



**SWINBURNE**  
\* \*  
UNIVERSITY OF  
TECHNOLOGY

**SWE30003**  
**Software Architectures and Design**  
  
Lecture 1  
Issues in Software Design

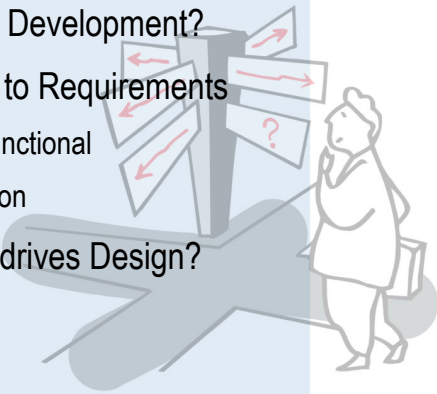


1

## Outline



- Unit of Study Overview
- What Drives Software Development?
- From Business Goals to Requirements
  - Functional vs. Non-functional
  - Verification & Validation
- What is Design, what drives Design?



2

2

## Software Development Activities



<b>Requirements Elicitation</b>	Establish customer's needs
<b>Requirements Analysis</b>	Model and specify the requirements ("what")
<b>Design</b>	Model and specify a solution ("how")
<b>Implementation</b>	Construct a solution in software
<b>Testing</b>	Verify the solution against the requirements
<b>Maintenance</b>	Repair defects and adapt the solution to new requirements

**NB: these are ongoing activities, not sequential phases!**

3

3

## Focus of Unit of Study

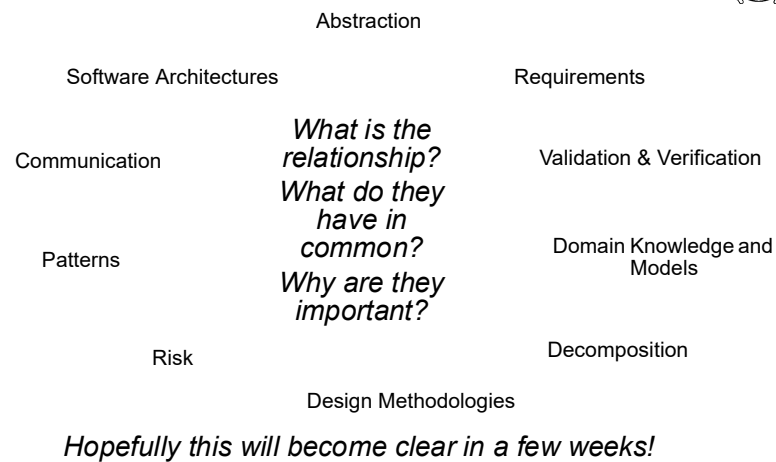


<i>Requirements Elicitation</i>	Establish customer's needs
<b>Requirements Analysis</b>	Model and specify the requirements ("what")
<b>Design</b>	Model and specify a solution ("how")
<b>Implementation</b>	Construct a solution in software
<i>Testing</i>	Verify the solution against the requirements
<i>Maintenance</i>	Repair defects and adapt the solution to new requirements

4

4

## What we will be talking about...



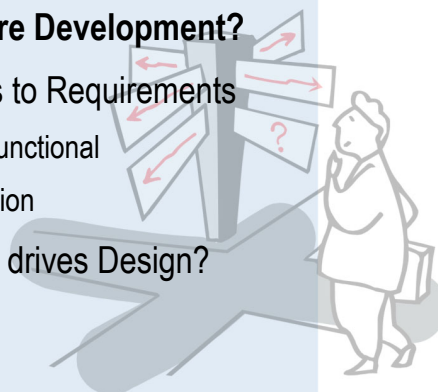
5

5

## Outline



- Unit of Study Overview
- **What Drives Software Development?**
- From Business Goals to Requirements
  - Functional vs. Non-functional
  - Verification & Validation
- What is Design, what drives Design?



6

6

## Why do we build Software Systems?



- To keep/retain customers
- To increase customer base
- To improve customer relationships
- To improve productivity/efficiency (within organization/team)
- Technology used no longer supported/outdated
- Changes in environment (legal, social)

☞ *To achieve some “Business” Goals!*

7

7

## Categories of Business Goals



**Business goals** fall into five broad categories:

- Reduce cost of ownership: development, maintenance, deployment, operation.
- Improve the quality of the system(s) compared with its predecessors with respect to performance, modifiability, security, reliability etc.
- Improve the capabilities/functionality offered by the system compared to its predecessors.
- Improve organization's market position.
- Improve external confidence in either the organization or the system

☞ *A particular system has generally more than one goal!*

8

8

## Views of Stakeholders



- Many systems involve a variety of different stakeholders:
  - ☐ Customers
  - ☐ Developers
  - ☐ Managers
  - ☐ Users
- All of them have different *views* and *priorities*
  - ☞ May even lead to *conflicting* goals!
  - ☞ Development approaches must be able to cater for different views and priorities!

9

9

## Business Goals (cont.)



*Priority of goals* need to be specified:

- Some goals are “nice to have”, others “need to have”, some “absolutely critical”
- Developers sometimes have to “push back” or make trade offs. Knowing priority gives insights.

*Source of goals* need be specified:

- Some goals are *inherent* to the system being developed
- Some goals are a result of market analysis
- Some goals are *arbitrary* - could cause problems!

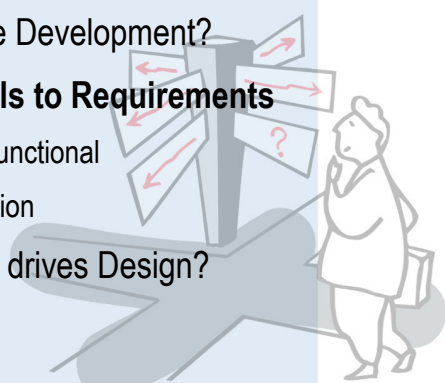
10

10

## Outline

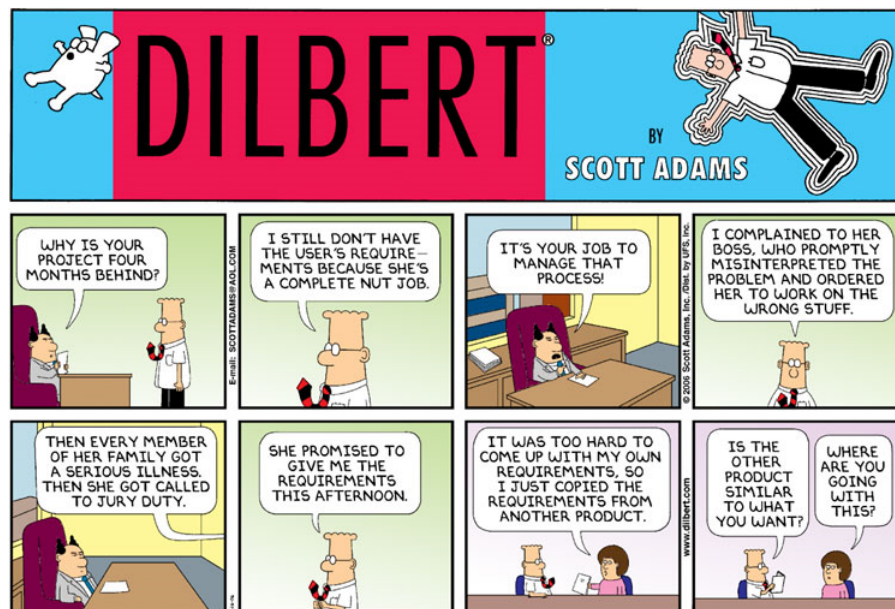


- Unit of Study Overview
- What Drives Software Development?
- **From Business Goals to Requirements**
  - Functional vs. Non-functional
  - Verification & Validation
- What is Design, what drives Design?



11

11



© Scott Adams, Inc./Dist. by UFS, Inc.

12

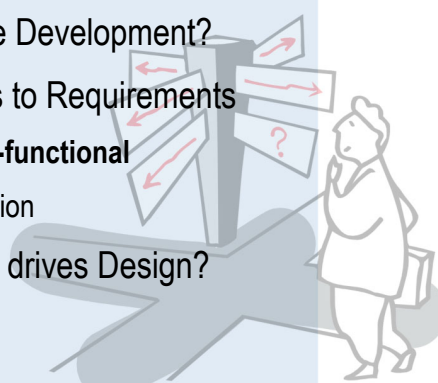
[http://en.wikipedia.org/wiki/Fair\\_use](http://en.wikipedia.org/wiki/Fair_use)

12

## Outline



- Unit of Study Overview
- What Drives Software Development?
- From Business Goals to Requirements
  - **Functional vs. Non-functional**
  - Verification & Validation
- What is Design, what drives Design?



13

13

## Functional and Non-functional Requirements



Functional requirements describe system *services* or *functions*

- Compute sales tax on a purchase
- Update the database on the server ...
- ☞ *In essence, anything that can be **directly expressed in code**...*

Non-functional requirements (also known as **quality requirements**) are *constraints* on the services and/or the development process

- "Reducing cost of deployment" cannot be directly expressed in code...

Domain requirements stem from the application domain of a system

- may be functional or non-functional

14

14

## Non-functional Requirements



<b>Product requirements:</b>	specify that the delivered product <i>must behave</i> in a particular way e.g. <i>execution speed, reliability, etc.</i>
<b>Organizational requirements:</b>	are a consequence of <i>organizational policies</i> and procedures e.g. <i>process standards used, implementation requirements, etc.</i>
<b>External requirements:</b>	arise from <i>factors which are external</i> to the system and its development process e.g. <i>interoperability requirements, legislative requirements, etc.</i>

15

15

## Examples of Non-functional Requirements



<b>Product requirement</b>	It shall be possible for all communication between the console and the user to be expressed in the <i>standard ASCII character set</i> .
<b>Organisational requirement</b>	The <i>system development process</i> and deliverable documents shall conform to the process and deliverables defined in <i>XYZCo-SP-STAN-95</i> .
<b>External requirement</b>	The system shall provide facilities that allow any user to check if personal data is maintained on the system. <i>A process must be defined and supported in the software that will allow users to inspect personal data and to correct any errors in that data.</i>

☞ We will talk more about non-functional requirements in Week 3

16

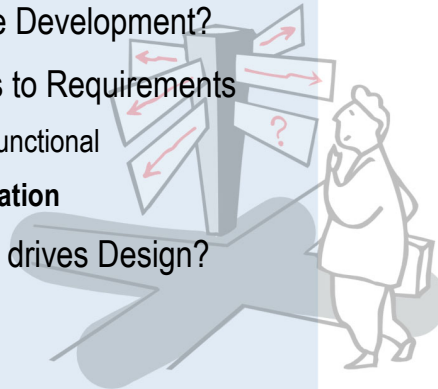
16



## Outline



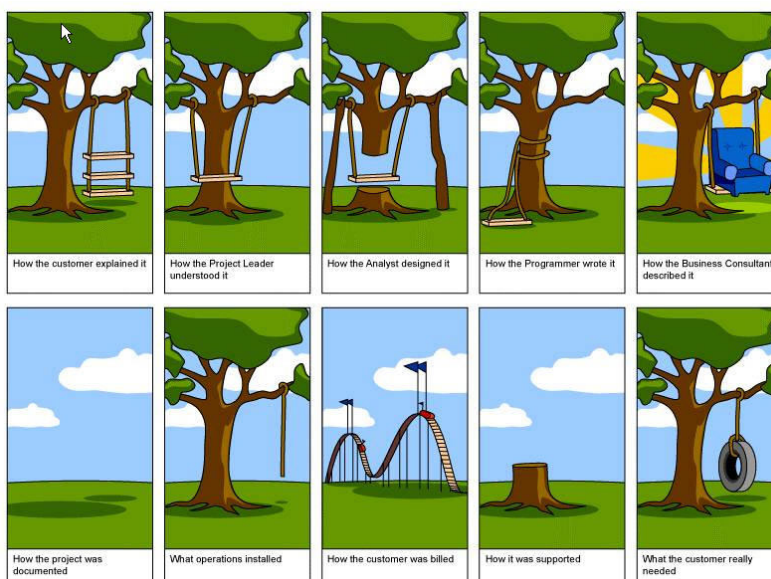
- Unit of Study Overview
- What Drives Software Development?
- From Business Goals to Requirements
  - Functional vs. Non-functional
  - **Verification & Validation**
- What is Design, what drives Design?



17

17

## Impedance Mismatches



18

18

## Verification and Validation



### Verification:

- Are we *building the product right*?
  - i.e., does it conform to the specification(s)?

### Validation:

- Are we building the *right product*?
  - i.e., does it meet stakeholders' *expectations*?

19

19

## Requirements Verifiability



Requirements must be written so that they can be *validated* and *objectively verified*.

### Imprecise:

- "The system should be *easy to use* by experienced controllers and should be organized in such a way that *user errors are minimized*."  
Terms like "*easy to use*" and "*errors shall be minimized*" are useless as specifications as they are too *vague*.

### Verifiable:

- "Experienced controllers should be able to use all the system functions *after a total of two hours training*. After this training, the average number of errors made by experienced users should not exceed two per day."

☞ You will hear *a lot* about requirements verifiability throughout the semester!

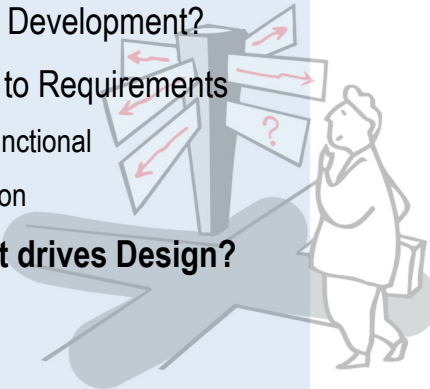
20

20

## Outline



- Unit of Study Overview
- What Drives Software Development?
- From Business Goals to Requirements
  - Functional vs. Non-functional
  - Verification & Validation
- What is Design, what drives Design?



21

21

## Software Design – a “Definition”



*“Software design is the activity of drawing UML class diagrams once a software system has been coded in order to satisfy the requirements of unit conveners.”*

*Would you trust the software of an air-traffic control system that was implemented without prior structural analysis etc.?*

22

22

## Design vs. Design



*“Design is the creative process of transforming the problem into a solution. The description of the solution is also called the design.”*

**Activity**

— Shari Lawrence Pfleeger, 1998

*“The design of a system determines a set of components and inter-component interfaces that satisfy a specified set of requirements.”*

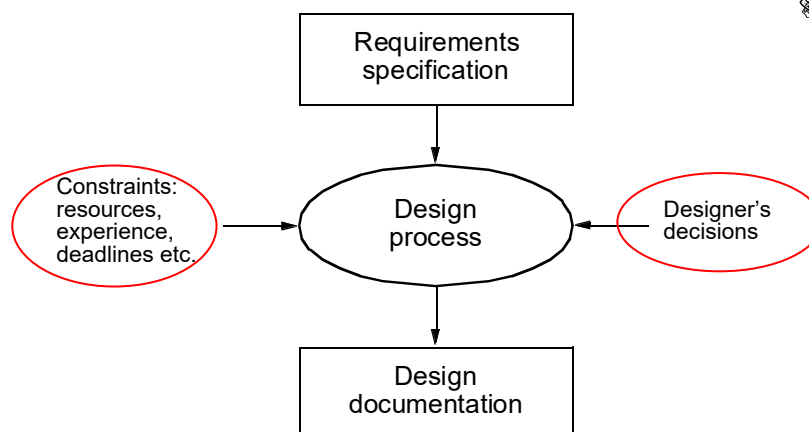
**Artefact**

— DeMarco, 1982

23

23

## Design Process



Source: David Budgen,  
*Software Design*, Addison-  
Wesley, 1994

24

24

## The Process of Decomposition



*Decomposition* is a process that helps to manage the complexity of a software system

- ☐ Starting with a *high-level view* of the system.
- ☐ Creating the low-level details of the features of the system in turn.
- ☐ Stop when we are satisfied with the level of detail.
- ☞ Also known as *Divide-and-Conquer*.

25

25

## Models



- A model is a *simplification of reality*.
- Models are created for better understanding of the *problem*, the *domain*, or the *system* to build.
- A model can capture and represent,
  - ☐ Structure (static model)
  - ☐ Behavior (dynamic model)
- Examples:
  - ☐ Architectural blue-print of a house
  - ☐ Hand-drawn screen ("wire-frame")
  - ☐ Class diagram



26

26



Q: *What has a bigger impact on the design structure of a software system: functional requirements or quality requirements?*

A: Quality requirements determine most structural design decisions, not functional requirements!

27

27



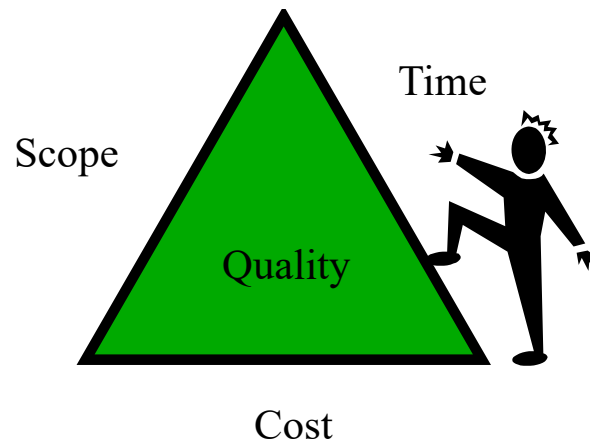
## Quality-Driven Design

- Quality requirements are those that *drive* the structural design of the software system.
- Leads to several questions:
  - ☐ How are quality requirements *specified*?
  - ☐ How are quality requirements *validated*?
    - ☞ *This is also important for functional requirements!*
  - ☐ How are quality requirements *achieved*?
  - ☐ How can understanding of the impact of quality attributes be used during the design process?
- ☞ *We will attempt to answer these questions over the next few weeks.*

28

28

## Forces in Software Projects



29

29

## Question to Answer Week 1



*In your own words, characterize the difference between **verification** and **validation** and discuss where in the software development lifecycle (SDLC) verification and validation activities should be conducted, respectively.*

30

30

## Catch-up Reading Lecture 1



- Len Bass, Paul Clements, Rick Kazman, *Software Architecture in Practice (2<sup>nd</sup> Edition)*, Addison-Wesley, 2003, Chapter 1 (available electronically through the Swinburne Library)

31

31

## Required Pre-Reading for Lecture 2



- Soren Lauesen, *Task Descriptions as Functional Requirements*, IEEE Software, March 2003. (on Canvas, under week 2)

32

32