

## SENG4420: Software Architecture

School of Electrical Engineering and  
Computer Science

---

---

---

---

---

---

---

---

### Text/Reference books

- Hofmeister, C., Nord, R., and Soni D. (2000):  
*Applied Software Architecture*, Addison-  
Wesley, ISBN 0-201-32571-3
- Bass, L., Clements, P. and Kazman, R (1998):  
*Software Architecture in Practice*, Addison-  
Wesley, 0-201-19930-0
- Fenton N. and Pfleeger, S. L. (1997):  
*Software Metrics – A Rigorous and Practical  
Approach*, 2<sup>nd</sup> Edition, PWS Publishing Co.,  
ISBN 0-534-95425-1

2

---

---

---

---

---

---

---

---

### Lecture Topics

- Basic concepts of software architecture
- Architecture design: global analysis
- Architecture design: conceptual view
- Architecture design: module view
- Architectural styles
- Architecture analysis
- Case studies
- Introduction to software metrics
- Software size
- Measurement for software product

3

---

---

---

---

---

---

---

---

## Assessment

- Assignment 1: Essay/Presentation
  - 20%
- Assignment 2: Architecture project
  - 40%
- Final examination
  - 40%
- You must achieve at least 40% of the final exam and an overall final mark of 50% or more to be eligible to pass the course
- Late submissions lose marks at the rate of 5% per calendar day. As usual, copying, plagiarism and other malpractices are not allowed in assignments and examinations.
- A completed and signed Assignment Cover Sheets (ACS) or a completed electronic ACS must accompany all assignment submissions. If the submission is electronic it is deemed signed. (ACS can be found at <http://www.eng.newcastle.edu.au/eecs/current/index.html>).

4

---

---

---

---

---

---

---

---

## Introduction to Software Architecture

---

---

---

---

---

---

---

---

## What is Software Architecture?

- Software Architecture of a program or a software system is the structure or structures of the system, which comprise
  - software *components*,
  - the externally *visible properties* of those components, and
  - the *relationships* among them
    - (Bass *et al.*, 1998).

Visible properties refer to the assumptions other components can make about a component's services, performance characteristics, fault handling, shared resource usage etc.

6

---

---

---

---

---

---

---

---

## What is Software Architecture? (Another definition)

- Software Architecture is
  - a collection of computational *components* that are
  - *shared* across a series of products or systems, together with
  - a description of the *interactions* between the components, together with
  - *constraints* on how they can be combined.

7

---

---

---

---

---

---

---

---

## What is a component?

- From an architecture view point
  - A component is an entity with a well-defined interface
    - It could be an object (specified as an OO class), a package (collection of objects), a database, a process, a library, or even another software system.

8

---

---

---

---

---

---

---

---

## Influential factors

- For the same requirements two architects may design two different architectures
  - Architecture is not *necessarily* tied to the requirements
- Other influences
  - Technical
    - Example: Development platform available (PC's vs. Unix)
  - Social
    - Example: A development house with several client-server architecture experts
  - Business
    - Example: Defence industry's concern on security

9

---

---

---

---

---

---

---

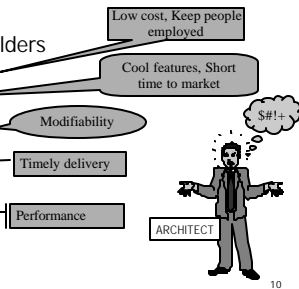
---

## Stakeholders in the system

- Architectures are influenced by stakeholders in the system

- Typical stakeholders

- Management
- Marketing
- Maintenance
- Customer
- End user



10

---

---

---

---

---

---

---

---

## Trade-offs

- Architecture is the earliest artifact in a project that allows competing concerns to be analysed

- Performance vs. Reliability
- Cost of current project vs. Long-term cost of software development
  - Example: Design for reuse

- Some of these concerns may be available in the requirements specification

11

---

---

---

---

---

---

---

---

## Managing stakeholders

- *An architect must*

- *identify and actively engage the stakeholders, and*
- *solicit and manage their needs and expectations*

- Use

- Architecture reviews
- Iterative prototyping

12

---

---

---

---

---

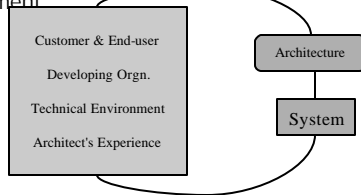
---

---

---

## Architecture Business Cycle (ABC)

- The environment influences the architecture
- The architecture influences the environment



13

---

---

---

---

---

---

---

---

## Architecture's influence on its environment

- Determines organization structure
  - Organizational setup matching the architecture
    - Organizational units to implement software units
- Redefines enterprise goals of the organization
  - A new successful product may lead to efficient and profitable development of similar products
- Affects customers
  - New requirements for updates
- New experience for the architect

14

---

---

---

---

---

---

---

---

- Pathfinder systems pass on their legacy to subsequent applications
  - Relational databases
  - Table driven operating systems
  - Graphical User Interfaces
  - World Wide Web

15

---

---

---

---

---

---

---

---

## Architecture-based process steps

- Creating the business-case
  - If architect is not involved, the product is not likely to align with business goals
- Requirements elicitation
- Designing or selecting the architecture
- Communicating the architecture to stakeholders
- Evaluating the architecture
- Implementing the architecture and ensuring conformance

16

---

---

---

---

---

---

---

---

## Requirements elicitation

- From customers and end-users
- Use-cases or scenarios
  - If using Object Orientation
- Finite state machine models, Formal Specifications
  - In safety -critical systems
- Domain analysis
  - Understanding the characteristics of the prior system (human and computer based)
  - Prototypes
- The desired qualities of the system influences its architecture

17

---

---

---

---

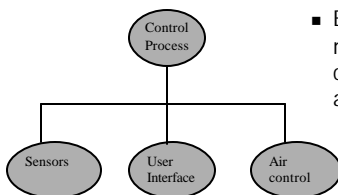
---

---

---

---

## Is this an architecture?



- It has components and links.
- But we need a lot more information to consider it as an architecture

18

---

---

---

---

---

---

---

---

## Unanswered questions

- What is the nature of the components?
  - Do they run on separate processors?
  - Are they processes?
  - Are they design-time components, or just run-time separation?
- What is the significance of the links?
  - Does the link mean communication, control, or synchronization?
- What is the significance of the layout?
  - Why is Control Process above the other three?
  - Can any component call any other component?

19

---

---

---

---

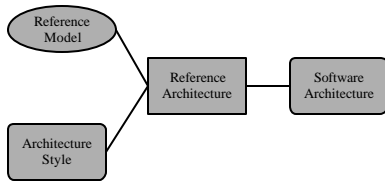
---

---

---

---

## Architecture concepts



20

---

---

---

---

---

---

---

---

## Architecture Style

- An architecture style is a description of
  - component types, and a
  - pattern of their runtime control and/or data transfer
- Think of it as a set of constraints on the architecture elements (components, links etc)
  - which defines a family of architectures that satisfy the constraints
- [More architecture styles comes later]

21

---

---

---

---

---

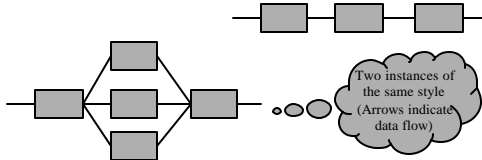
---

---

---

## Example of a style: Pipeline

- Pipeline architectures
  - The output of one component is the input of another component
  - Computation involves transformations on streams of data



22

---

---

---

---

---

---

---

---

## Reference Model

- It is a division of functionality together with data flow between the pieces
- Example
  - A compiler's functionality can be expressed as:
    - lexical analysis, syntactic analysis, code generation etc.
  - The data flow between these steps are also well understood by compiler writers:
    - lexical analysis results in tokens
    - syntactic analysis uses a parse tree, etc.
- The existence of a reference model shows the maturity of a domain.

23

---

---

---

---

---

---

---

---

## Reference Architectures

- A reference architecture is a reference model mapped onto software components (that will implement the functionality) and links that will implement the data flow
  - The mapping need not be one to one.
- Example:
  - The compiler reference model can be mapped on to a pipeline architectural style to get the traditional reference architecture for compilers

24

---

---

---

---

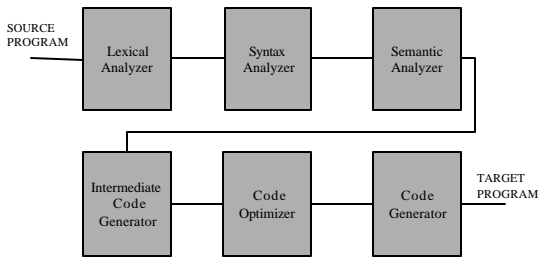
---

---

---

---

## Reference architecture for compilers



25

---

---

---

---

---

---

---

---

## Software architecture is important

- Communication among stakeholders
  - Software architecture represents a common high-level abstraction of a system and can be used as a vehicle for communication
- Early design decision
  - Defines constraints on an implementation
  - Inhibits or enables a system's quality attributes
  - Helps in evolutionary prototyping
  - Dictate organizational structure
- Transferable abstraction of a system
  - Product lines share a common architecture
  - Systems can be built by externally developed components

26

---

---

---

---

---

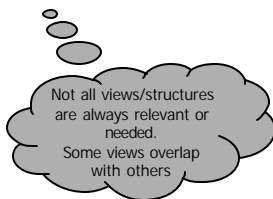
---

---

---

## Architectural structures

- Also known as architectural views
  - Conceptual or logical structure
  - Module (or subsystem) structure
  - Process or coordination structure
  - Physical structure
  - Uses structure
  - Calls structure
  - Data flow
  - Control flow
  - Class structure
  - Execution structure
  - Code structure



27

---

---

---

---

---

---

---

---

## Conceptual or logical view

- The units are abstractions of the system's functional requirements
  - Reference model, for example, is an example of the conceptual structure
- Linked by shares-data-with relation
- Useful for understanding the interactions between entities in the problem space
- In many systems (often not the huge ones), the module structure may reflect the conceptual structure

28

---

---

---

---

---

---

---

---

## Module view

- The units are work assignments
- Have products (such as interface specification, code, test plans) associated with it
- Linked by is-a-subsystem-of relation
- Useful for allocating labor and other resources

29

---

---

---

---

---

---

---

---

## Process or coordination view

- The units are processes or threads
- Discusses the running system
- Linked by synchronizes-with, can't-run-with, waits-for etc. relations
- Relevant for concurrent and real-time systems

30

---

---

---

---

---

---

---

---

## Physical view

- Mapping of software onto hardware
- Units are processors
- Linked by communication paths
- Relevant for distributed or parallel systems

31

---

---

---

---

---

---

---

---

## Uses view

- Units are modules or procedures
- Linked by the assumes-the-correct-presence-of-relation
- Helps in incremental build
  - Construct the used modules, then the using modules
  - If A uses B, and B uses C, build C, then B, then A.
- Engineer the Uses-Structure carefully for building extensible systems

32

---

---

---

---

---

---

---

---

## Calls view

- Units are procedures or modules
- Linked by calls relation
- Used to trace flow of execution
  - and reduce bottlenecks

33

---

---

---

---

---

---

---

---

## Data flow view

- Units are modules
- Linked by may-send-data-to relation
- The link is labelled with the data transmitted
- Useful for requirements traceability

34

---

---

---

---

---

---

---

---

## Control flow view

- Units are programs, modules or system states
- The link is becomes-active-after
- Useful for verifying
  - the functional behaviour of the system
  - timing properties
- Identical to the calls structure
  - if control transfer is only through procedure calls

35

---

---

---

---

---

---

---

---

## Class view

- Units are classes (in object technology)
- The link is inherits-from
- Supports reasoning about collections of similar behaviour

36

---

---

---

---

---

---

---

---

## Execution view

- Sequencing of executable objects
- Allocation of functional components to runtime entities
- Handling communication, coordination and synchronization
- Runtime platform
- System performance, resource usage, load balancing
- Flow of control at runtime
  
- Overlaps with Process, Control flow and Physical views

37

---

---

---

---

---

---

---

---

## Code view

- The organization of code into:
  - files
  - libraries
  - object code
  - binaries
  
- versions
  - Configuration management

38

---

---

---

---

---

---

---

---

## Relating views to each other

- Each of these views provides a different perspective and design handle on a system
  - The conceptual and module views emphasize static properties but address different concerns
  - The process and physical views gives us the dynamic or runtime system structure
- These views are not independent
  - Elements of one view may be connected to elements of other views
  - Individual projects tend to consider one view dominant and cast other views
- Changes in one view may influence the other views
  - A change to the control view of a system may influence the process and physical views

39

---

---

---

---

---

---

---

---