



1

## Overview

- Software Engineering
- What Software Developers do
- Software Development Process Models

2

# Software Engineering

CRICOS 00111D  
TOD 3059

3

## What is Engineering?

- "Application of scientific knowledge and mathematical methods to practical purposes of the design, construction or operation of structures, machines, or systems."
- "The art or science of making practical application of the knowledge of pure sciences, as physics or chemistry, as in the construction of engines, bridges, buildings, mines, ships, and chemical plants."
- "The branch of science and technology concerned with the design, building, and use of engines, machines, and structures."

4



## What is Engineering?

Creating cost-effective solutions ...

... to practical problems ...

... by applying scientific knowledge ...

... building things ...

... in the service of mankind

Engineering enables ordinary people  
to do things that formerly required virtuosos

5

## What is Engineering?

Creating cost-effective solutions ...

... to practical problems ...

... by applying *codified* knowledge ...

... building things ...

... in the service of mankind

Engineering enables ordinary people  
to do things that formerly required virtuosos

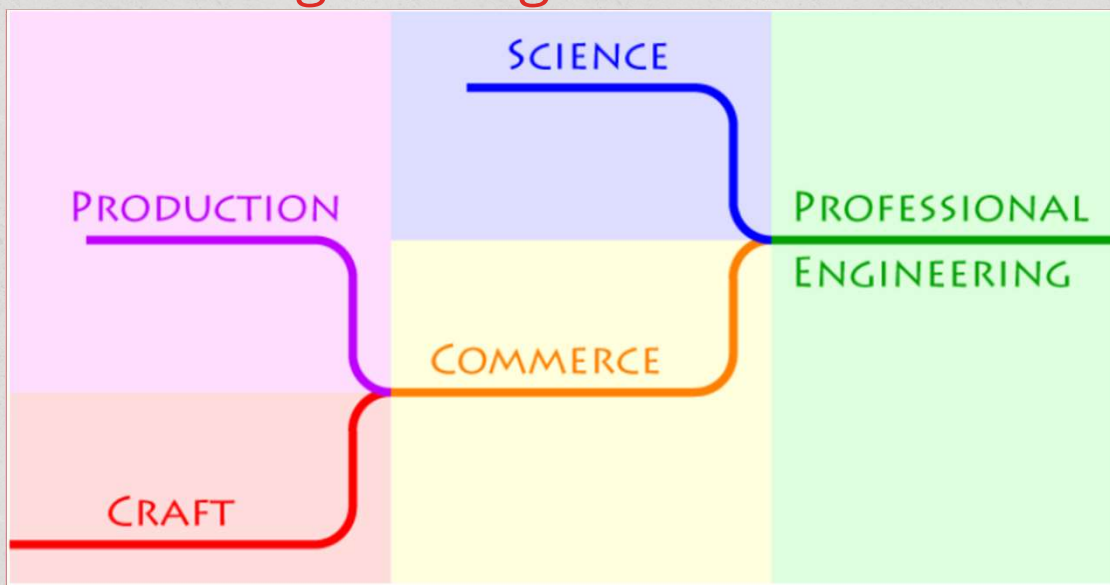
6

## What is Engineering?

- limited time, knowledge, and resources force decisions on trade-offs
- best-codified knowledge, preferentially science, shapes design decisions
- reference materials make knowledge and experience available
- analysis of design predicts properties of implementation

7

## What is Engineering?



8

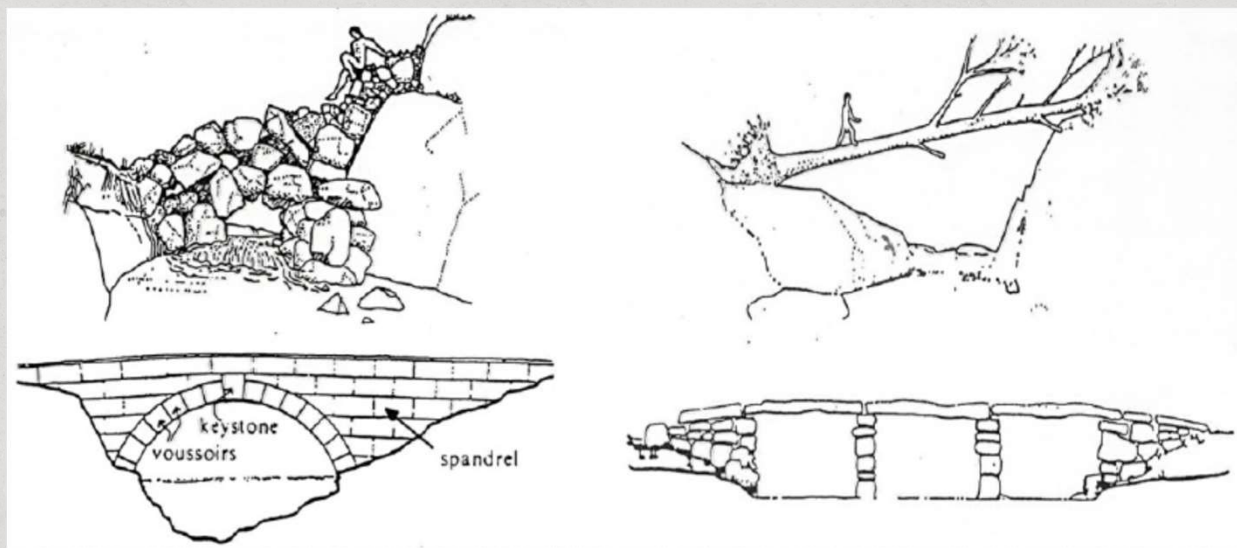
## What is Engineering?

- Model: Civil Engineering
- Example: Bridges



9

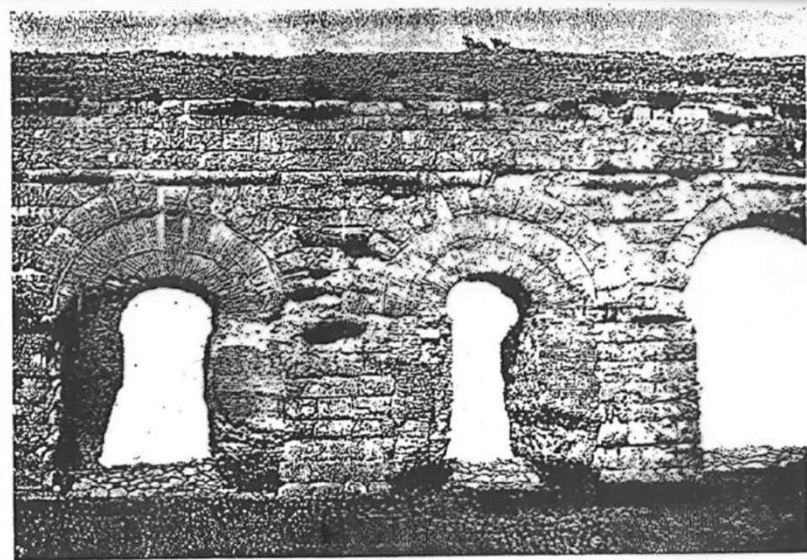
## What is Engineering?



10

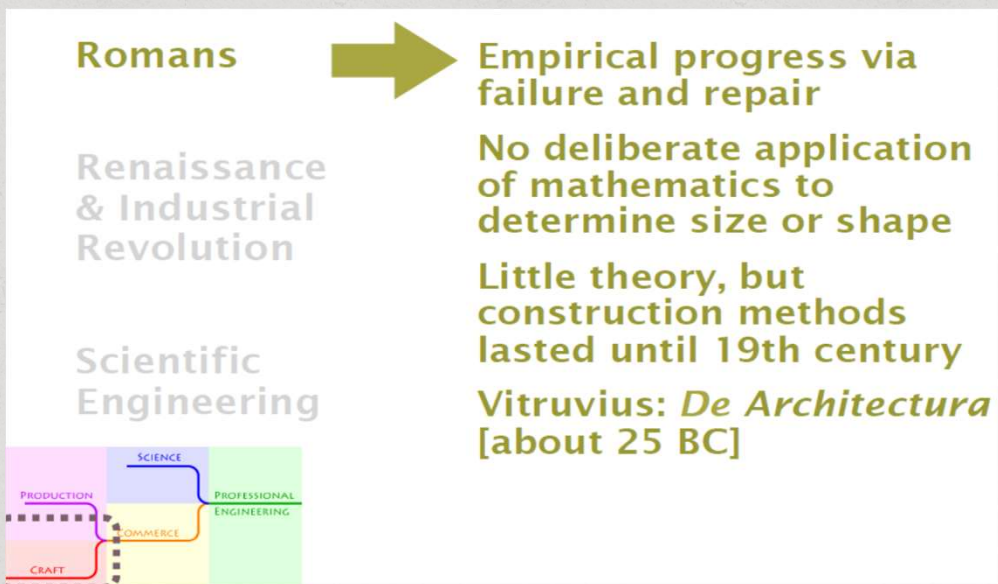


## What is Engineering?



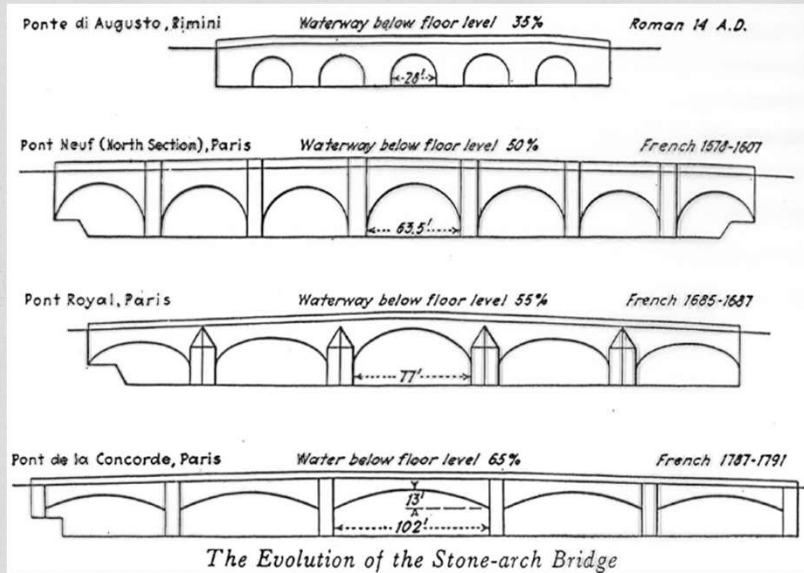
11

## What is Engineering?



12

## What is Engineering?



13

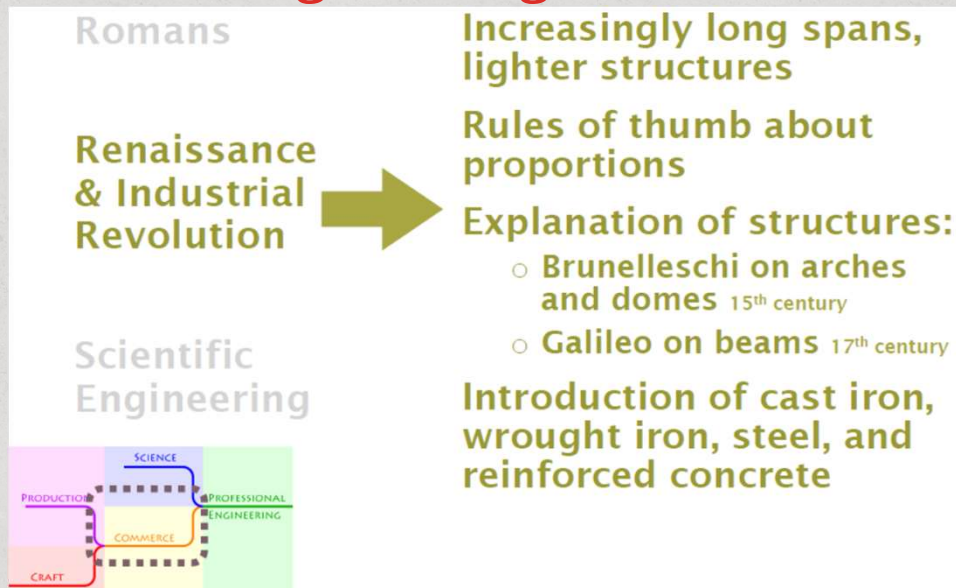
## What is Engineering?



14

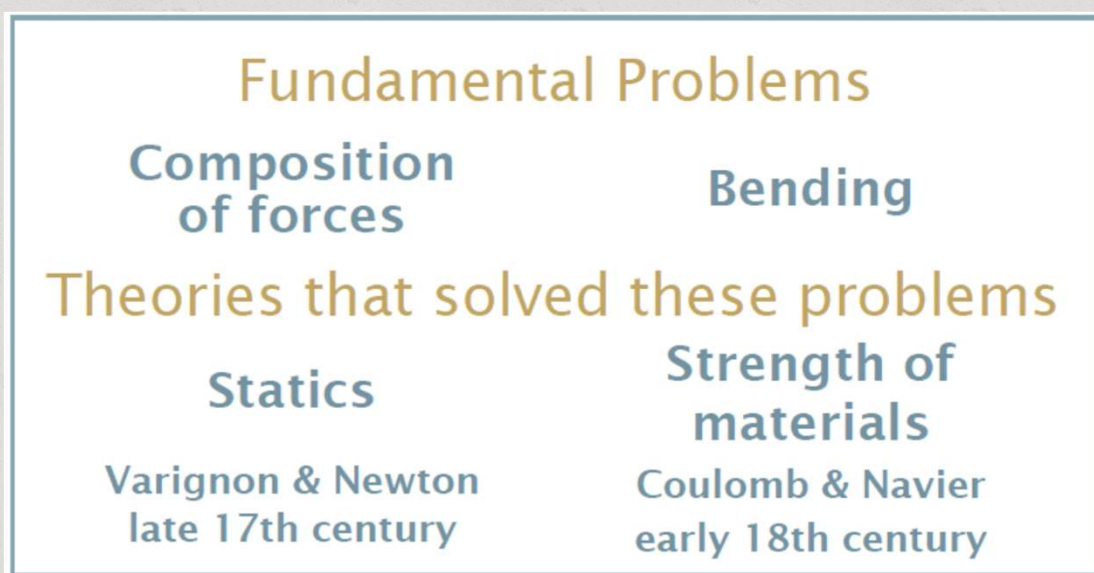


## What is Engineering?



15

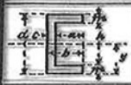




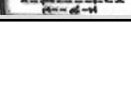
## What is Engineering?



16

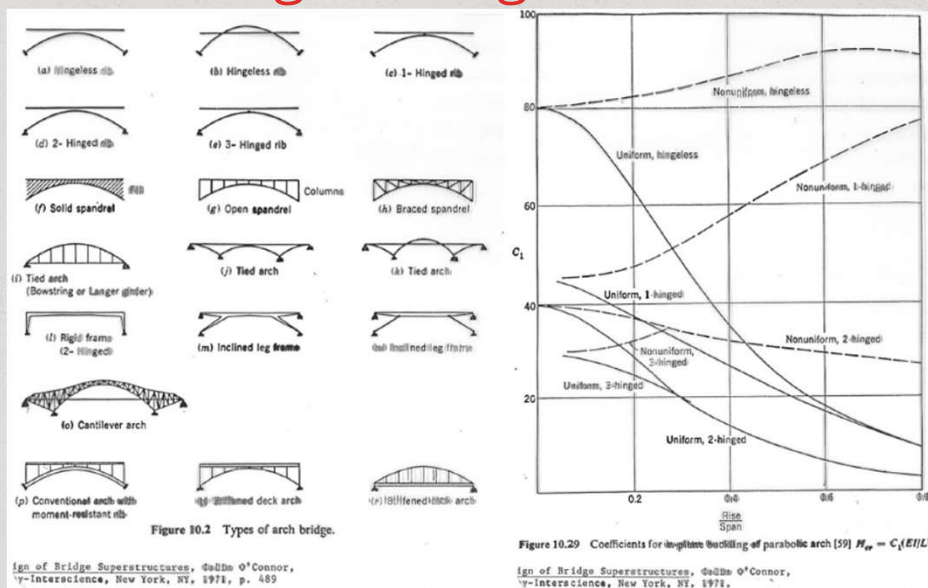


# What is Engineering?

Section	Area of Section $A$	Distance from Axis to Extremities of Section $y$ and $y_1$	Moment of Inertia $I$	Section Modulus $S = \frac{I}{y}$
	$bd - ah$	$y = \frac{d}{2}$	$\frac{1}{12} (bd^3 - ah^3)$	$\frac{bd^3 - ah^3}{6d}$
	$bd - ah$	$y = b - y_1$ $y_1 = \frac{2 bhm + ha^2}{2A}$	$\frac{1}{3} (2 mb^3 + ha^3) - Ay_1^3$	$\frac{I}{y}$
	$bd - 2 ah$	$y = \frac{d}{2}$	$\frac{1}{12} (bd^3 - 2 ah^3)$	$\frac{bd^3 - 2 ah^3}{6d}$
	$bd - 2 ah$	$y = \frac{b}{2}$	$\frac{1}{12} (2 mb^3 + ha^3)$	$\frac{2 mb^3 + ha^3}{6b}$
	$bmi + ha$	$y = d - y_1$ $y_1 = \frac{d^2e + m^2(b-e)}{2A}$	$\frac{1}{3} (ry^3 + by_1^3 - 2 a (y_1 - m)^3)$	$\frac{I}{y}$
	$bmi + ha$	$y = \frac{b}{2}$	$\frac{1}{12} (mb^3 + ha^3)$	$\frac{mb^3 + ha^3}{6b}$

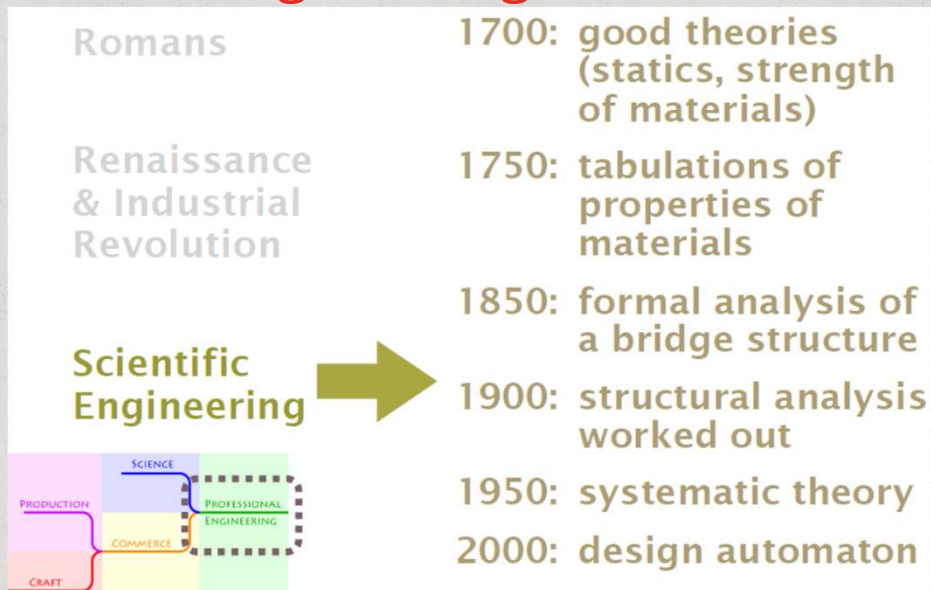
17

# What is Engineering?



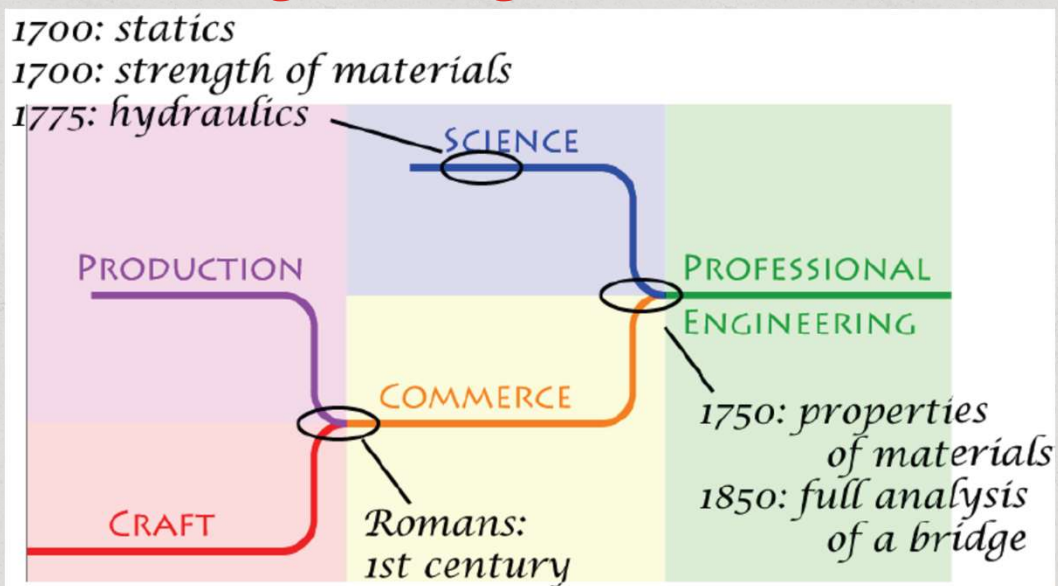
18

# What is Engineering?



19

# What is Engineering?

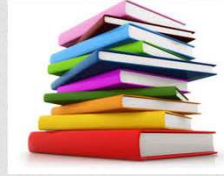


20



## What is Software?

- Generic term for organized collections of computer data and instructions.
  - Computer programs and associated documentation.



- Software products may be developed for a particular customer or may be developed for a general market.

21

## What is Software Engineering?

From the definition of Engineering:

Creating cost-effective solutions ...

... to practical problems ...

... by applying codified knowledge ...

... building things ...

... in the service of mankind

22

## What is Software Engineering?

From the definition of Engineering:

*The branch of computer science that*

creates cost-effective solutions ...

... to practical *computing* problems ...

... by applying codified knowledge ...

... *developing software systems* ...

... in the service of mankind

23

## What is Software Engineering?

- Software is design-intensive -- manufacturing costs are minor
- Software is symbolic, abstract, and constrained more by intellectual complexity than by fundamental physical laws

24

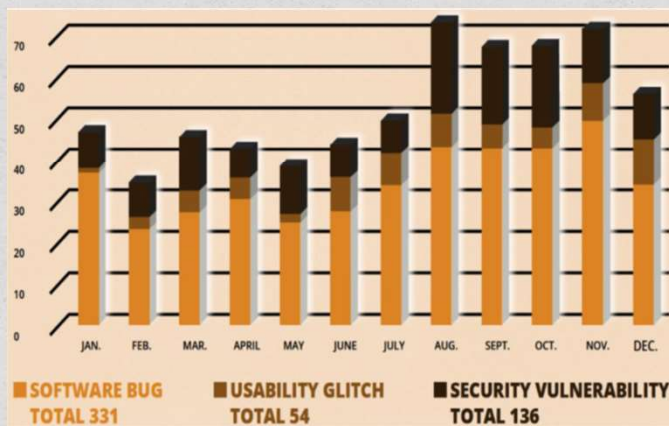


## Why Software Fails?

- No functioning software system results.
- The resulting software does not sufficiently address the need of the users.
- Software system contains incorrect computations.
- The software system is too difficult to use correctly.
- The system response time is too slow to be used without disappointment.
- Users requirements change rapidly.

25

## Software Engineering



- In 2017 alone, software failures caused the loss of over 1.7 trillion US dollars in 2017 alone
- Affected the lives of about 3.7 billion people (half of the world population)

Source: Tricentis' "Software Fail Watch"

26

## Important Attributes of Good Software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

27

## Software Engineering in a Nutshell

Very roughly

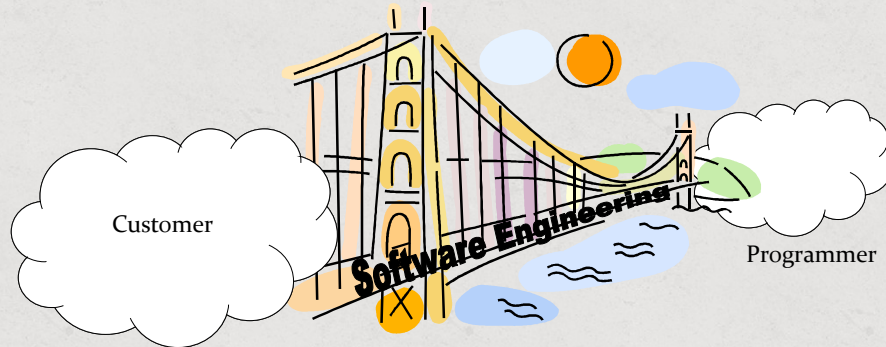
Software engineering is concerned with the **efficient** and **timely delivery** of software that meets **stated requirements**.

28



# Role of Software Engineering

A bridge from customer needs to programming implementation

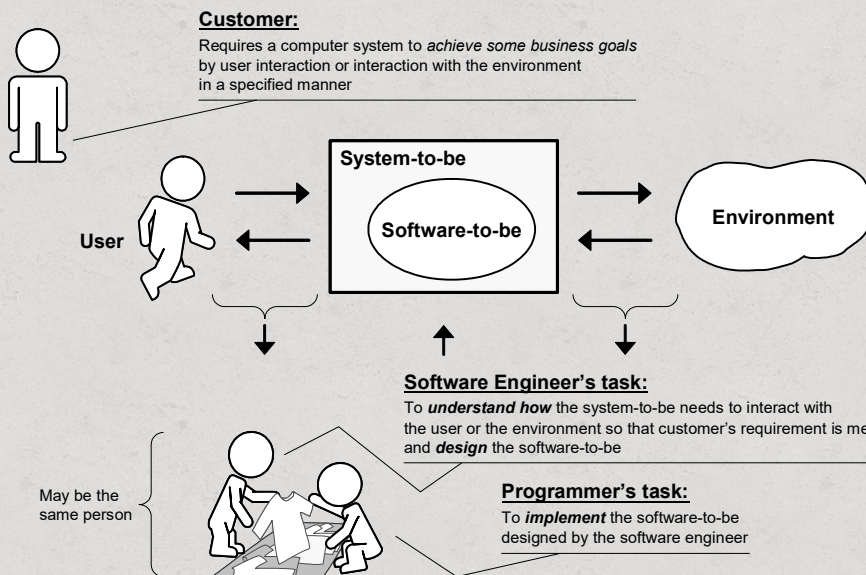


## First law of software engineering

Software engineer is willing to learn the problem domain  
(problem cannot be solved without understanding it first)

29

# Role of Software Engineering



30

# Software Developers

CRICOS 00111D  
TOD 3059

31

## What Software Developers Do....

- Writing software code
- Testing software + Documenting software
- Writing software code + Documentation + Deploying software
- All of the above + MORE

32



# What Software Developers Do....

(Source: [US Bureau of Labor Statistics](#))

- Software developers typically do the following:
  - **Analyze** users' needs and then **design**, **test**, and **develop** software to meet those needs
  - **Recommend** software upgrades for customers' existing programs and systems
  - **Design** each piece of an application or system and **plan** how the pieces will work together
  - **Create** a variety of models and diagrams (such as flowcharts) that instruct programmers how to write software code
  - **Ensure** that a program continues to function normally through software **maintenance** and **testing**
  - **Document** every aspect of an application or a system as a reference for future maintenance and upgrades
  - **Collaborate** with other computer specialists to create optimum software
- **Software developers are in charge of the entire development process for a software program**

33

# What is Software Development Process?

- A set of **disciplined activities**, subject to constraints and resources, **to build a software system** with pre-defined purposes from **beginning to completion**
- AKA "Software lifecycles"
- Objectives – To understand the concept of software development process and the context in which software is developed
  - Different approaches,
  - Different software process models

34

# Software Development Lifecycle

CRICOS 00111D  
TOD 3059

35

## Software Process Activities

- Software specification
- Software development
- Software validation
- Software evolution

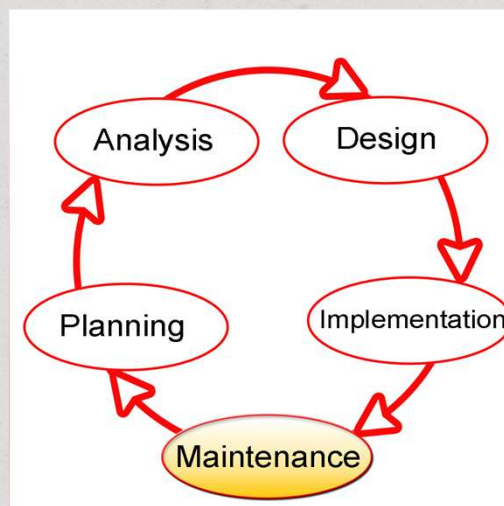
36

## Software Development Stages

- Requirements Analysis & Specification
- Conceptual/System/Architectural Design
- Detailed/Program Design
- Implementation/Coding
- Unit & Integration Testing
- System Testing/Validation
- System Delivery/Deployment
- Maintenance

37

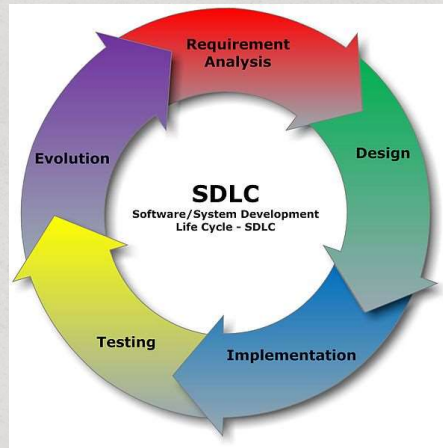
## System Development Lifecycle (SDLC)



38

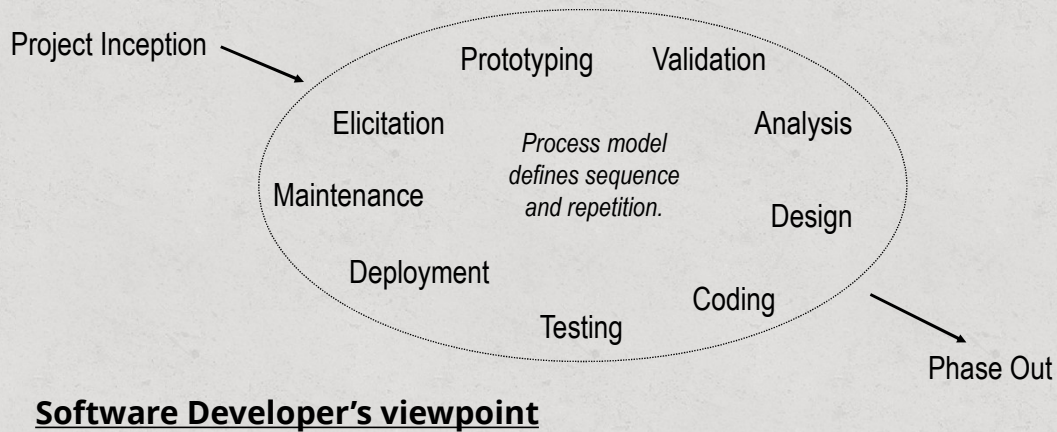


## SDLC – A More Software-Oriented View



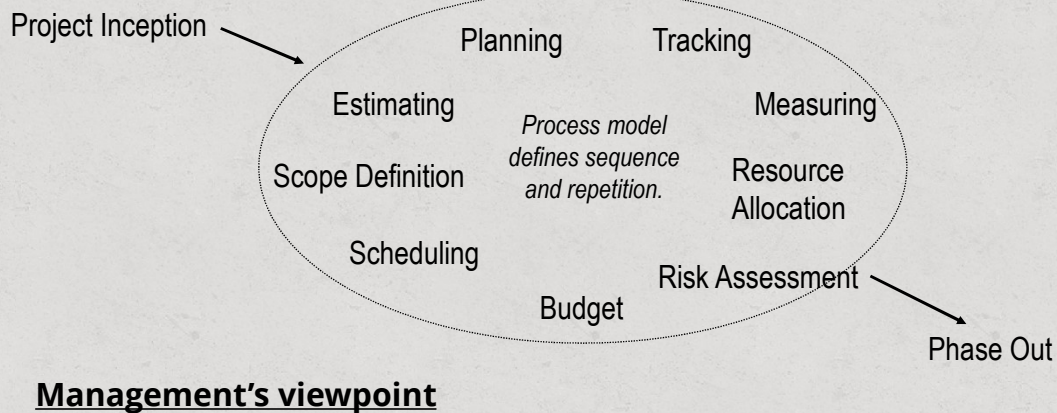
39

## What is Software Development?



40

# What is Software Development?



41

## Modelling the Software Process

- A software process model is a representation (abstraction) of the software development process
- The main purpose of the abstraction is to explain the different approaches to software development
- The model captures the primary activities that we undertake in software development

No single software process model is suitable for all different types of software

42

## Single vs Multiple Software Process Models

- For small systems, it is OK to have a single software process model
- For large systems, project teams are required to apply different process models on different components of the systems
  - Example: An internet banking application to handle day-to-day banking transactions

43

## Plan-Driven vs. Agile Processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.

44



## Plan-Driven vs. Agile Processes

- The waterfall model
  - Plan-driven model.
  - Separate and distinct phases of specification & development.
- Incremental development
  - Specification, development and validation are interleaved.
  - May be plan-driven or agile.

45

## Software Development Process Models

- **Big Design Up Front (BDUF)**
  - **Waterfall**
  - **V-model**
- **Spiral**
  - Iterative process
  - Unified process
- **Agile**
- **Formal Methods**

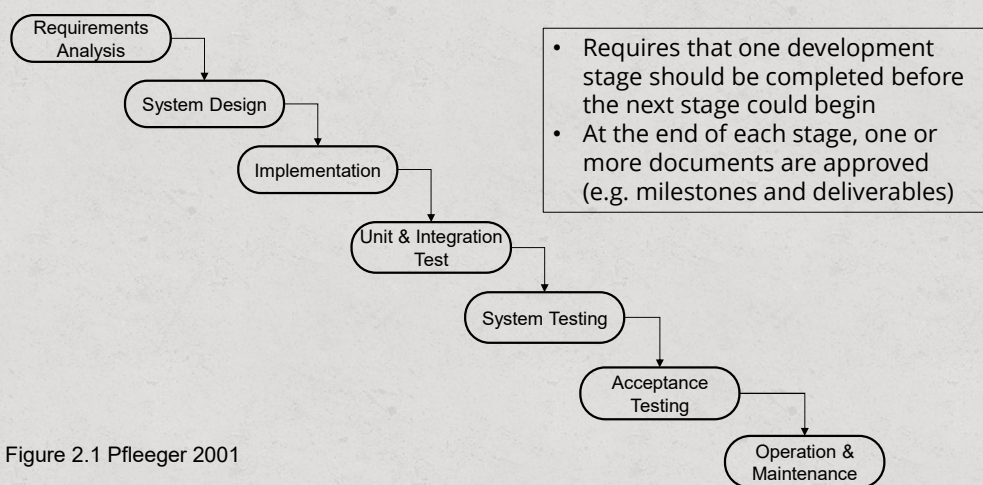
46

# The Waterfall Model

- Big Design Up Front (BDUF)
  - Benington 1956, Royce 1970, et al
- Emphasis on getting **design** absolutely right before progressing to **development**
- Waterfall applies this to all phases: each must be finalized before moving to the next
- Waterfall Stages:
  - Requirements
  - Design
  - Implementation
  - Verification
  - Maintenance

47

# The Waterfall Model



Adapted from Figure 2.1 Pfleeger 2001

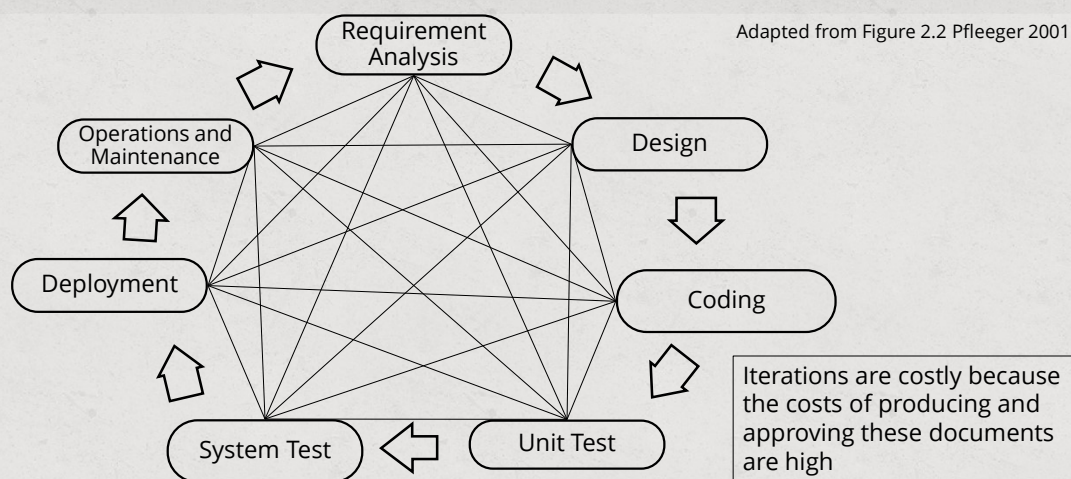
48

## The Waterfall Model – Too Simplistic?

- This model has problems!
- Once a stage or phase has been “completed”, it can not be revisited
  - Requirements need to be frozen, design cannot be altered
- Not true in practice
- Need to have many iterations
  - Requirements may not have been understood, customers may have changed their minds
- The actual development process may look more like this...

49

## The Waterfall Model in Practice?

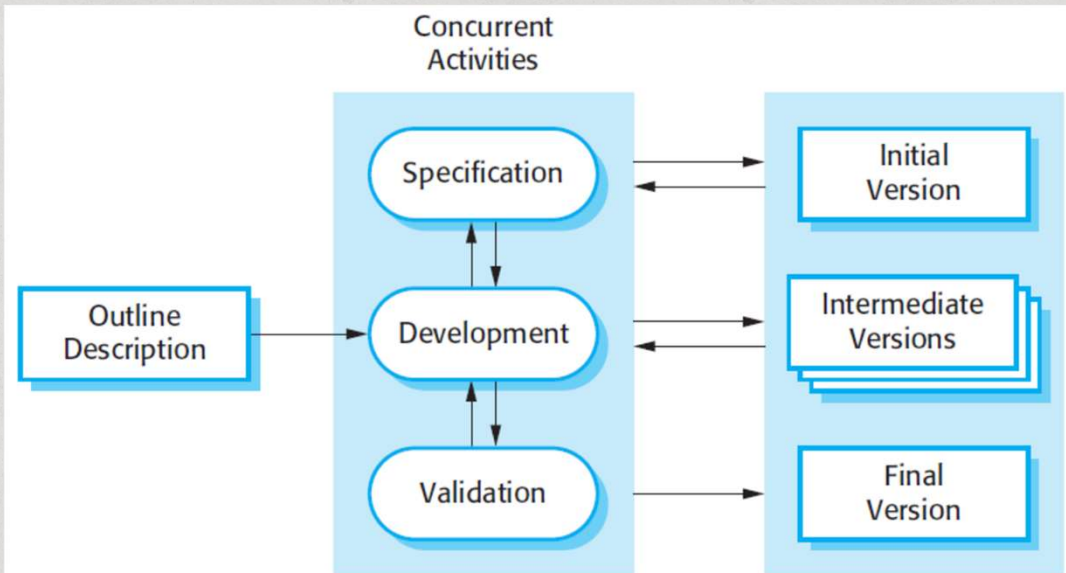


50





## Incremental Development



53

## Agile Software Development (2001)

- Agile Development

- Beck et al 2001

Value-driven

rather than

Plan-driven

Dynamic

rather than

Static

- Agile Manifesto

- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

– Beck et al 2001

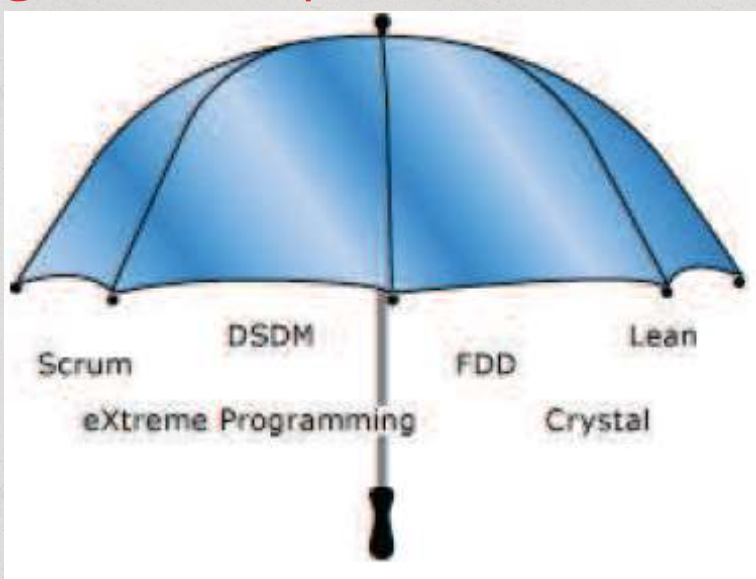
54

## Agile Development



55

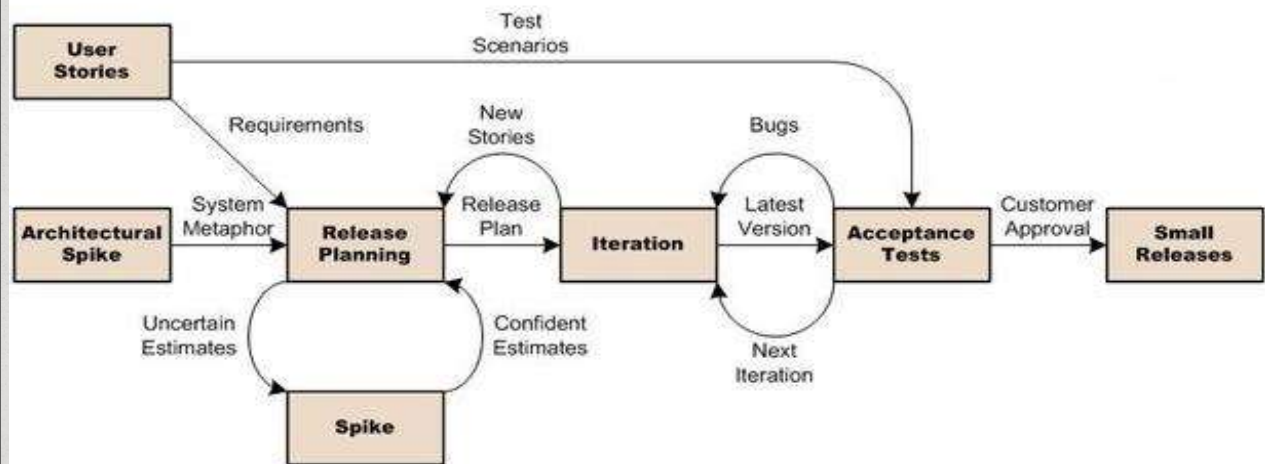
## Agile Development



56

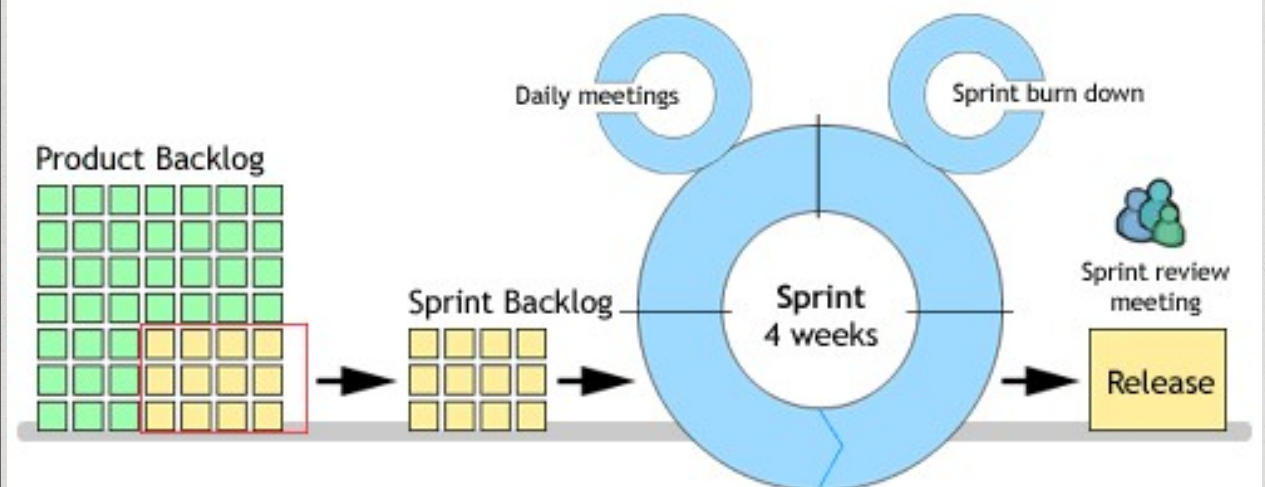


## Agile Development – eXtreme Programming



57

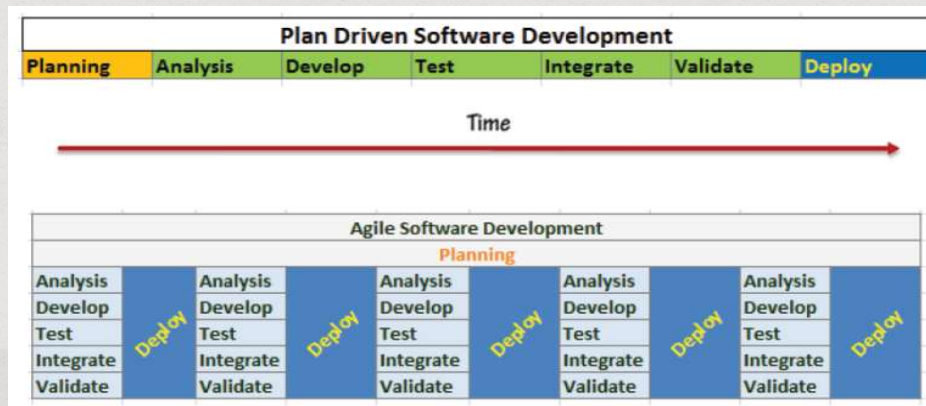
## Agile Development – Scrum



58

## Agile vs. Plan-Driven

- One increment vs. frequent release
- Start after plan vs. plan all the time



59

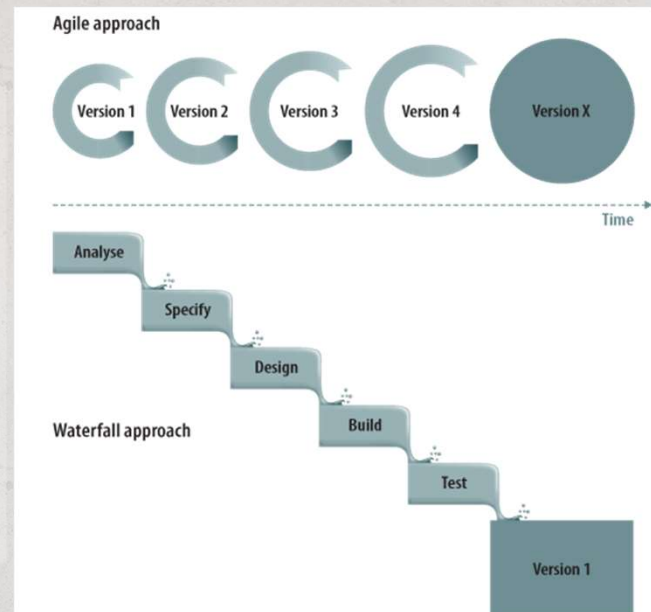
## Agile vs. Plan-Driven

- Agile vs. plan-driven
  - Communications: continual vs. formal
  - Quality: iterative vs. lots of gates to control
  - Inspect: as being done vs. after completion
  - Start: filling a need vs. predicting deliverable

60

## Agile vs. Waterfall

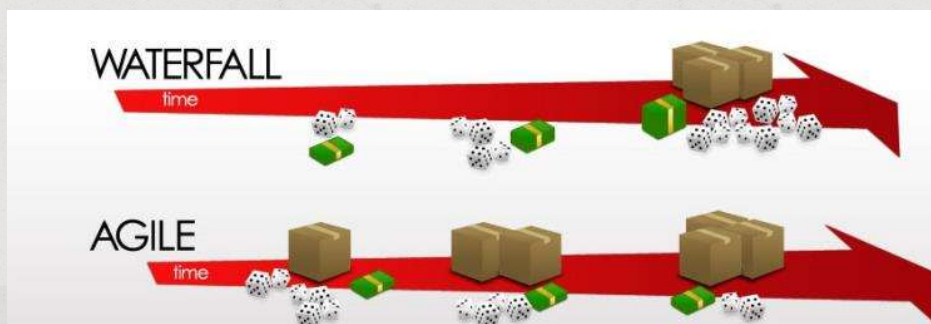
- Phase vs. iteration
- Sequential vs. concurrent



61

## Agile vs. Waterfall

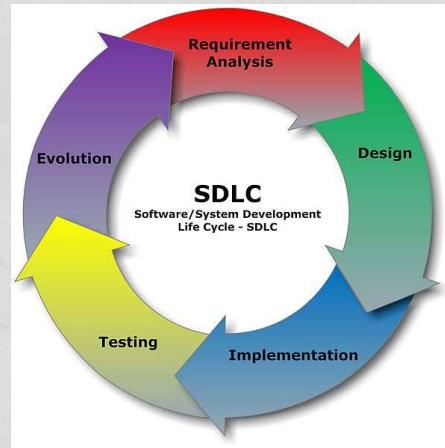
- Working code after each iteration
- Evolving requirements
- Early verification



62



## SDLC – A More Software-Oriented View



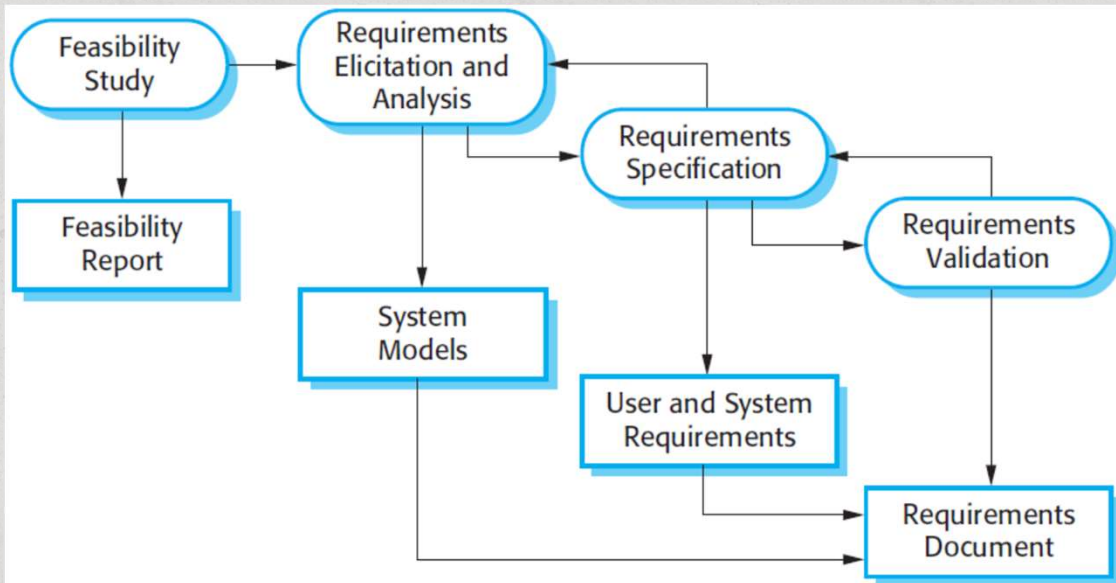
63

## Software Specification

- The process of establishing what services are required and the constraints on the system's operation and development.
- Requirements engineering process
  - Feasibility study
  - Requirements elicitation and analysis
  - Requirements specification
  - Requirements validation

64

## Requirements Engineering



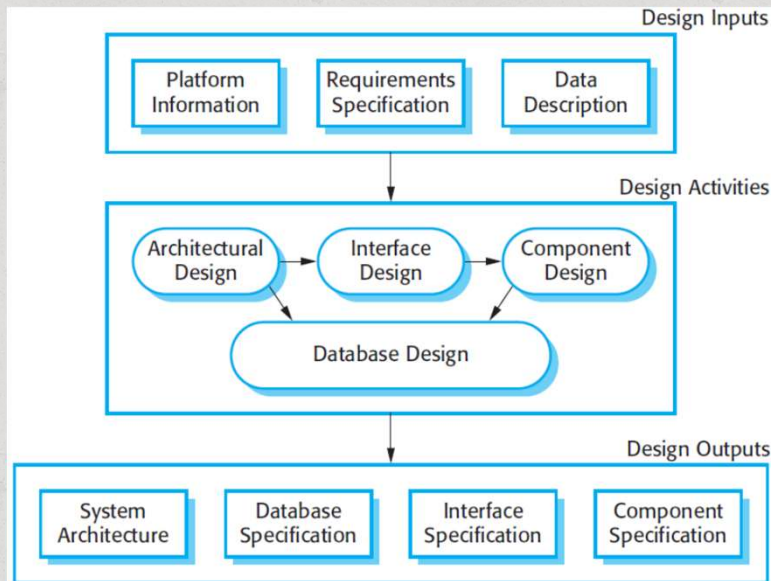
65

## Design and Implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

66

## Model of Design Process



67

## Design Activities

- *Architectural design*, where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.
- *Interface design*, where you define the interfaces between system components.
- *Component design*, where you take each system component and design how it will operate.
- *Database design*, where you design the system data structures and how these are to be represented in a database.

68

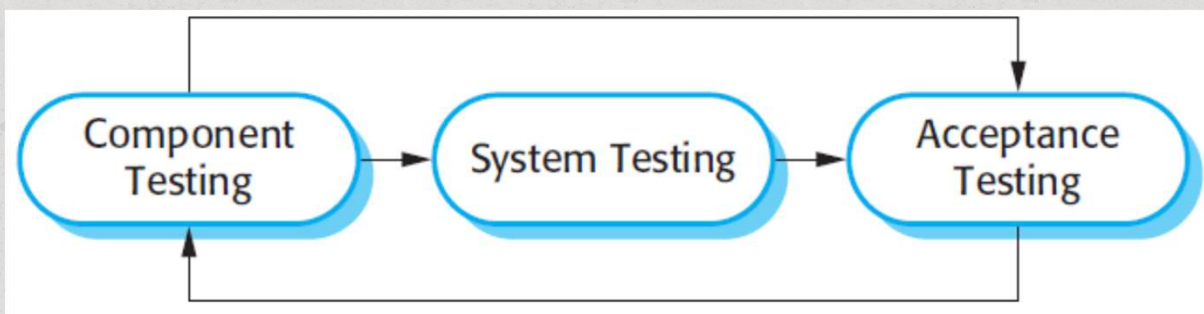


## Software Validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V & V activity.

69

## Stages of Testing



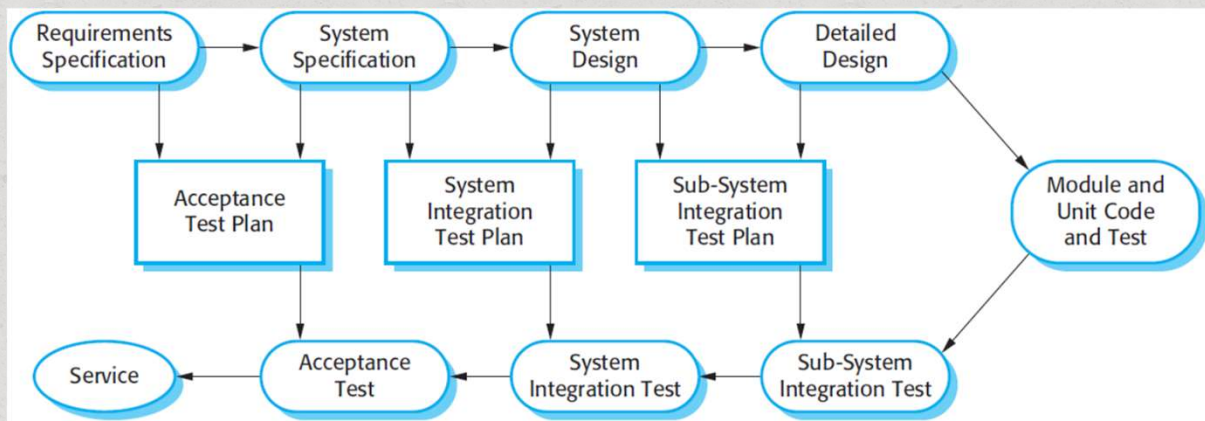
70

## Testing Stages

- Development or component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- System testing
  - Testing of the system as a whole.
  - Testing of emergent properties is particularly important.
- Acceptance testing
  - Testing with customer data to check that the system meets the customer's needs.

71

## Testing in Plan-Driven Process



72

## Summary

- Demand for high-quality software
  - Software Engineering
- Software Development Lifecycle
  - Requirements Engineering
  - ...
  - Software Testing
  - ...

73

## References

- Process models are covered in most Software Engineering Text Books
  - Sommerville, I. *Software Engineering*
  - Pressman, R. *Software Engineering: A Practitioner's Approach*
  - Pfleeger, S. L. *Software Engineering Theory and Practice*
  - Larman, C. *Applying UML and Patterns Introduction to OOA/D and Iterative Development*

74



## Next

- Tutorial
  - A “mini” software development project
- Next week
  - Software Requirements Specification