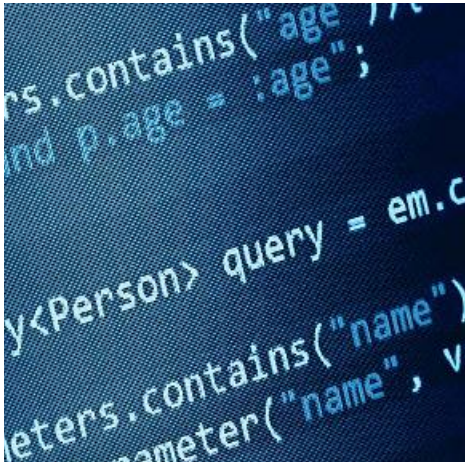
CS3500 Software Engineering

2017/2018



Dept. Computer Science
Dr. Klaas-Jan Stol

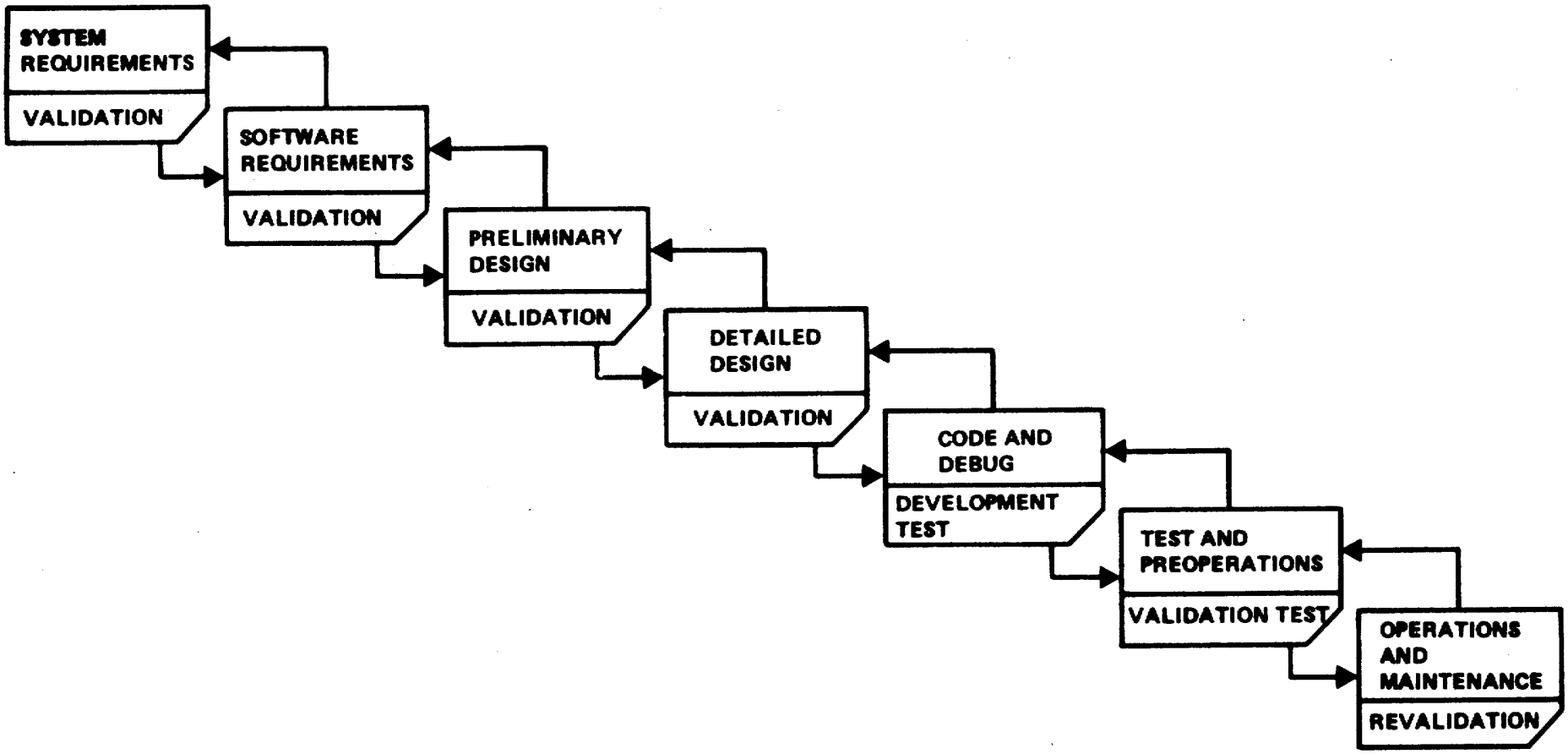
UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Welcome to CS3500

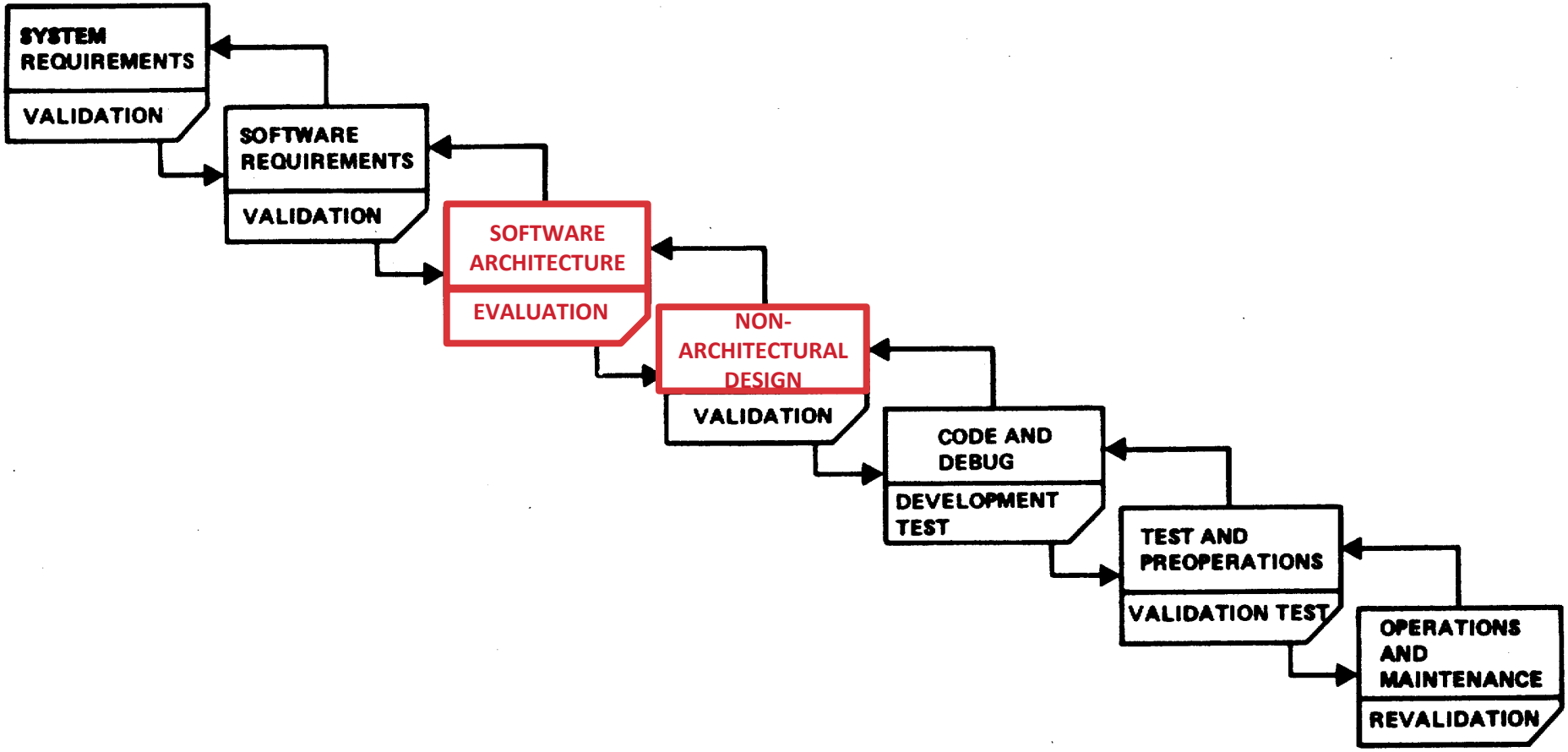
Software Architecture

Part I

Where are we now?



Scope of this lecture



Modern terminology:

- Preliminary Design → Software Architecture
- Detailed design → Non-architectural design

On metaphors

- A metaphor is a figure of speech in which a word or phrase is applied to an object or action to which it is not literally applicable.

- A thing regarded as representative or symbolic of something else.

Metaphors in software development

- **Waterfall model**
  Top-down development process
- **Building software**
  Developing and/or compiling software
- **Cloud**
  The Internet

Civil engineering v. software architecture

	Construction	Software
Stakeholders	Builders, carpenters, electricians, plumbers, house buyers, etc.	Developers, customers, end-users
Views	Plumbing, electrical wiring, walls & windows	Logical, deployment, use-case view, dynamic view, etc.
Who's responsible for design?	Architect	Software architect
What's happening?	Building, constructing	Building, constructing (software)
Components & connectors	Pre-fab walls, doors, windows, bricks, cement, etc.	Database management systems, middleware, frameworks, "glue code"
Design output	Architecture design	(Software) architecture design
Who's building it?	Builders	Developers

Architectural styles

Contents

SECTION I

What is Software Architecture?

Software architecture emerged in the 1990s as an important topic within software engineering.

Developing an intuition for SA

Consider the architecture of processors

- CISC:
  Complex Instruction Set Computers
  Many highly specialized instructions

- RISC:
  Reduced Instruction Set Computers
  Limited set of simple instructions

Developing an intuition for SA

Multiplying in CISC vs. RISC

CISC:

```
MULT 2:3, 5:2
```

RISC:

```
LOAD A, 2:3
LOAD B, 5:2
PROD A, B
STORE  2:3, A
```

Developing an intuition for SA

	CISC	RISC
	Emphasis on hardware	Emphasis on software
	Includes multi-clock complex instructions	Single-clock reduced instruction only
	Memory-to-memory: “LOAD” and “STORE” incorporated in instruction	Register to register: “LOAD” and “STORE” independent (separate) instructions
	Small code size, high cycles per second	Low cycles per second, large code size
	Transistors used for storing complex instructions	Spends more transistors on memory registers

The choice between CISC and RISC depends on trade-offs.

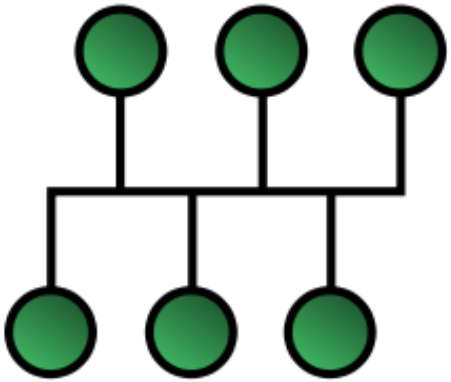
Adapted from: https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/

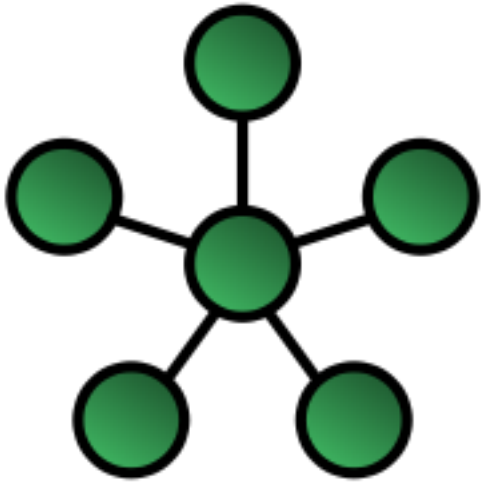
Developing an intuition for SA

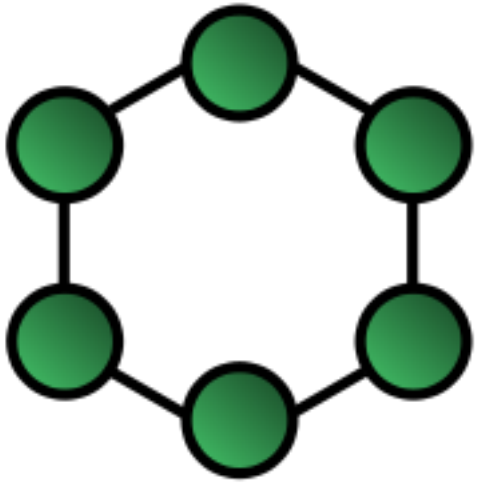
Consider the architecture of networks

- Components:
  nodes, connections
- Topology:
  Star, Ring, Bus, and others

Developing an intuition for SA

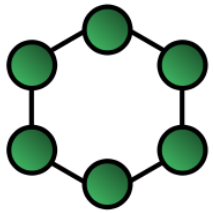


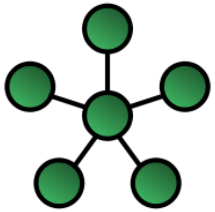


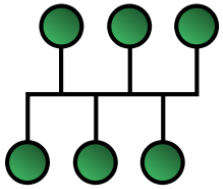


Images from Wikipedia.

Developing an intuition for SA







	Ring	Star	Bus
	+ Better performance at high network loads than bus topology	+ Easy to implement	+ Inexpensive (single wire)
	+ No need for network server	+ Simple to extend	- High cost of managing network
	- Weakest link between 2 nodes is bottleneck	- Central hub is single point of failure	- Single point of failure (1 cable)

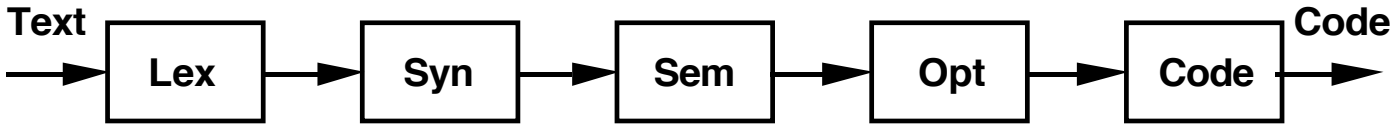
The choice between Ring, Star, and Bus topologies depends on trade-offs.

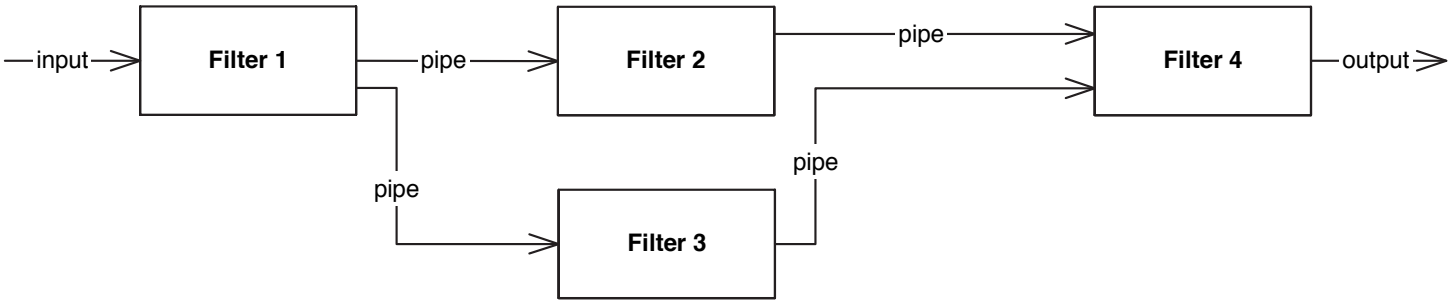
Design Decisions & Trade-offs

- Choice between CISC and RISC architectures is a (hardware) architectural design decision
- Choice between Star, Ring, Bus topology is a (network) architectural design decision
- Choice informed by trade-offs: the good and the bad, the pros and cons.
- Your decision will have consequences for the overall system's quality attributes.

EXAMPLE

Architectural style of a compiler
Pipes & Filters





Command on Unix/Linux:

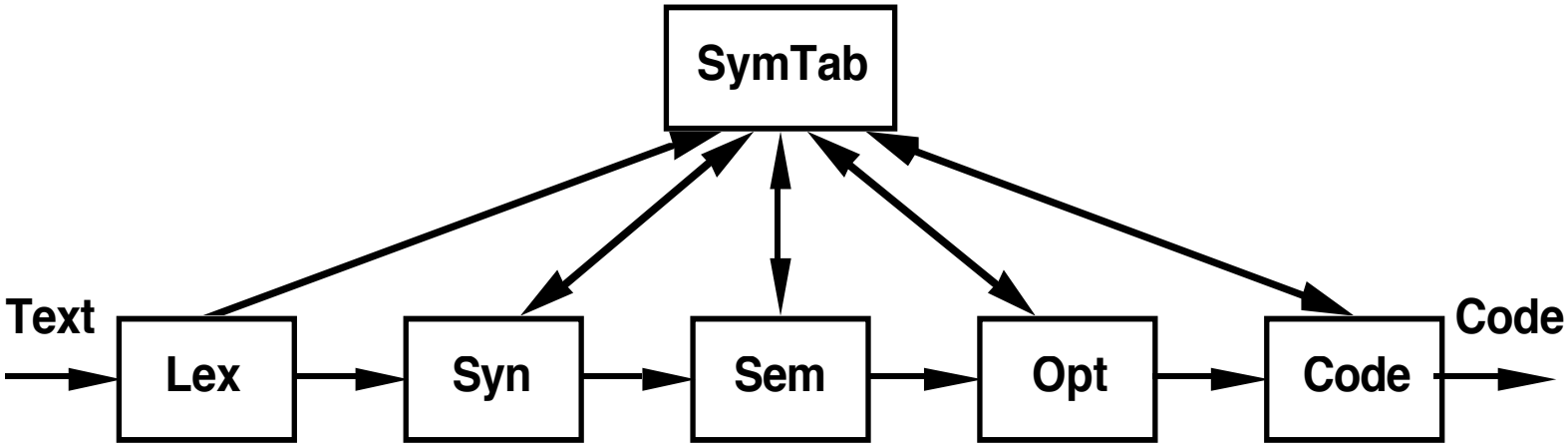
```
$ filter1 | filter2 | filter 3
```


Images from Garlan & Shaw, Avgeriou & Zdun

EXAMPLE

Architecture of a compiler

Evolution from the Pipes & Filters architecture

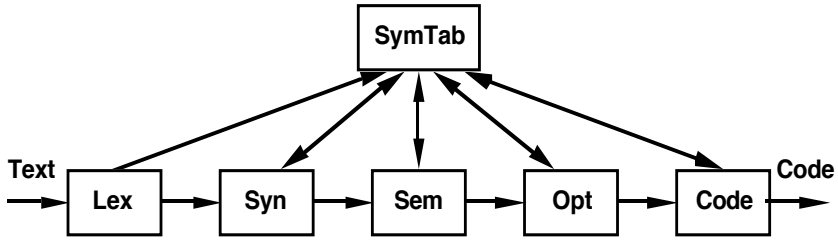


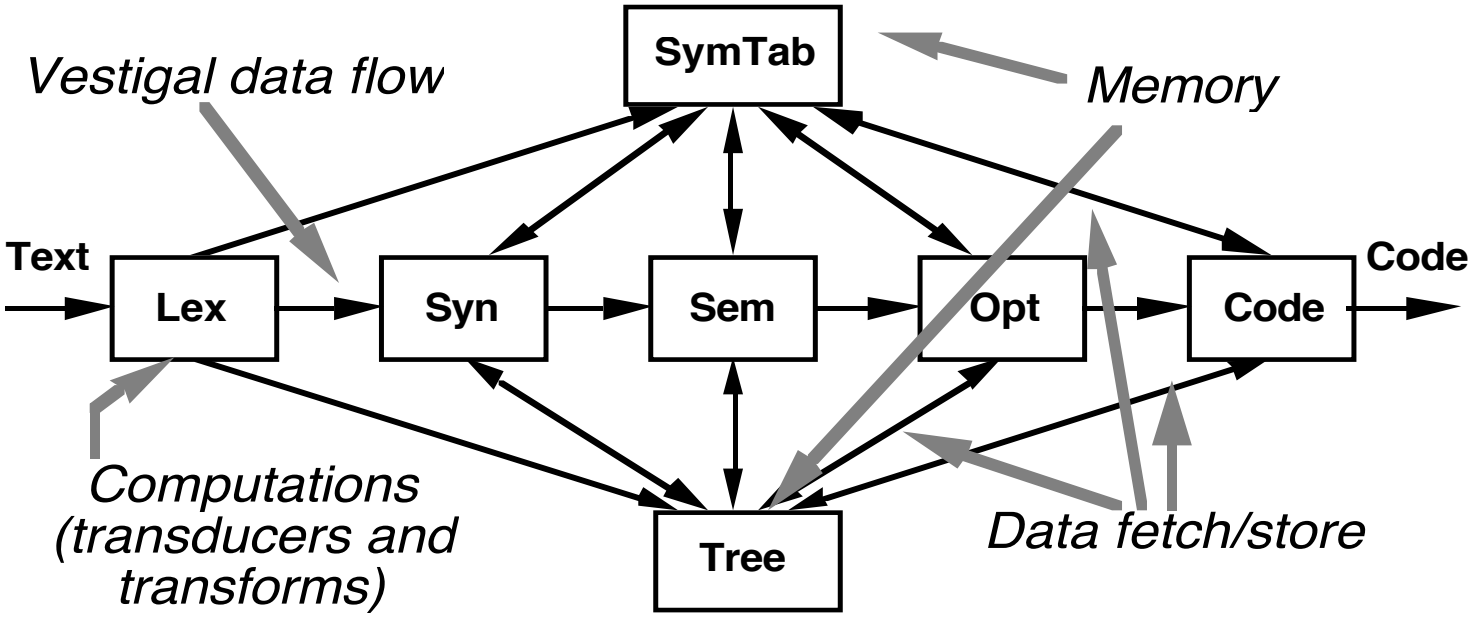
Images from Garlan & Shaw

EXAMPLE

Architecture of a compiler

Evolution from the Pipes & Filters architecture



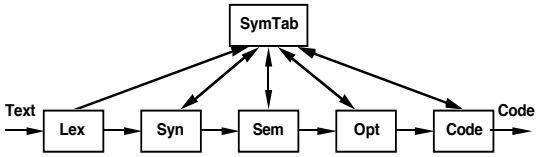


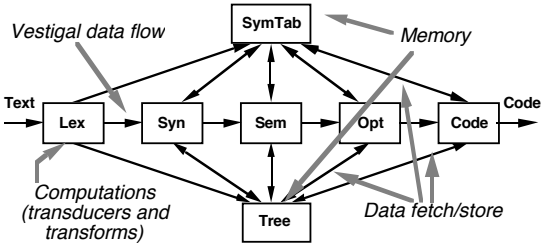
Images from Garlan & Shaw

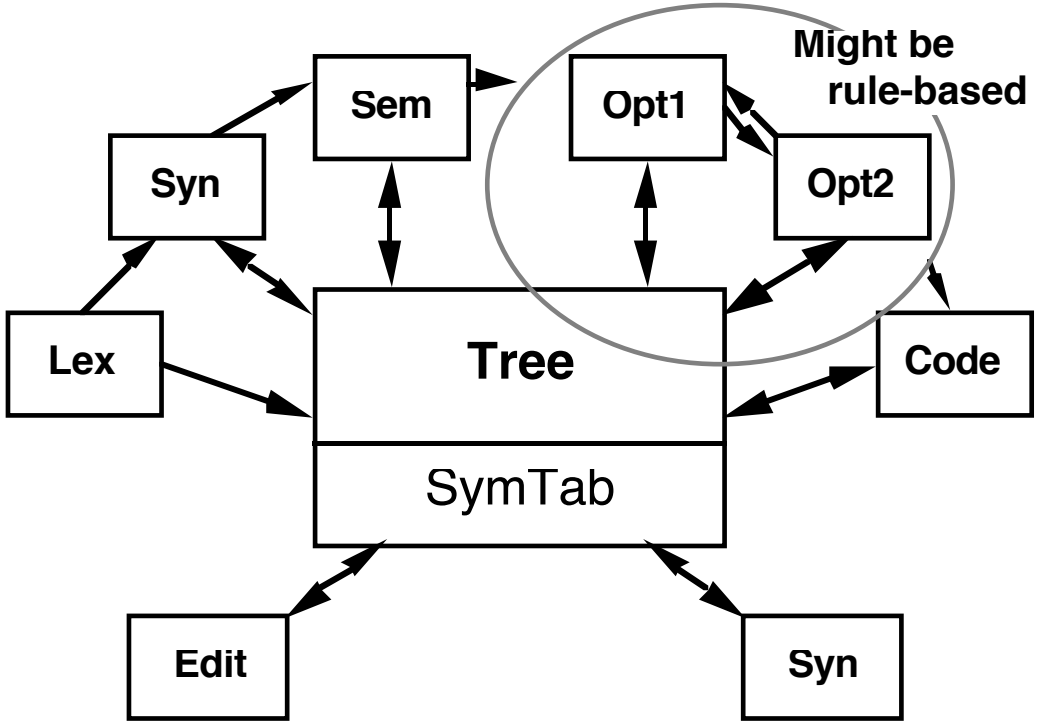
EXAMPLE

Architecture of a compiler

Evolution from the Pipes & Filters architecture



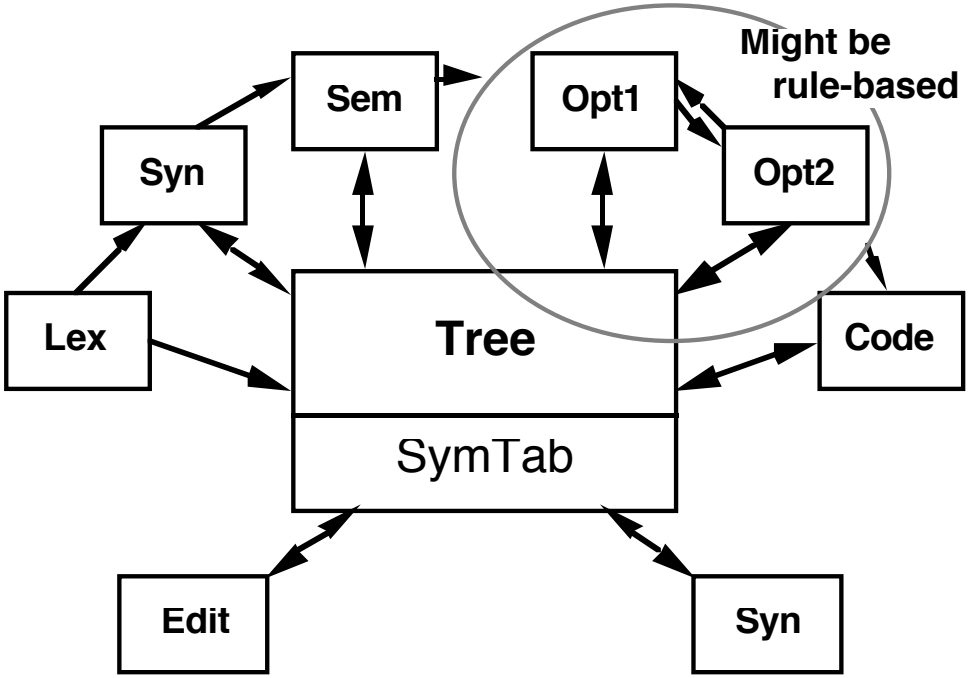


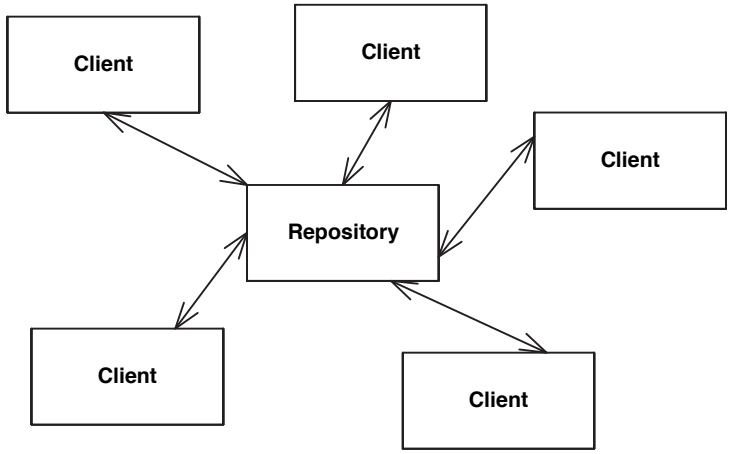


Images from Garlan & Shaw

EXAMPLE

Architecture of a compiler
Repository architecture





Images from Garlan & Shaw, Avgeriou & Zdun

Architectural styles and trade-offs

- Similar to processor and network architectures, software architectures are selected based on trade-offs.

- Selected software architectural style has consequences for its quality attributes.
  - Performance
  - Security
  - Modularity
  - Portability
  - Etc.

Common architectural styles

- Client-server
- Peer-to-peer (p2p)
- Pipes & Filters
- Layers
- Repository
- Service-oriented architecture (SOA)

SECTION II

Definitions

1. Architecture

2. Architectural drivers

3. Architectural Significant Requirements

4. Quality Attribute

5. Architectural significant

Definition 1

Software Architecture (SA):

The structure(s) of a system comprising software elements, externally visible properties of those elements, and the relationships among them.

Software elements, Properties, Relationships

Definition 2

Software Architecture (SA):

Software Architecture
= {Elements, Form, Rationale}

—Perry & Wolf

Form refers to composition style, and rationale provides the justification of design decisions.

3 Levels of Software Design

1. Architecture
   - System capability & Quality attributes
   - Composition of subsystems
   - Components & Connectors

2. Code
   - Algorithms & data structures
   - Programming languages
   - Pointers, arrays, procedures

3. Executable
   - Memory maps, data layouts
   - Call stacks, Register allocations
   - Bits, words, operators

Definitions

Architectural drivers:

The set of Quality Attributes (QA) that shape a system's software architecture.

Definitions

Architectural Significant Requirement (ASR)

A requirement that has profound impact on a system's architecture, and which represents significant value to at least one stakeholder.

Definitions

Module

A unit of implementation.

Examples include (sets of) source files, XML and configuration files, BNF specifications for parsers.

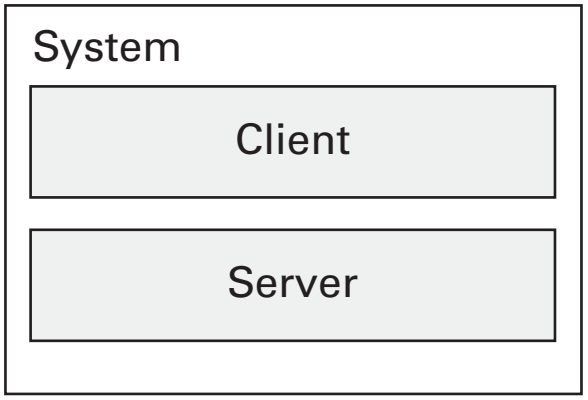
Definitions

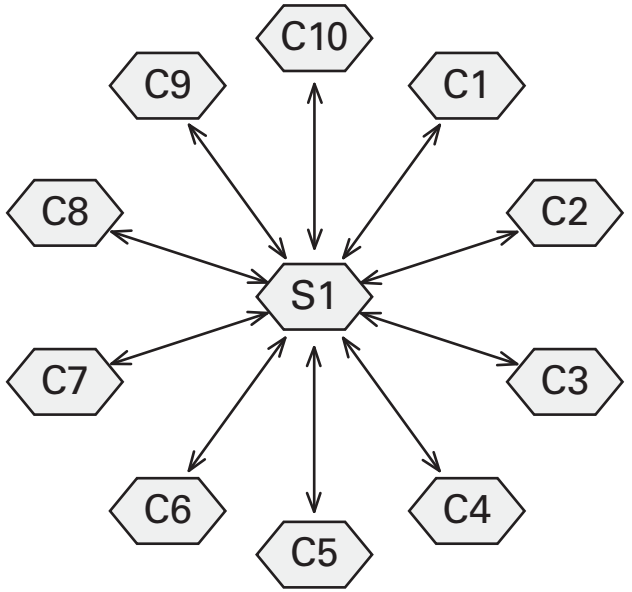
Component

A runtime entity that can be deployed independently.

Module vs. Component

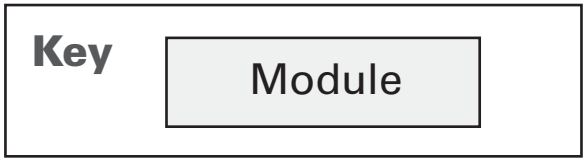
2 modules, 11 component

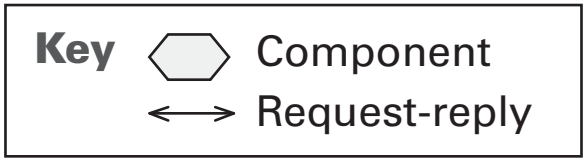




Decomposition view

Client-server view





Definitions

Architectural style

A specialization of elements and relation types, together with a set of constraints on how they can be used.

Architectural style and architectural pattern are effectively the same thing.

Next time on CS3500 Software Engineering…

SECTION III

Documenting Software Architectures

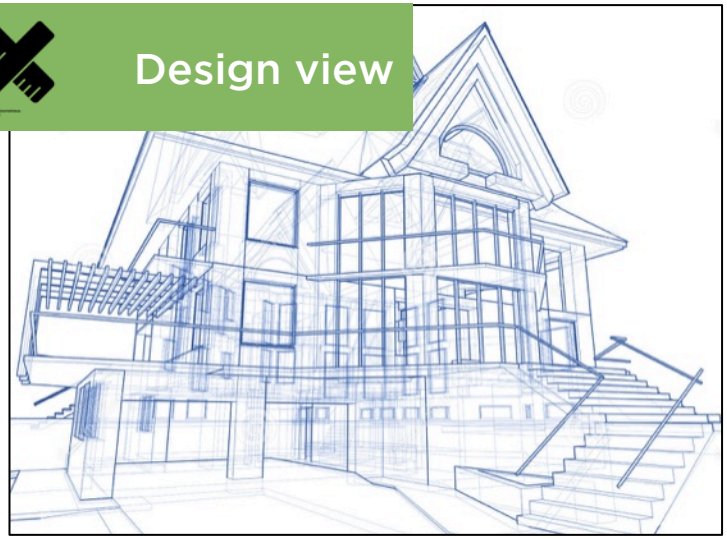
1. Architecture views

Different stakeholders, different views

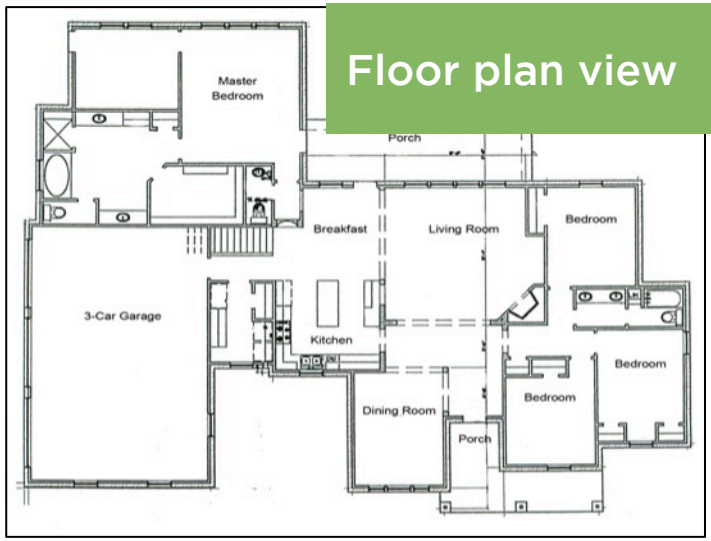
- In building construction, different stakeholders have different concerns
- Each stakeholder has different views on the project:
  - An architect focuses on design and integrity
  - An electrician focuses on wiring and sockets
  - A plumber focuses pipes
  - Builders construct walls and roofs
  - *... etc.*

Civil engineering as a metaphor for SA

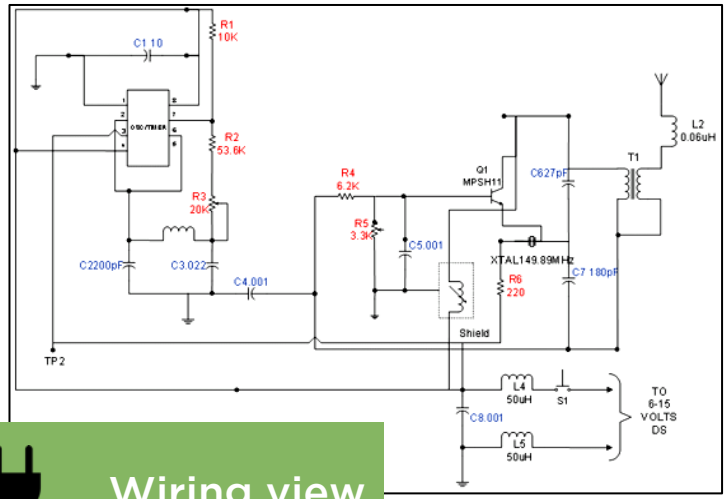
Design view



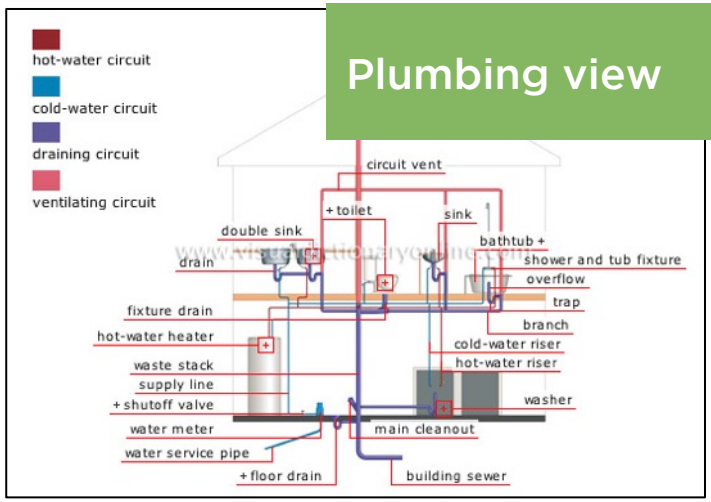
Floor plan view



Wiring view



Plumbing view



Architectural views

- Different models to capture architectural views:
  - Kruchten's 4+1 views model
  - Siemens Four View model (S4V)
  - Rozanski & Woods' model
  - Philips' CAFCR model
  - IEEE 1471-2000 standard
  - Clements et al.'s Views and Beyond approach

- Sets of views vary across these approaches, but are fundamentally similar.

Which is best?

Which “views” model you pick is a matter of preference.

Same ~~pizza~~ system, different slicing.

This section to be continued ...

Thank you for your attention

Software Architecture End of Part I

Questions & suggestions can be sent to:
k.stol@ucc.ie