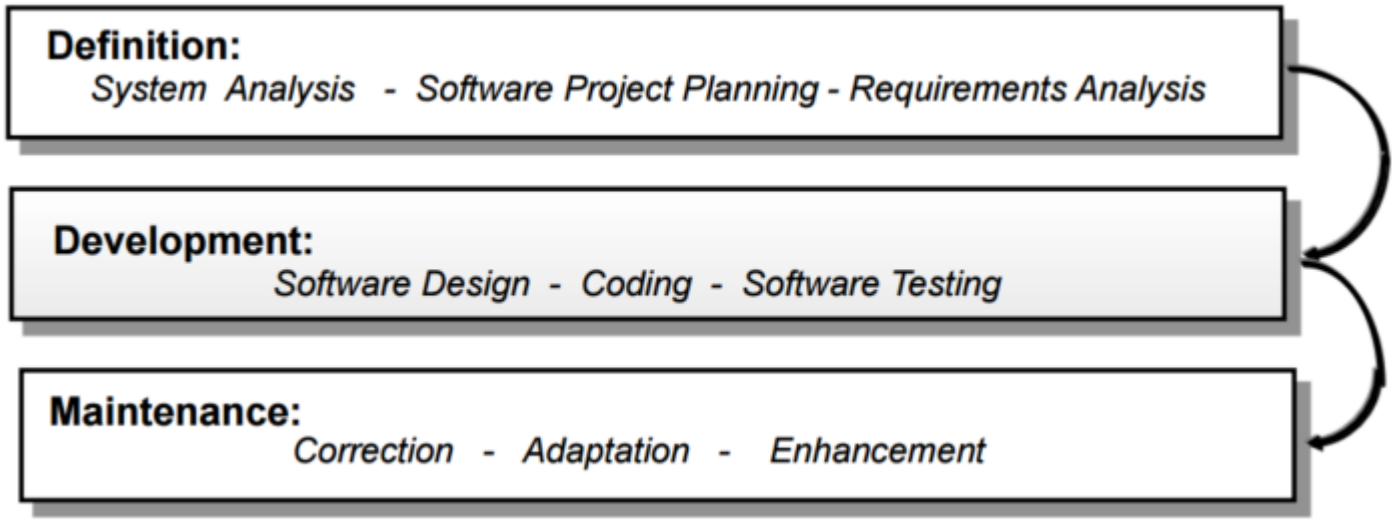
- 1) **Understand the problem** (*Communication and analysis*)
  - Who are the stakeholders?
  - What are the unknowns?
- 2) **Plan the solution** (*Modelling and software design*)
  - Have we seen this problem before?
  - Has a similar problem been already solved? Plagiarism
  - Can sub problems be found?
- 3) **Carry out the plan** (*Write the code*)
  - Does the solution conform to the plan?
  - Has the code been reviewed for correctness?
- 4) **Examine the result** (*Test it*)
  - Is each component testable?
  - Does the solution produce results as defined originally?

General Software Engineering Questions

- **What** is the problem to be solved?
  - Requirements definition
- **What** are the characteristics of the software (system) used to solve the problem?
  - Analysis
- **How** will the system be realised/constructed?
  - Design
- **How** will design and construction errors be uncovered and dealt with?
  - Test
- **How** will the system be supported long-term?
  - Maintenance

Overview of Software Engineering

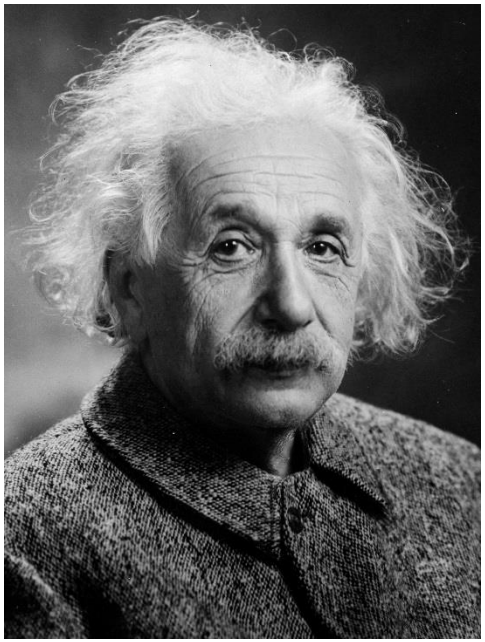
- There are three generic phases, regardless of paradigm:
  - Definition, **a focus on the *What*.**
  - Development, **a focus on the *How*.**
  - *Maintenance,* ***focuses on Change***



Software Engineering

"A clever person solves a problem
A wise person avoids it"

Albert Einstein



Why is Software Engineering difficult?

- 1) "Curse of flexibility"
  - So many solutions to a given problem
  - So flexible that one can start working with it before fully understanding what is needed to be done
  - "The untrained can get partial success"
    - They can get something "working"
    - Scaling up is hard to do "spaghetti"

Why is Software Engineering difficult?

- 2) Complexity
  - The underlying factor is intellectual manageability
    - A “simple” system has a small number of unknowns in its interactions within the system and with its environment.
    - A system becomes intellectually unmanageable when the level of interactions reaches the point where they cannot be thoroughly
      - Planned
      - Understood
      - Anticipated
      - Guarded against

Why is Software Engineering hard…

- 3) Intangibility
  - Hard to diagnose the problem
    - In order to fix an issue you need to diagnose the problem
  - Reliant on team to provide estimates of its progress, current state and completion dates
- 4) Lack of historical usage information
  - Often team has no prior knowledge of these systems
  - Usually doing "new things"
- 5) Large discrete state spaces
  - State can grow large "Curse of dimensionality"

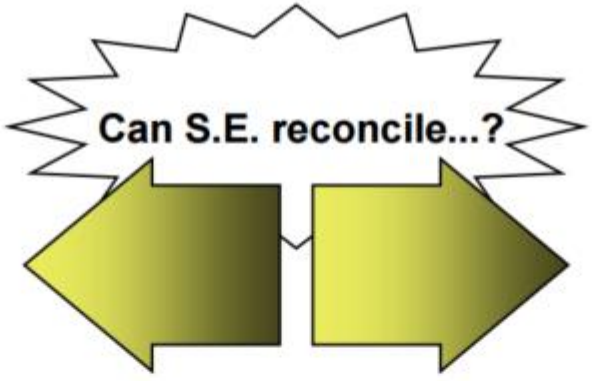
Solution(s)…?

- There is no single best approach or solution to the affliction facing software development. But the following are essential ingredients of a good Software Engineering DISCIPLINE:
  - Comprehensive, integrated methods for all phases of software development;
  - Better tools for automating these methods;
  - More powerful building blocks for implementation;
  - Techniques for quality assurance;
  - Overriding philosophy of control, coordination and management;

Software Engineering should…

- Provide a clear statement of the project mandate & objectives;
- Create effective means of communication;
- Increase user involvement & ownership;
- Provide an effective management framework to support productivity & pragmatism;
- Establish quality assurance procedures;
- Provide sound resource estimation and allocation procedures;
- Ensure the effectiveness and durability of systems produced;
- Encourage the re-usability of code and/or solutions;
- Reduce the organisation's vulnerability to the loss of software development personnel (MJ);
- Reduce and support post implementation maintenance of systems;

Software Engineering: A Solution



"The User"

- The right system
- Correct functional reqmts.
- On-time delivery
- Within budget delivery
- Project control & visibility
- Quality:
  - Usability
  - Reliability
  - Integratability
  - Durability
  - Portability

The Developer"

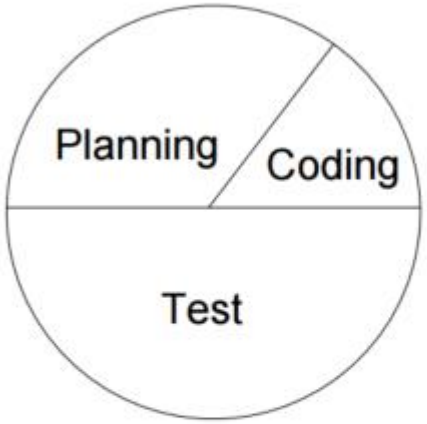
- Productivity
- Low cost
- On-time delivery
- Within budget delivery
- Project control
- Development team coordn.
- Quality:
  - Functionality
  - Usability
  - Relaibility
  - Maintainability

Prevailing myths – Pressman (1997)

- **Management myths:**
  - We have a book of Standards and Procedures available to guide development
  - State of the Art hardware for our development teams will make them more productive
  - People and time are interchangeable, if we get behind schedule, we allocate more people
- **Customer myths:**
  - A general statement of objectives defines what we require
  - Software is flexible and change is easy
- **Practitioner's myths:**
  - Once we write the program,…our job is done
  - Until I get it running…, I have no way of checking its quality
  - The only project deliverable is a working program

Myths…

Development Costs

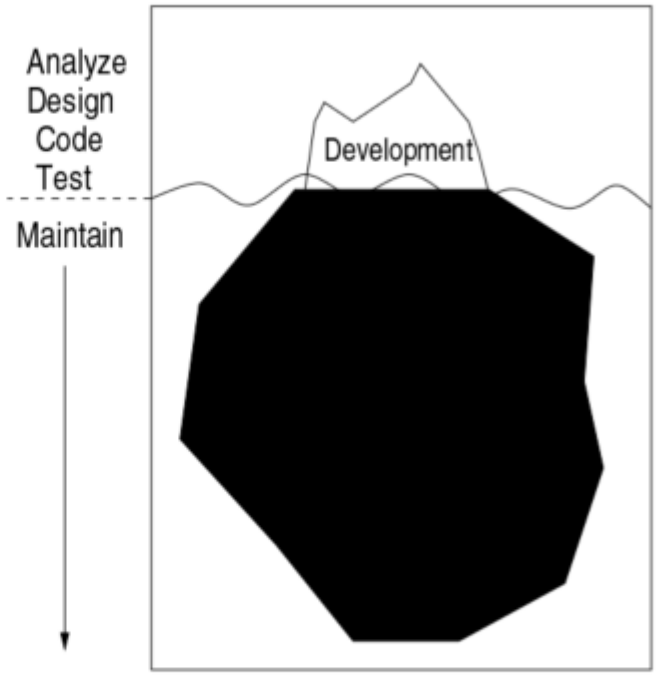


1/3 planning

1/6 coding

1/4 component test

1/4 system test



Development costs are only the tip of the iceberg.

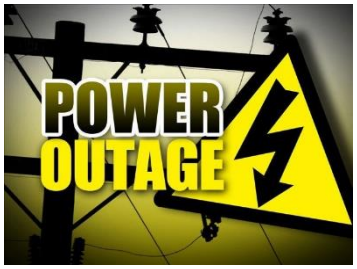
Case study – Scamall project

- For years (late 90s early 00s) APC made fortunes on securing these

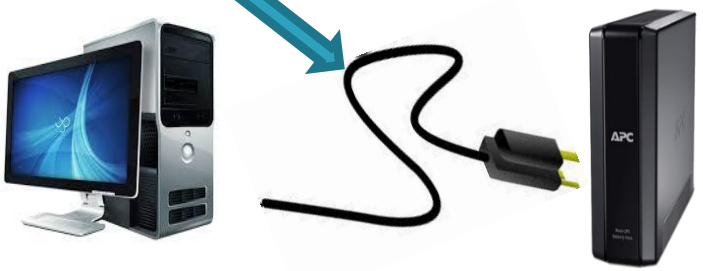
- HBN – $2.1bn 2014

How it works

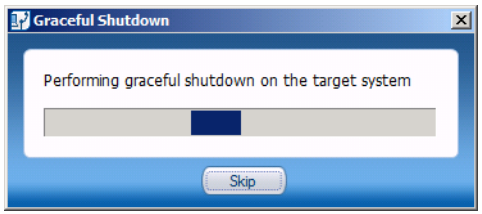
1) Storm



2) Power outage

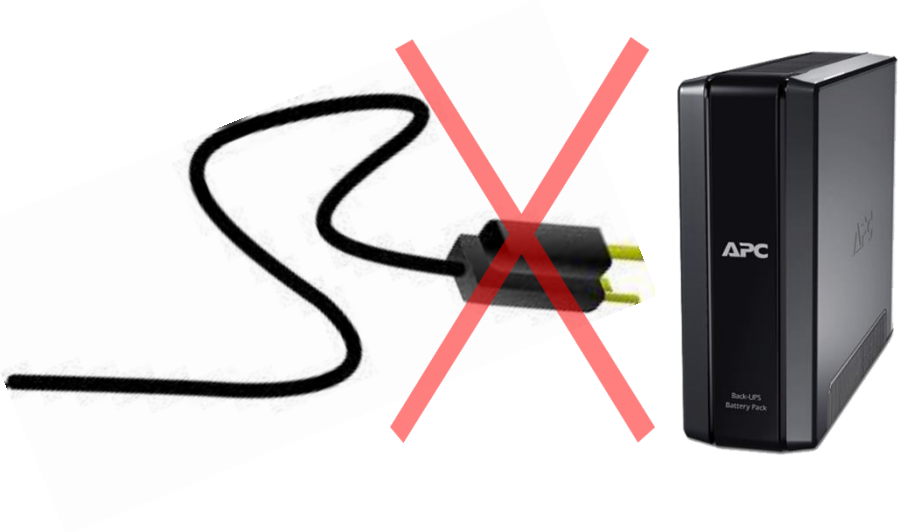


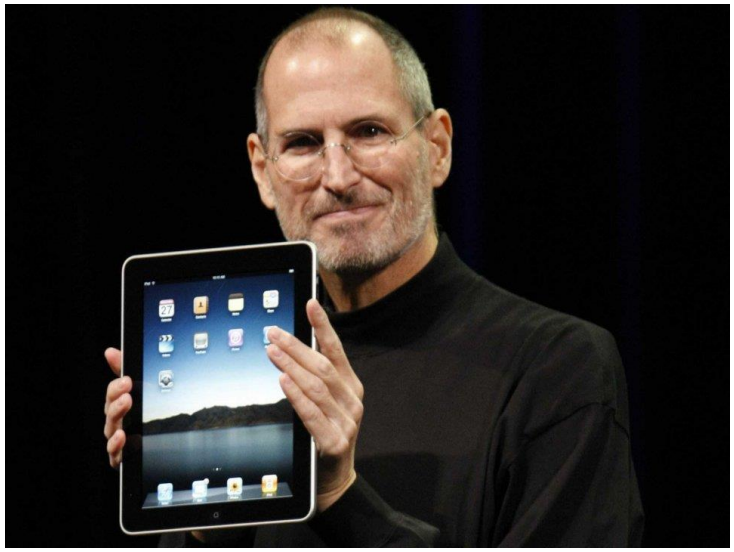
3) PC keeps running



Laptops/tablets have backup power

- However this guy came along





- Market collapsed

Refocus on securing the network

Sales were flat

A bright idea…

Cloud

Mobile

Real time

Scamall project

- Scamall project was born
  - Project started Q2 in 2012
  - Expected to last for 1 year
  - Budget was €10m
    - Real time comms with the cloud
    - 1 million devices
    - Allow remote functions
    - iOS & Android
- Stakeholders
  - Cloud team
  - Firmware team
  - Mobile app team
  - Web services team

ioBridge®
Connect things.

kinvey

What actually happened?

- Project has been shelved…
  - No first release
  - Entered Beta 4 years late
  - Costs hit €100m+
  - Delivered half of what was promised
    - Most of features in the original spec had been removed or scaled back

What went wrong?

- This project attempted to follow best practises but lacked execution
  - Project managers were used to Waterfall methods (an older pm process) and now they had to use Agile methods
  - Java engineers, now required to be Node.js engineers
  - Cloud team consisted of contract engineers and people left frequently
  - Firmware team (US), Software team (IRE), Mobile dev team (India), Cloud team (Fra + US)

What should have been done better?

- No single or quick solution
  - Whilst the project lacked execution, many things were outside of any one individual's control
  - No matter how highly trained and experienced a project manager you have, when you have diverse and fast changing requirements, it's always difficult
  - Really great architects designing the cloud solutions
- Without Software Engineering methods the project wouldn't have made as much progress as it did.

What went wrong cont…?

- Feature (scope) creep
  - More conversations with customers
- Thrashing
  - Team were Java engineers, now overnight they were switched to Node.js
- Tolerating
  - The SE cloud was designed for monitoring a few installs not 1m devices sending data every 30 seconds
- Compromising
  - Settling for sub-optimal solutions in order to move work along
- Integration problems
  - Many teams with often scarce documentation
  - Interfaces could change weekly
- Redesign and rewriting during test
  - Testing was a pain as the cloud servers kept dropping
- No documentation of design decisions

What is Software Engineering?