UNIVERSITY OF TECHNOLOGY SYDNEY

SOFTWARE ARCHITECTURE

Introduction

Dr Tom McBride

University of Technology, Sydney

UTS:
ENGINEERING AND
INFORMATION
TECHNOLOGY

Topics

- What is architecture
- Why bother to design an architecture
- Why is architecture so hard
- Becoming a good architect
- Common architectural challenges

Text books


A
Software
Architecture
Primer
John Reekie
Rohan McAdam


Software
Architecture
in Practice
Second Edition
Len Bass
Paul Clements
Rick Kazman


Essential
Software
Architecture
Second Edition
Ian Gorton
Springer


HASSAN GOMAA
SOFTWARE
MODELING
& DESIGN
UML, USE CASES, PATTERNS,
& SOFTWARE ARCHITECTURES
CAMBRIDGE

Blogs and other sources

- http://www.infoq.com/architecture-design/

- To be extended as good sources are found

UNIVERSITY OF TECHNOLOGY SYDNEY

WHAT IS ARCHITECTURE

Architecture – definitions

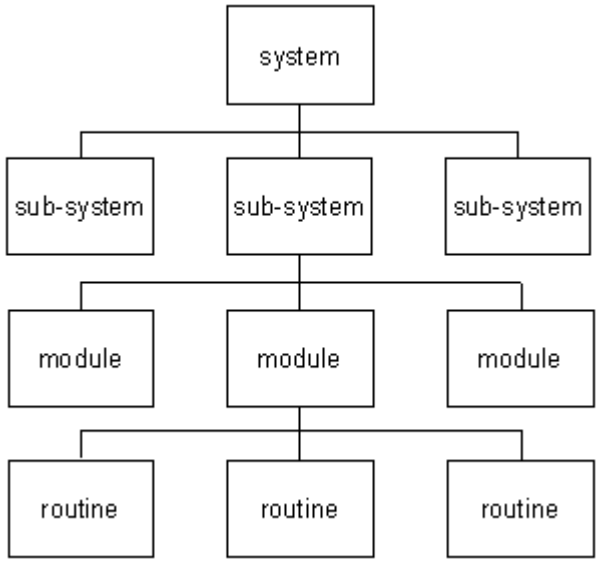
- The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships between them. (Bass, L., Clements, P. and Kazman, R. (2003), *Software architecture in practice, Addison-Wesley, 2nd ed)*
- The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. (ISO 42010/IEEE Std 1471)

What is architecture

- Describing what architecture is says very little about its importance or what to pay attention to.
- An architecture supports what a system can do and prevents what it should not do.
- An architecture is not a passive element of a system but instead provides the structure that enables the system to function, to become a participant in its environment.

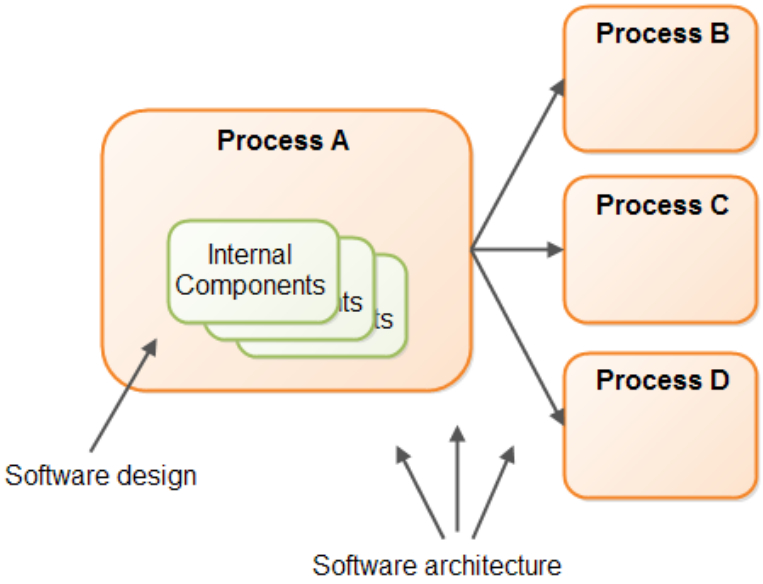
Structural elements of a system

- An architecture is an abstraction of a system
  - Concerned about interactions between structural elements
  - What a system element does and how it does it are architecture concerns. i.e. its functions and behaviour
  - The implementation details of those elements are not usually architectural concerns
  - Example, a component that encrypts data may be architectural but the encryption algorithm itself is not usually an architectural concern.
- Architecture exists at many levels
  - High level system architecture
  - Low level component architecture
  - Usually stop at the level of coded routines.



The difference between architecture and design can be a matter of scale

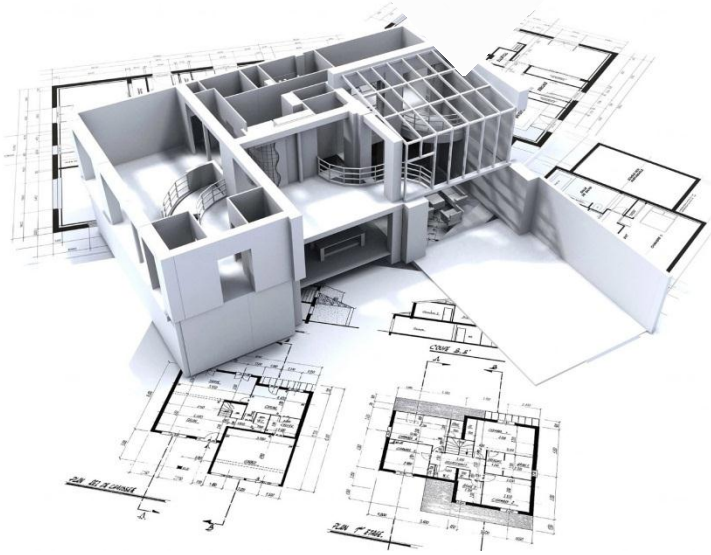
- Architecture concerns structure – design does not always concern structure
- Architecture of a single program
  - Classes/objects and their relationships could be considered as architecture
  - Implementation details of a class's methods is not considered architecture
- Usually a matter of scale and abstraction
  - A system is made up of sub-systems
  - Which are made up of components
  - Which are made up of sub-components or objects



Architecture of…

- A building
  - The walls, spaces, doors, columns are architectural
  - The wall surface and appearance is not usually considered part of the architecture
  - Lifts, corridors, stairs, plumbing and other services support the life within the building

- A computing system
  - The systems/sub-systems on servers and their responsibilities are architectural
  - The interfaces between sub-systems are architectural
  - The protocol used to communicate between sub-systems is not usually considered to be architectural



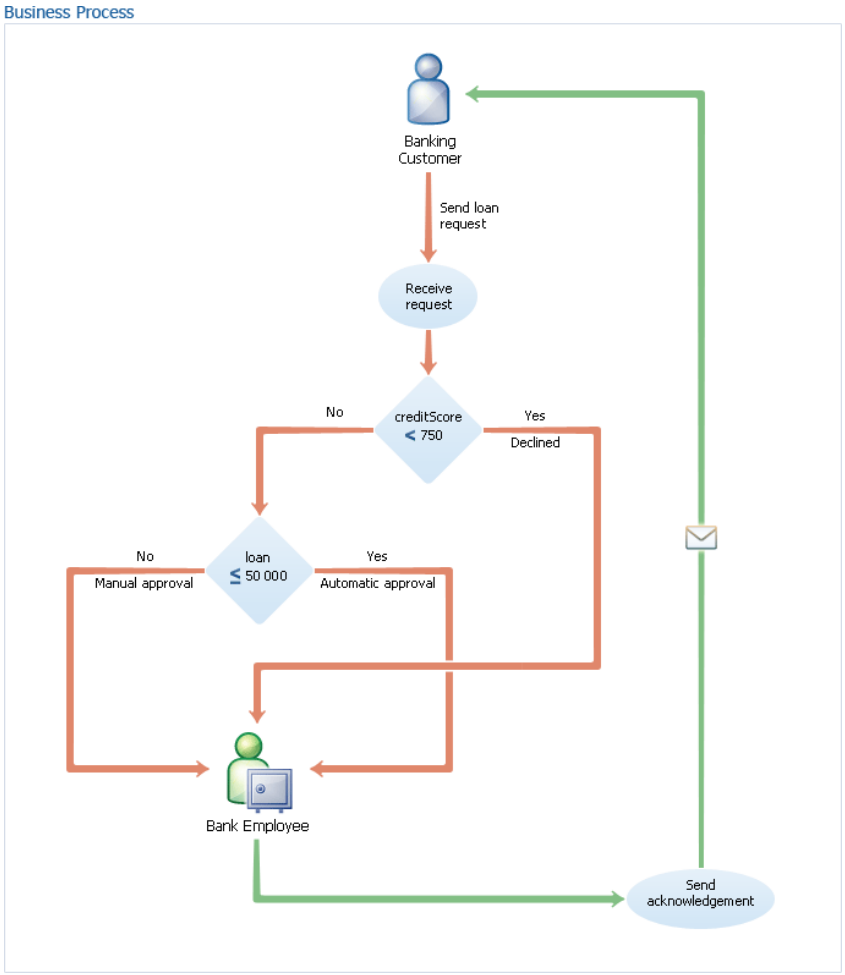
Summary

- Definition helps decide what is or is not architecture.
- Architecture is concerned with structural elements of a system, their externally observable properties and the relationships between them, but its purpose is to support the system's participation in the work around it.
- Not everything has an architecture

WHY IS ARCHITECTURE IMPORTANT

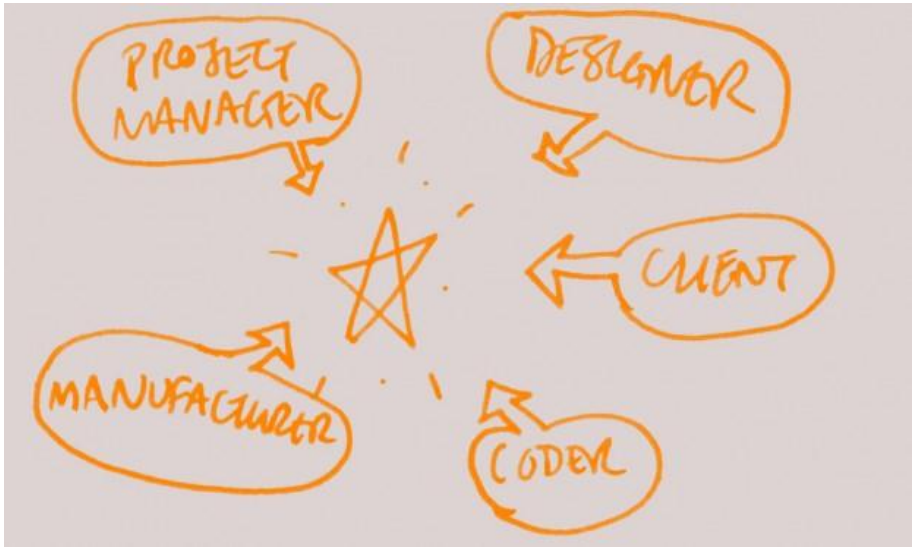
Architectures allow us to explore the problem

- Many systems now are being developed to support poorly understood problems
  - E.g. develop a system to increase market presence
  - E.g. develop a web based consumer banking system
- Architectures are relatively inexpensive to develop until they are implemented
- An architecture is a model of an intended product. Constructing a model allows us to understand and predict the developed product's capabilities.



Architectures allow communication among stakeholders

- An architecture is a "boundary object"
- The business owner can understand what the system will do.
- The architect can understand what components are needed to provide the required capabilities.
- Developers can understand what the properties of the components to be developed.

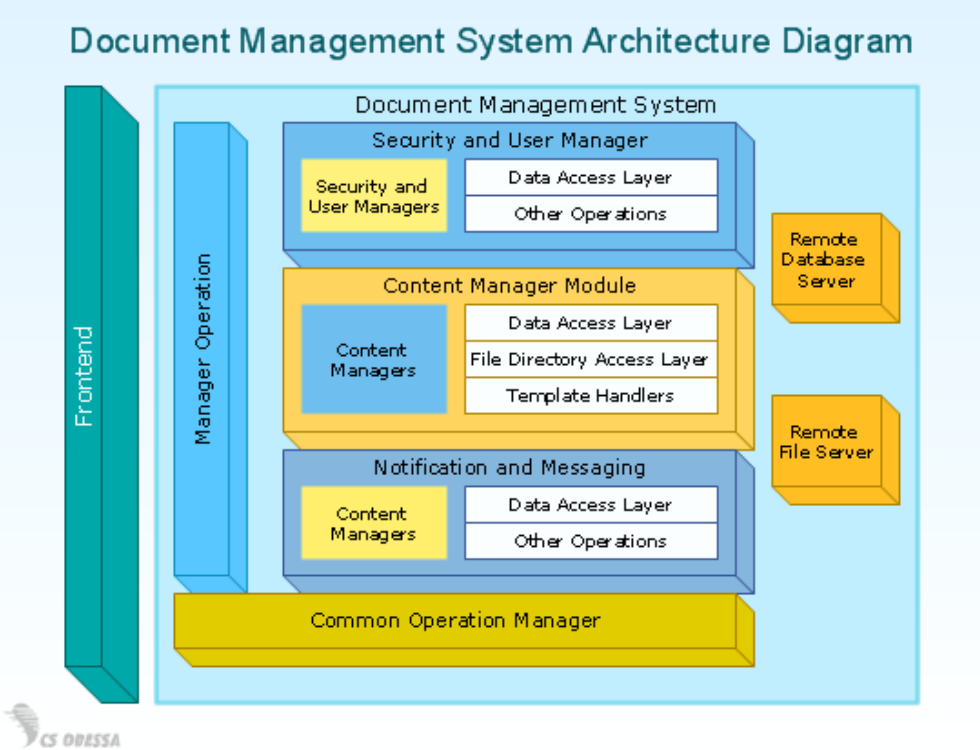


A transferable abstraction of a system

- Software architecture constitutes a relatively small, intellectually graspable model of how the system is structured and how its elements work together.
- This model is transferable across systems – we can use it to develop another, similar system

The importance of architecture

- Architectures last a long time and are very hard to change once they are implemented
- Architectures determine what can and can't be done
- Architectures determine behaviour (qualities)



Architectures last a long time

- Programs come and go but architectures go on forever.
- Component parts of a system might be replaced but replacing the entire system is seldom done.

Architecture development is limited by the context

- An architecture must satisfy its objectives
- Architecture development is limited by its constraints
  - Knowledge of the problem and possible solutions
  - Legacy systems and decisions
  - Time and budget
  - Technology available for the solution
  - Modelling languages

Architectures limit what can and can't be done

- It is very hard to alter architectural constraints once they are in place.
  - The maximum size of a record
  - The maximum number of records that can be processed
  - The maximum number of user that can be supported
  - The ease of maintenance and modification
  - How secure the system can be
- Examples of architectural constraints
  - Specific connections between components instead of flexible interfaces like message bus or façade
  - Loosely coupled vs tightly coupled components

An example of architectural constraints

- A company had developed several systems to manage steel production.
- Each was developed separately and most used some information or service provided by one or more of the other systems.
- The end result was a rat's nest of interconnected systems.
- The computers on which they ran were being phased out.
- The systems had to be redesigned separated and re-implemented without stopping the overall services.

Architectures determine emergent characteristics of a system

- Functionality can usually be changed but behavioural qualities like performance, responsiveness, security are mostly determined by the architecture.
- Systems that were designed for very high performance are difficult to maintain.
- Systems designed to be easily maintained and modified are usually difficult to speed up.

Summary

- Architectures last a long time.
- Architectures are boundary objects between customer and developer.
- Architectures are models of an intended system that allow us to understand, explore and predict the capabilities of the developed system

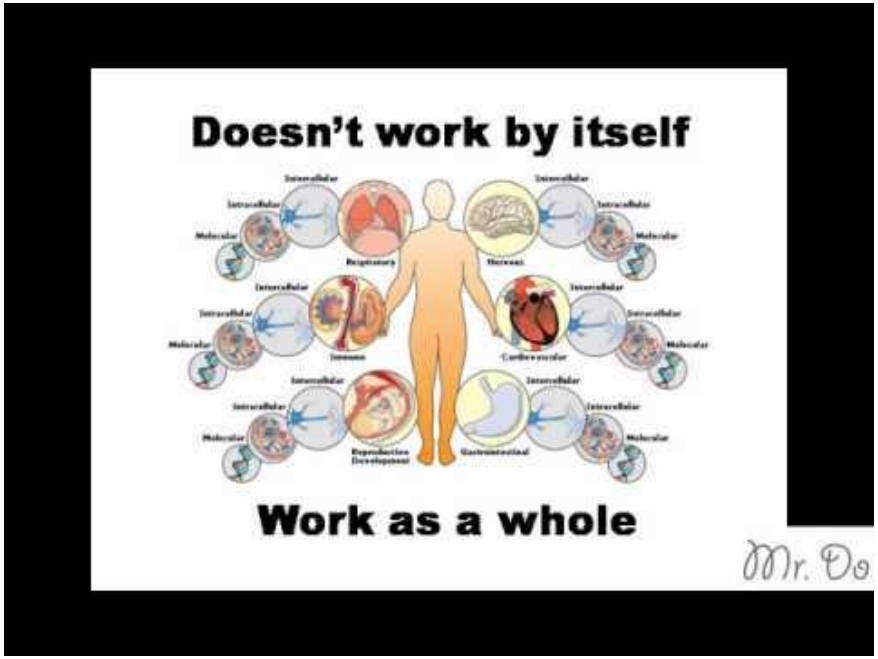
WHY IS ARCHITECTURE SO HARD

What is so hard about designing an architecture

- Architecture of any non-trivial system usually comes at different levels of abstraction
- There isn't a "right" answer
- Architectural qualities must be traded off against each other
- Architectures concern system, that have emergent properties
  - Not simple causal products
  - Unexpected interactions among the components
- There is so much going on that it is difficult to remember all of it and all of the interactions

Emergent properties

- All systems demonstrate emergent properties
  - Arise from interactions among its components
  - Most of the non-functional properties of a system are emergent
  - Some emergent properties are unexpected and unintended
  - Sometimes referred to as 2nd , 3rd, or Nth order effects
- We are OK with direct causal (1st order) reasoning
- We are seldom good at reasoning about 2nd and 3rd order effects



It gets better with experience

- Architecture is not something that can be learned from a text book or a lecture
- To become a proficient architect requires practice
- But there are patterns and precedent architectures that can be adapted to your problem

Summary

- Architecture is hard because you are dealing with systems that have emergent properties and behaviours
- There is so much going on that it is difficult to think of it all at once
- You can get better with practice
- There is an existing body of knowledge

UNIVERSITY OF TECHNOLOGY SYDNEY

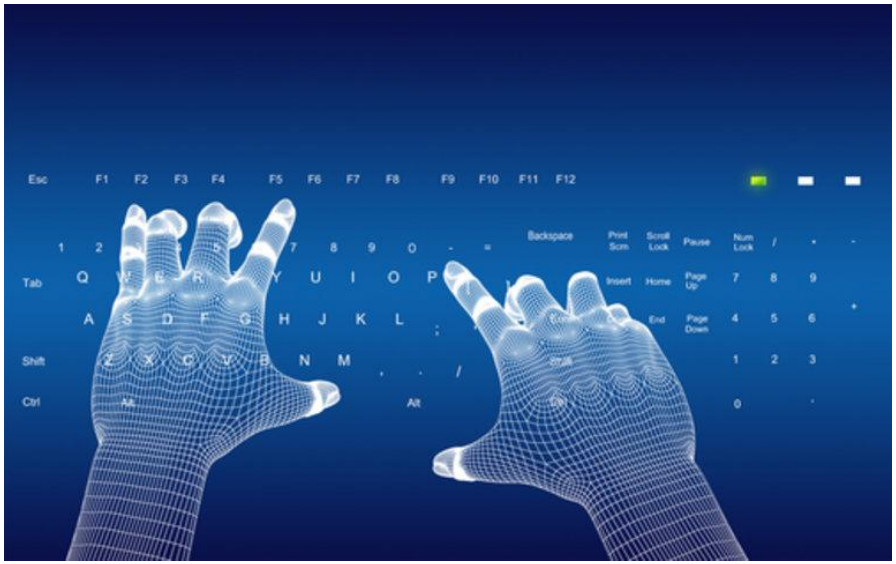
BECOMING A GOOD ARCHITECT

Secrets of great architects

- All great architects have mastered the ability to conceptualize a solution at distinct levels of abstraction.
- By organizing the solution into discrete levels, architects are able to focus on a single aspect of the solution while ignoring all remaining complexities. (Awalt, D. (2004), *Secrets of Great Architects, Available from http://msdn.microsoft.com/en-us/library/aa480041.aspx Last accessed August 2008*)

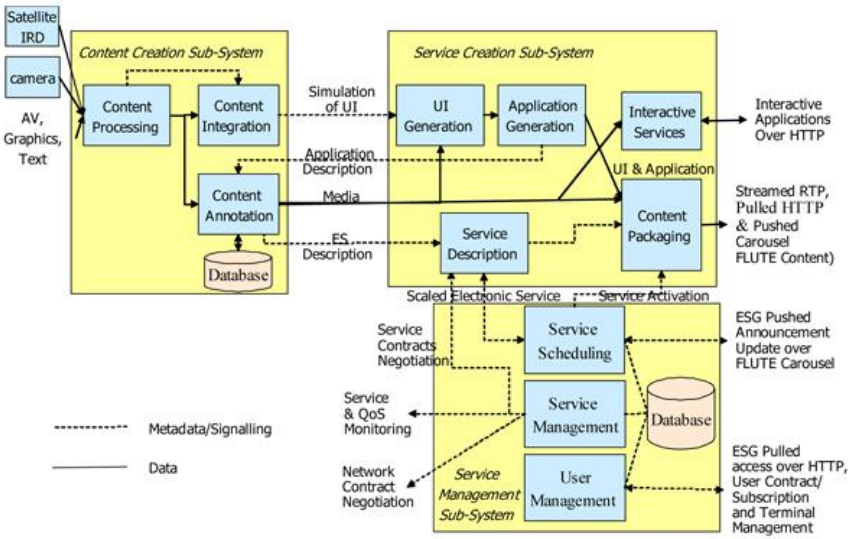
The rush into coding

- Most software developers know that they should decompose the solution into levels of abstraction. However, this is very difficult to apply in practice on actual projects.
- Upon encountering the first hurdle, it's easy to abandon the levels in the rush to start coding.
- Great architects work through the challenges and are disciplined to maintain the levels throughout the entire project lifecycle.
- They realize that if they don't, they'll eventually drown in the complexity.



Layers of abstraction

- System designers model computer systems using levels of abstraction.
- Each level is well defined and provides a distinct perspective of the system.
- Many systems are designed at three primary levels: system, subsystem, and component.
- Common across many engineering disciplines



Simple Framework: Four Levels of Abstraction

- Domain
  - Highest level of abstraction treats the system as a black box, considering only the external functions the system must provide
- Business process
  - The static view of the domain level models the class structure and their relationships of the objects witnessed in the use case.
- Logical
  - The logical level zooms inside the system black box, exposing the high-level design of the system. The architect selects the technology and defines the high-level system structure.
- Physical
  - The physical level of abstraction captures the structure of the system implementation.

Summary

- Great architects are able to think about the system at different levels of abstraction
- Too many software engineers rush into coding
- Layered abstraction is a common engineering technique
- Four levels of abstraction covers most software systems

COMMON ARCHITECTURAL CHALLENGES

Increased size and complexity of systems

- Boxes and arrows are not longer enough. They are not richly expressive nor sufficiently disciplined to communicate with the required precision.
- Increased need to incorporate legacy systems or packaged systems.
- Trend toward implementing systems as collaborations of loosely coupled components, many from external providers.
- Increased focus on product lines.

Modelling more complex problems

- Architecture can, and should, be used to model both the problem and its possible solutions.
- As organizations tackle less well understood and less well ordered problems, there is greater need to develop good models.
  - It is becoming too expensive to experiment on the real world.
  - Solutions are tried on models, then applied to the world.
  - This requires considerable skill in model development.