

# Conceptual Architecture

Lian Loke  
John Reekie  
University of Technology, Sydney

Terms of Use: Creative Commons Attribution-ShareAlike 2.0  
<http://creativecommons.org/licenses/by-sa/2.0/>

---

---

---

---

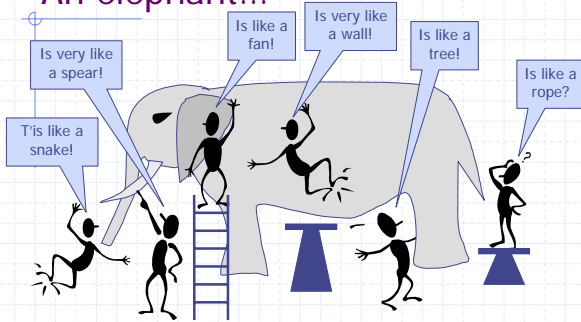
---

---

---

---

## An elephant...



As told by American poet John Godfrey Saxe, based on an Indian fable

---

---

---

---

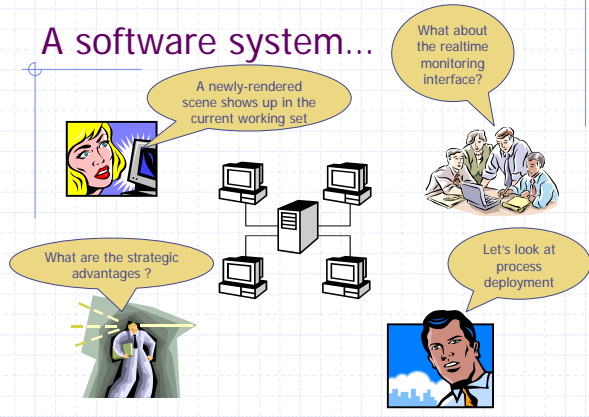
---

---

---

---

## A software system...



---

---

---

---

---

---

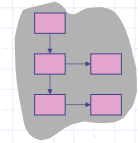
---

---

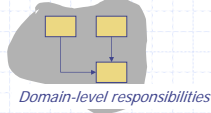
## Architectural views

A view expresses a particular cross-section of architectural concerns

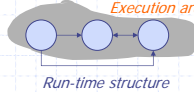
*Implementation architecture*



*Conceptual architecture*



*Execution architecture*



Some authors recommend/prescribe a particular set of views... others don't...

---

---

---

---

---

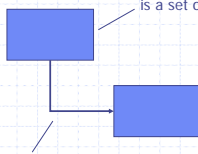
---

---

---

## Components and connectors

A conceptual component is a set of responsibilities



Sample responsibilities:  
•PlayBackClipSequence  
•SynchronizeWithVideo  
•PrefetchClips

A conceptual connector indicates communication between components

The **conceptual architecture** structures the system in terms of its domain-level responsibilities

---

---

---

---

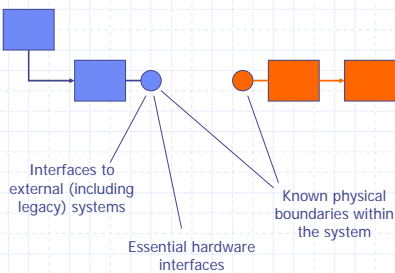
---

---

---

---

## External interfaces



Note: A stakeholder is **not** an external system!

---

---

---

---

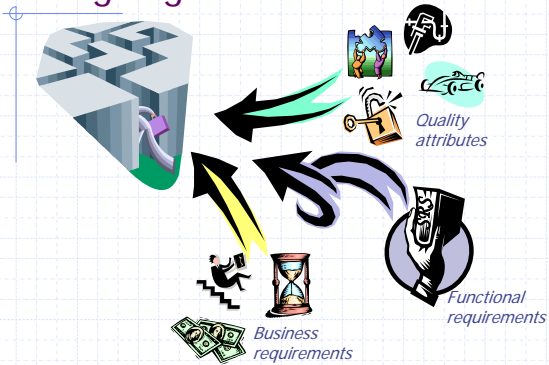
---

---

---

---

## Designing an architecture?



---

---

---

---

---

---

---

---

## Don't get stuck



*Design is an iterative process. At each step, discover more about the problem, and more about the solution.*

*Let's start with a simple procedure...*

---

---

---

---

---

---

---

---

## 1. Obtain a system narrative

Custom Shooz plan to advertise using conventional means, but want the website to be a location where customers can find out about their custom range, get the measurement kit, and customize and order shoes. They also want the site to interface to their accounting system.

The President of Custom Shooz, Funk O. Sole, explains:  
"So, what we did was develop a little measurement kit that we send out to folks and they have to send it back. We've improved it over the last couple of years so that it's almost foolproof. Once we have the customer's measurement kit in, we can produce almost any shoe from our range - all the custom stuff, like stitched-on patterns, dye colours and finishes, laces and buckles, can be done without them ever being within a thousand miles of our store!"

(Extract from system narrative in Workbook)

---

---

---

---

---

---

---

---

## 2. Identify key concepts

Custom Shooz plan to **advertise** using conventional means, but want the **website** to be a location where **customers** can find out about their **custom range**, get the **measurement kit**, and **customize and order shoes**. They also want the site to **interface to their accounting system**.

The President of Custom Shooz, Funk O. Sole, explains:  
 "So, what we did was develop a little **measurement kit** that we send out to folks and they have to send it back. We've improved it over the last couple of years so that it's almost foolproof. Once we have the customer's measurement kit in, we can **produce almost any shoe** from our **range** - all the custom stuff, like **stitched-on patterns, dye colours and finishes, laces and buckles**, can be done without them ever being within a thousand miles of our store!"

---

---

---

---

---

---

---

---

---

---

## 3. Refine to components

- Advertise - abstract concept ⇒ X
- Website - implementation ⇒ ?
- Customers - stakeholder ⇒ **Customer account + Personalised page**
- Custom range ⇒ **Product range**
- Measurement kit ⇒ **Customer measurements**
- Customise shoe** ✓
- Order shoe** ✓
- Accounting I/F - external system ⇒ **Acct I/F**
- Produce shoe - function/external system ⇒ **Shoe production**
- Patterns and finishes - part of Customise shoe

---

---

---

---

---

---

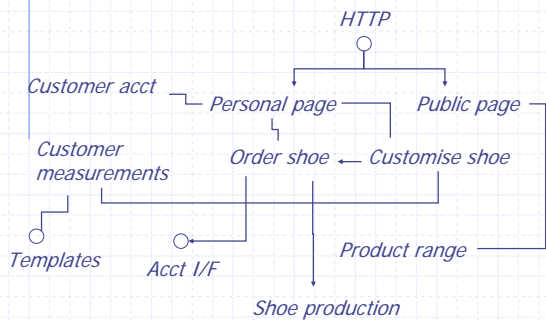
---

---

---

---

## 4. Draw and connect




---

---

---

---

---

---

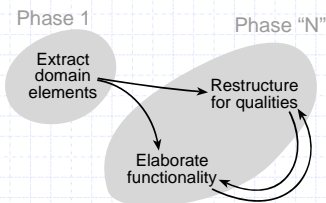
---

---

---

---

## This is just a starting point



A series of further iterations elaborates the architecture to improve functionality and quality attributes



---

---

---

---

---

---

---

---

## Refine the architecture

- ◆ Add or split components
- ◆ Clarify responsibilities
- ◆ Identify stereotypes
- ◆ Create data models
- ◆ Explore behavior



A component is a set of *related* responsibilities. So, split a component if responsibilities are *not* related ...



Replication can be considered at this stage, to account for performance and availability needs

---

---

---

---

---

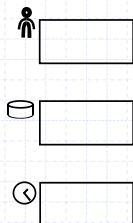
---

---

---

## Conceptual stereotypes

- ◆ Does a component have special types of responsibilities?
  - User presentation
  - Persistent storage
  - Realtime response



A **stereotype** indicates that a component (or in UML, a class) has certain properties or attributes.

---

---

---

---

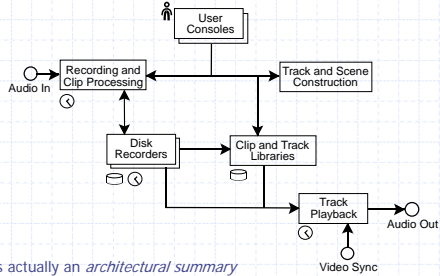
---

---

---

---

## A stereotype example



This is actually an *architectural summary diagram*. It is useful for providing an at-a-glance overview of a complex system.

---

---

---

---

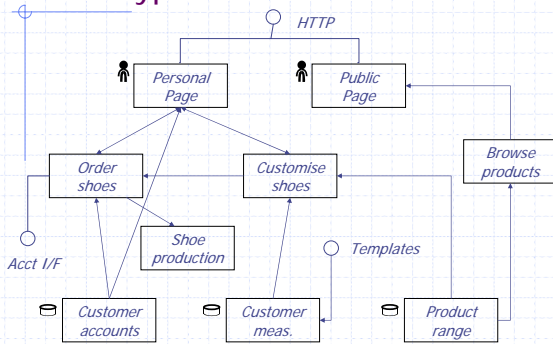
---

---

---

---

## Custom Shooz architecture with stereotypes




---

---

---

---

---

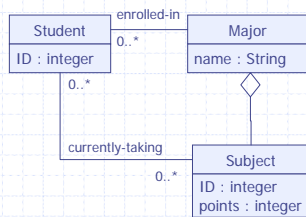
---

---

---

## Data models

- ◆ A data model captures the *essential* structure of data
  - Data along connectors
  - Persistent data




---

---

---

---

---

---

---

---

## What is behaviour



A system has function, structure and behaviour

- ◆ Behaviour is the set of actions that the system performs
- ◆ Behaviour can be explored through:
  - Role-play
  - Use Case maps
  - Sequence diagrams

---

---

---

---

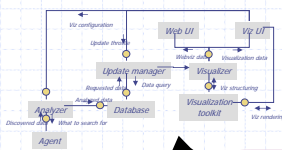
---

---

---

---

## How can we explore deeper?



The system must have some behavior in response to activity in the usage narratives

One, two! One, two! And through and through  
The vorpal blade went snicker-snack!  
He left it dead, and with its head  
He went galumphing back...

He took his vorpal sword in hand:  
Long time the manxome foe he sought  
So rested he by the Tumtum tree,  
And stood awhile in thought.



And as in uffish thought he stood,  
The Jabberwock, with eyes of flame,  
Came whiffling through the tulgey wood,  
And burbled as it came!



Diagram and text illustrating system behavior and usage narratives.

---

---

---

---

---

---

---

---

## Extract events from narratives

Elvis runs a café in Las Vegas. He's quite a character with a devastating sense of style. He's heard about Custom Shoos and wants to *check out their stuff*. He jumps on the website and likes the look of the Cuban heeled boots with embossed trim. What he really wants is a pair like those pictured but in white leather with black and silver trim to fit his rather large feet with an unusually high instep. He *plays around* with the possibilities by *customising* the basic design to suit his tastes. He doesn't want to register and place an order just yet, so he *adds his customised design to a wish-list* so he can access it next time he visits the site.

➡ *browseRange*  
*customiseShoe*  
*saveToWishList*

Text and arrow illustrating the extraction of events from a narrative.

---

---

---

---

---

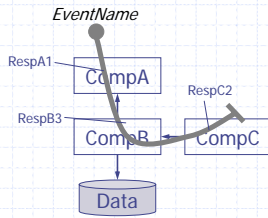
---

---

---

## Events trigger use-case maps

- ◆ Use-case maps allow us to visualise a path of action through a system
  - A trace shows the sequence of activities
  - Activity is triggered by an event
  - Each time the trace crosses a component, it exercises a responsibility



Use-case maps facilitate understanding of macroscopic behaviour

---

---

---

---

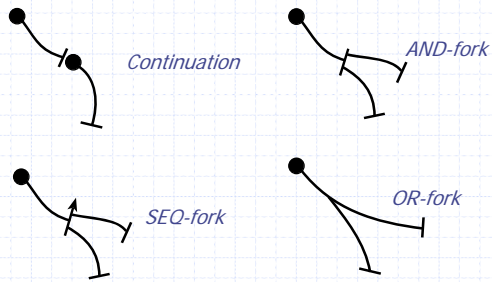
---

---

---

---

## Use-case map notation




---

---

---

---

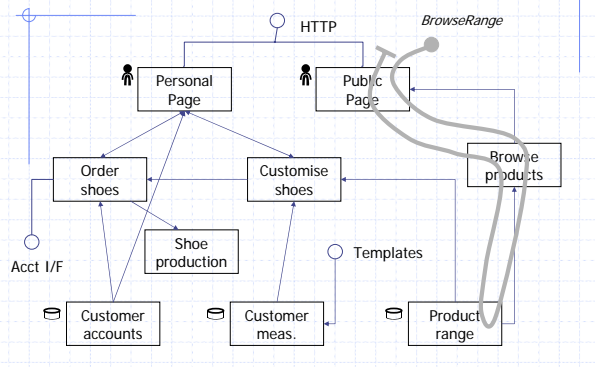
---

---

---

---

## Custom Shooz Use Case Map (1)




---

---

---

---

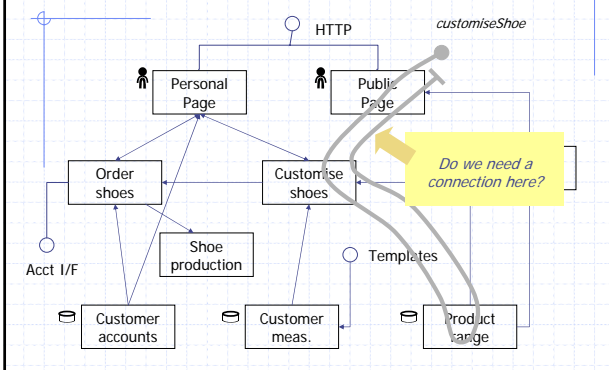
---

---

---

---

## Custom Shooz Use Case Map (2)



---

---

---

---

---

---

---

---

## An architectural gotcha...

An AntiPattern that occurs frequently in Software Architecture assignments is...

An *anti-pattern* describes an undesirable solution to a recurring set of contextual factors. It also describes how to refactor into a desirable solution.

---

---

---

---

---

---

---

---

## The Blob...

If unchecked, The Blob will eat your entire Software Architecture

Architecture

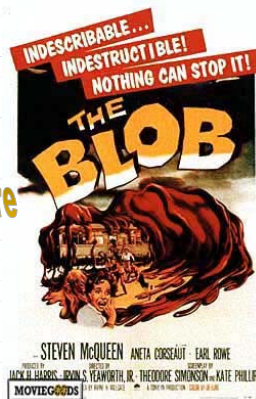
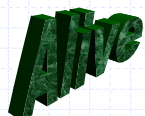


Image from <http://www.moviegoods.com>

---

---

---

---

---

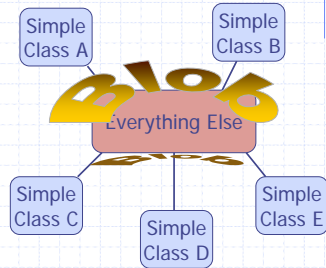
---

---

---

## Object-oriented version

- ◆ A complex controller class surrounded by simple data classes
- ◆ A procedural design in an OO environment
- ◆ Originates with requirements that dictate a procedural solution
- ◆ A result of poor growth strategies



---

---

---

---

---

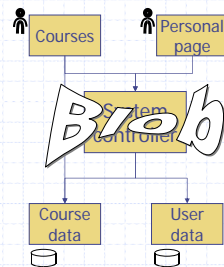
---

---

---

## Architectural version

- ◆ A complex application or logic component surrounded by presentation or data components
- ◆ Often has names containing "controller" or "manager"
- ◆ Indicates inability or unwillingness to decompose the system based on domain-level responsibilities



---

---

---

---

---

---

---

---

## Refactoring

- ◆ Object-oriented re-design
  - Move behavior away from the Blob
  - Identify related "lumps"
  - Coalesce with related simple classes
- ◆ Architectural re-design
  - Identify all Blob responsibilities
  - Split into multiple components based on the responsibilities of the Blob and surrounding components

- ◆ Or:
  - Scrap it and start over

*AntiPatterns provide real-world experience in recognizing recurring problems & providing a detailed remedy*

---

---

---

---

---

---

---

---