UNIVERSITY OF TECHNOLOGY SYDNEY

SOFTWARE ARCHITECTURE IN CONTEXT

Tom McBride
University of Technology Sydney

UTS: ENGINEERING AND INFORMATION TECHNOLOGY

Introduction

- Identify the context in which an architecture is developed
- The role of architecture
- Some examples of architecture

What do I need to know before I start?

What should the architecture achieve?

What constrains possible architectures?

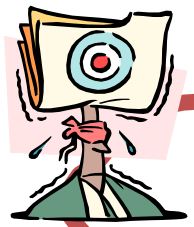
What determines a better architecture?

THE CONTEXT OF AN ARCHITECTURE

Solutions must fit the context

- A context is comprised of
  - The functional requirements
  - The goals (or objectives) of the stakeholders
  - Constraints
  - Risks and Opportunities

Stakeholders…

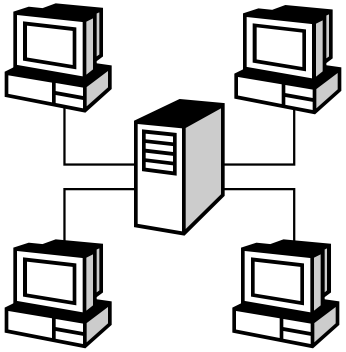


Architect

User

Regulators

Customer's customer



Product support

Developer

Customer

Supplier

Sysadmin

Stakeholder goals

- Identify the stakeholder goals systematically
  - Functional goals
  - System qualities
  - Technology goals
  - Organization capability goals

Constraints and enablers

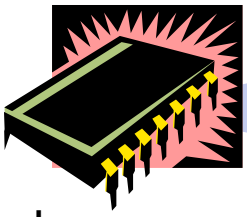
- Start with the obvious constraints
  - Time
  - Money
  - Resources
  - Expertise
  - Technology
  - Integration to existing systems
- What can this system take advantage of
  - Common goal across many potential customers
  - Technology
  - Organizational capability

Constraints and enablers…

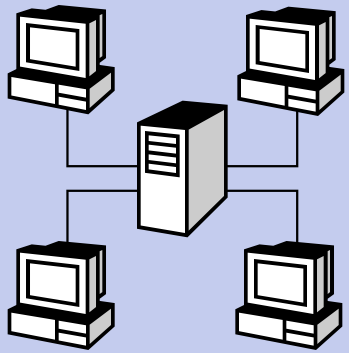
Operating system

Not many Similar systems

Lots of Java developers



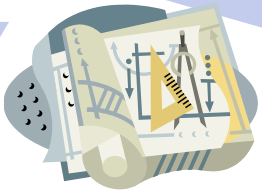
Faster processor



Regulations

Legacy systems

Standards



Policies

Ethics and environment

Risks

- Identify risks systematically
  - What are the risks that could threaten each goal of each stakeholder
- Persistent failure to learn from experience
  - All organizations have a collection of failure stories.
  - Use them to review “Could this happen again, on this project?”

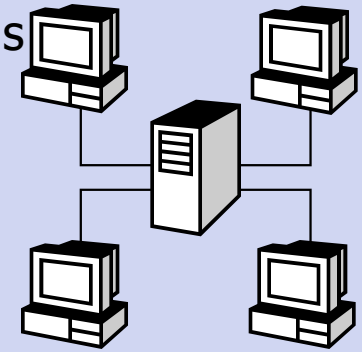
Opportunities and risks…

Sell a few systems

Tax issues

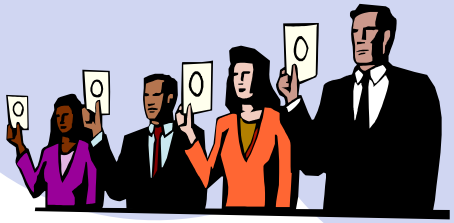
Misjudge the market

Sell a lot of systems



Build a reputation

Go to IPO

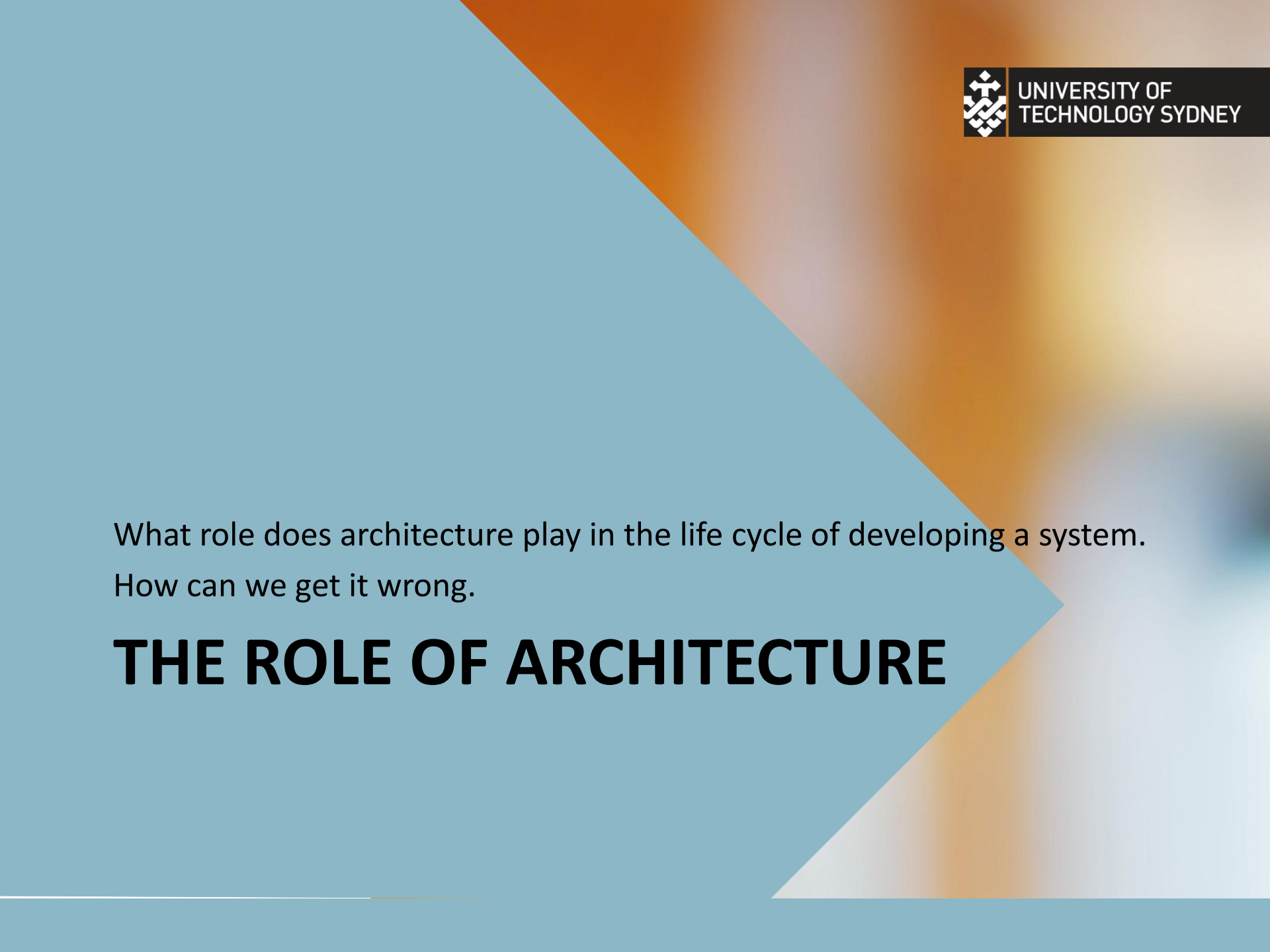


Deliver a poor performer

Late to market

Summary

- Start by identifying the stakeholders, constraints and enablers, risks and opportunities
- It’s complicated and you won’t have all the answers right away
- But the context guides development from inception to operation and disposal

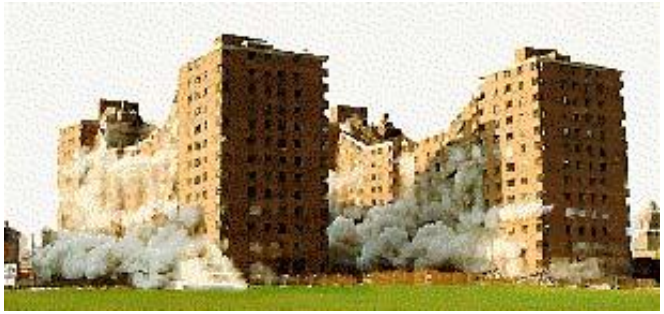


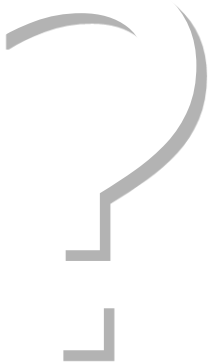
What role does architecture play in the life cycle of developing a system.

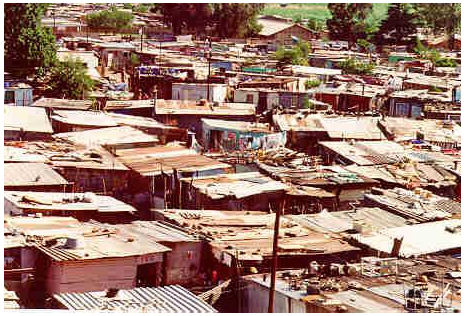
How can we get it wrong.

THE ROLE OF ARCHITECTURE

What is the role of architecture?



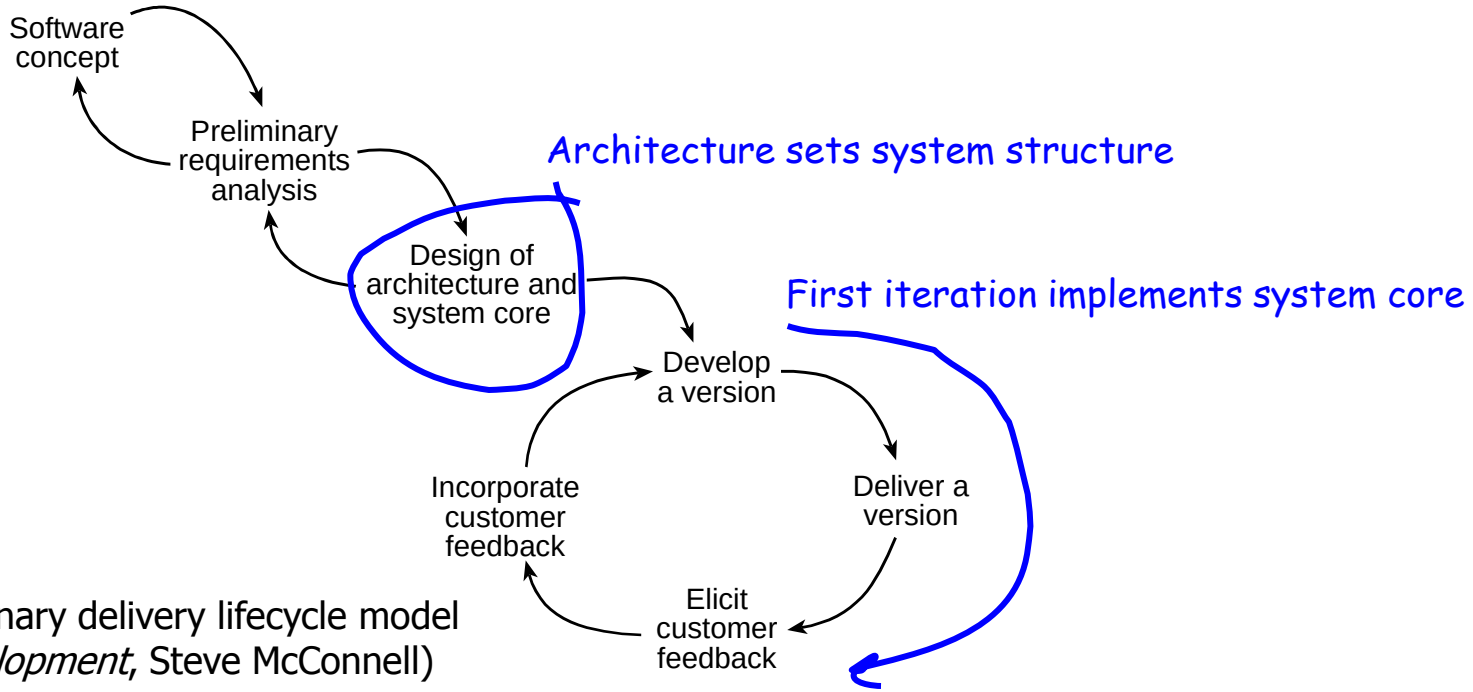




Leaning tower image from Gary Feuerstein.
Other images from *The Big Ball of Mud*, by Yoder and Foote.

Fiji Architecture

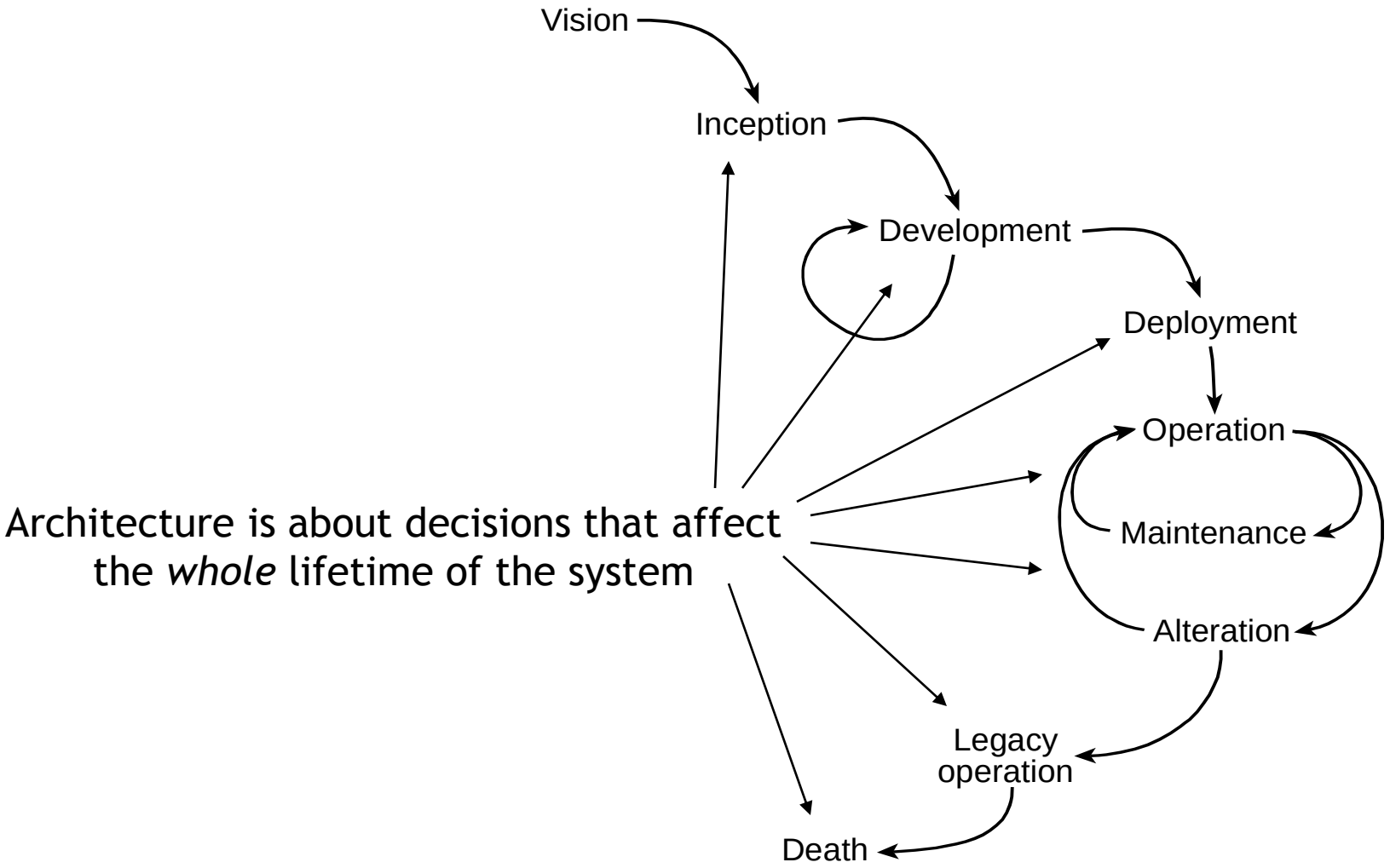
Development lifecycle



The evolutionary delivery lifecycle model (*Rapid Development*, Steve McConnell)

Architecture plays a vital role in establishing the structure of the system, early in the development lifecycle. It's where you make the big mistakes, and the biggest of them is thinking you can work it out as you go

System lifetime

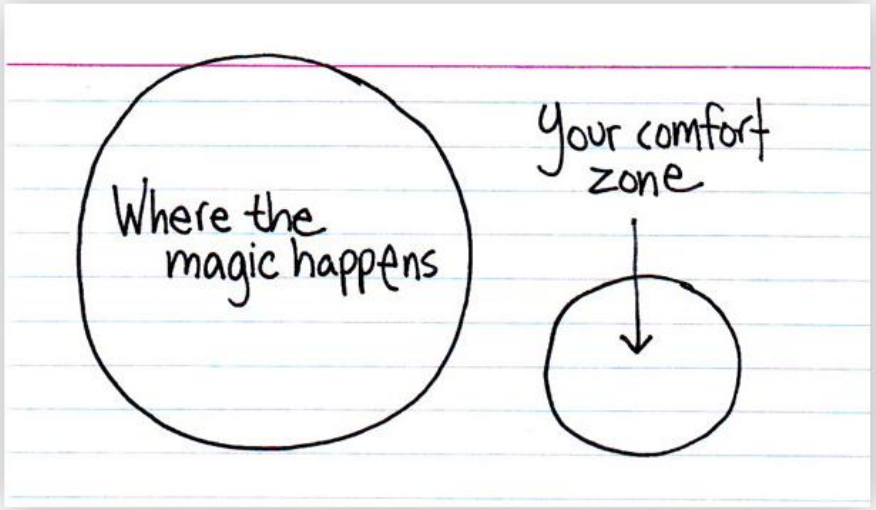


Where software spends most of its life

- In maintenance
- You have built it, and it works.
  - Then you are asked to speed it up from 10 transactions per second to 200
  - Didn't design it to run on separate processors
  - If you didn't allow for scaling up, you are in trouble
- You have built it and it runs
  - Until it strikes that combination of circumstances that break it.
  - Now you need to change some part of it
  - If it wasn't designed to for modifying or testing, you will now have to pay the price of that

Where we get into trouble

- Confuse program design with software architecture
- Staying in the comfort zone of “programmer”
- Hackers program, software engineers design architecture
- Wanting to work it out as you go (muddle through)
- Encouraged by agile methods
  - Be warned – agile is not for beginners
- Our creations are more complex than we can imagine, require more precision than we are accustomed to, more complex than we can reason about without some form of external representation



Summary

- Architecture lasts a long time
- Software systems spend most of their life being modified
- We get into trouble when we treat architecture as something that stops us cutting cool code

UNIVERSITY OF TECHNOLOGY SYDNEY

SOME EXAMPLES OF ARCHITECTURE

Let’s look at some examples…

- Aside: In engineering, we work with models a lot. A model is a representation that abstracts from inessential details, and can be manipulated in ways that the “real thing” cannot.

Web bulletin board

- 3 Tier architecture common in web application systems
- Security
- Maintainability

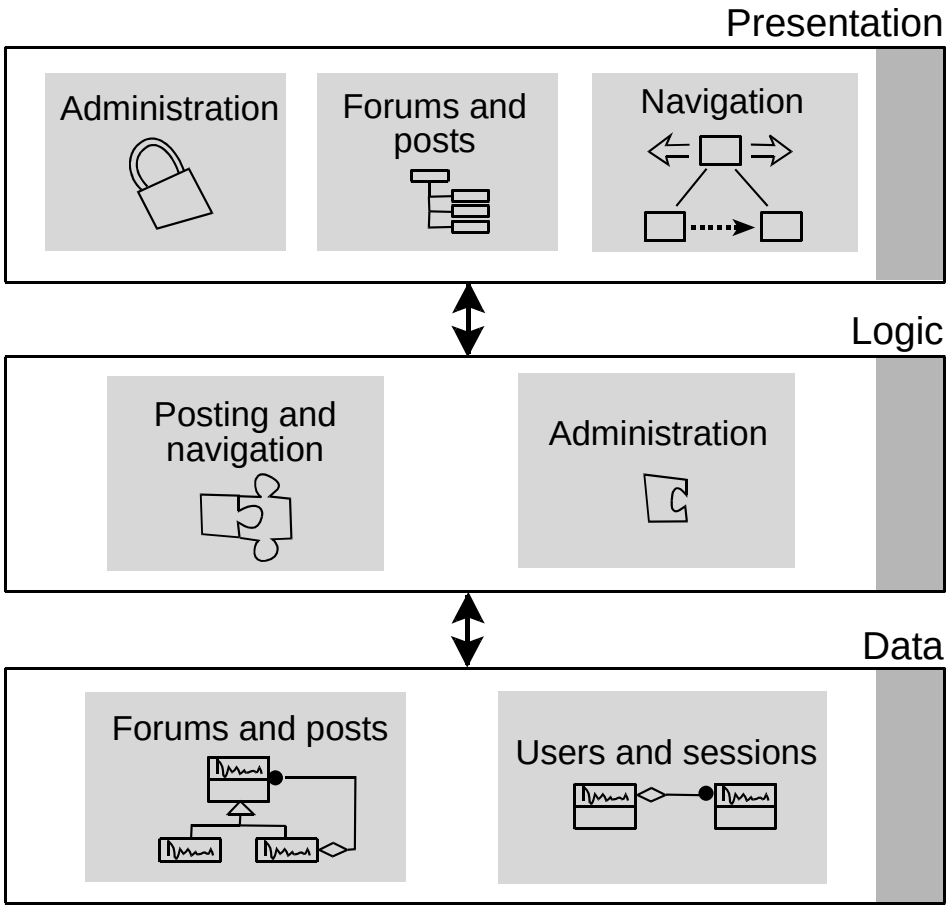
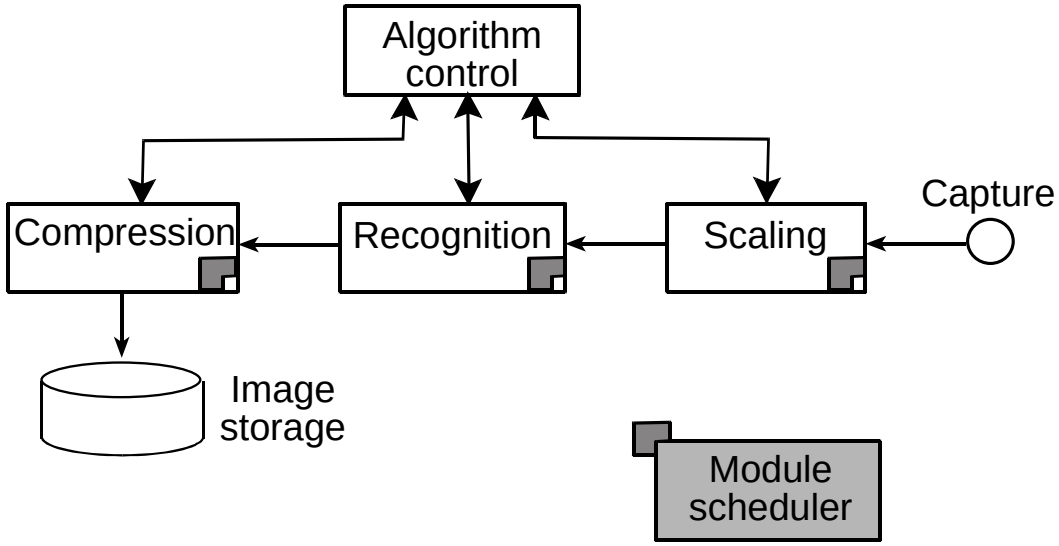
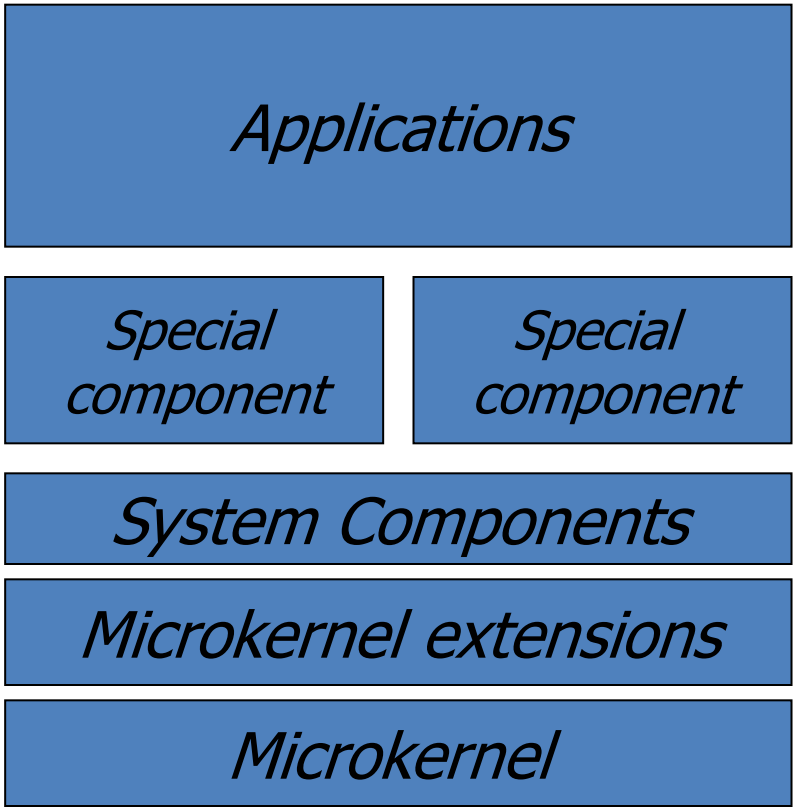


Image processing

- Performance
- Configurability
- A processing “pipeline” is commonly used in realtime and embedded systems



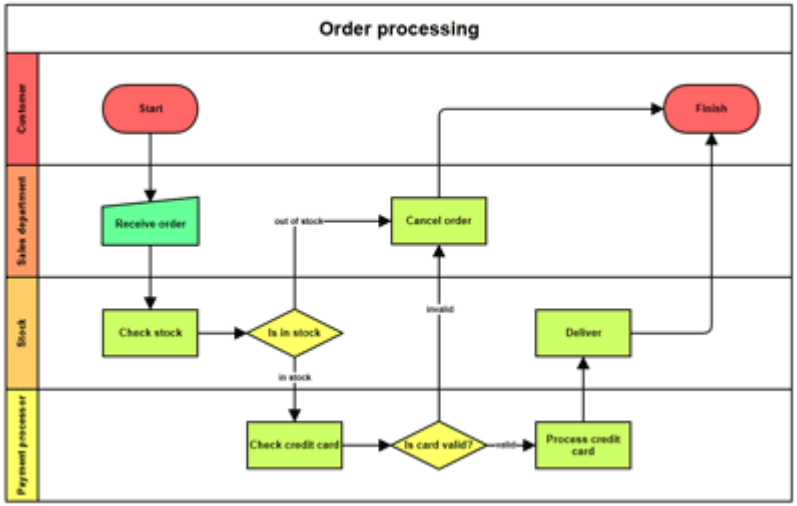
Embedded systems



- Embedded systems have become less constrained by limited resources and processing power, and more connected to other systems so that there is less and less difference between embedded systems and any other system.
- Autonomous robots, autonomous cars, control systems

Workflow management systems

- System as an active participant in the work.
- Orchestrates the work.
- B2B systems, single organization systems.



Summary

- An architecture is a model of what we want to build
- Models are useful to explore the problem and its intended solution
- There are some well established architectures
- More specific architectures in “Patterns”