CSC40232: SOFTWARE ENGINEERING

Professor: Jane Cleland-Huang
Architecture
Wednesday, April 19th
sarec.nd.edu/courses/SE2017

Department of
Computer Science and
Engineering

Dog Houses to sky scrapers

WHY ARCHITECTURE?

2

What is Architecture?

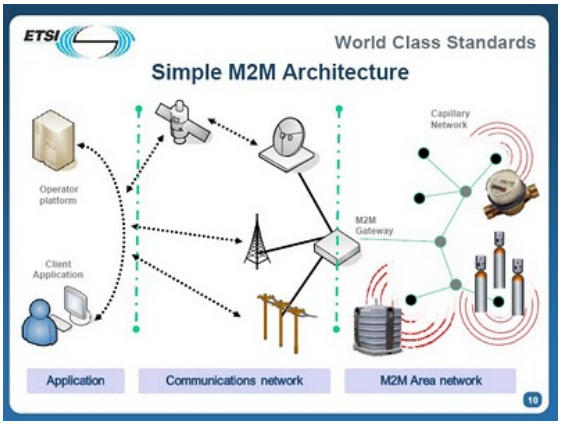
- The software architecture of a program or computing system is the **structure or structures of the system**, which comprises software elements, the externally visible properties of those elements, and the relationships among them.
- An abstract view of the system that distills away implementation, algorithm, and data details – focusing instead on **behavior** and **interaction** of black box elements.

- (Bass definition)

INTRODUCTION

3

Marchitectures



No nutritional value.
Great for 'marketing' the architecture, but provides little information to the developers.

INTRODUCTION

4

An Architecture must answer these questions:

- Which requirements are the structuring and decisions based on?
- Which are the major logical and physical system building blocks?
- How are the system building blocks related to one another?
- What responsibilities do the system building blocks have?
- What interfaces do the system building blocks have?
- How are the system building blocks grouped or layered?
- What are the specifications and criteria used to divide the system into building blocks?

https://www.healthcare.gov/

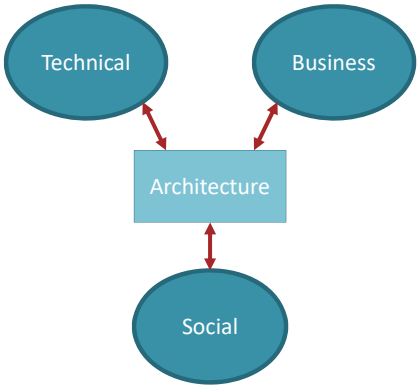
"Other" definitions ???

- Architecture is **high level design**
- Architecture is the **overall structure** of the system (which structure?)
- Architecture is the structure of the components, their interrelationships, AND the **principles and guidelines** governing their design and evolution.
- Architecture is **purely a description of components and connectors**. (i.e. focus on runtime architecture).
- Architecture is the **composition of a set of architectural design decisions**. (Jan Bosch)

6

What drives architectural solutions?

- NOT requirements alone! Different architects would produce different solutions from the same requirements.



- Architecture is influenced by technical, business, and social influences.
- ….and conversely, it influences technical, business, and social environments.

7

INTRODUCTION

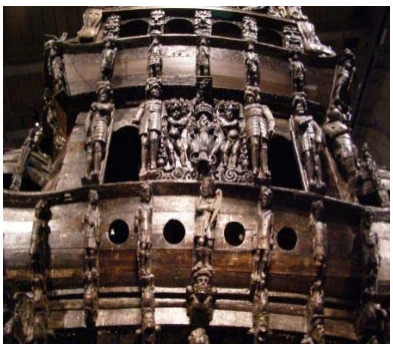
A Lesson from History

- 1625 the Swedish king Gustavus Adlophus ordered new warships.
- The VASA was built in Stockholm. It was to be the mightiest warship in the world, armed with 64 guns on 2 gundecks.

8

INTRODUCTION

A Lesson from History



- The architect, Hybertsson had to balance many concerns:
  - Swift time to deployment
  - Performance
  - Functionality
  - Safety
  - Reliability
  - Cost
- His experience told him to design the VASA as though it were a single-gun-deck ship and then extrapolate.
- "Luckily" for Hybertsson he died one year before the final launch!

9

A Lesson from History

- The project was completed to specifications.
- On Sunday, August 10th, 1628 the ship set sale, waddled out into Stockholm's deep water harbor, fired her guns in salute, and promptly rolled over and sank.
- A post mortem analysis showed that:
  - The ship was well built but badly proportioned.
  - Hybertsson failed to balance conflicting constraints, failed to manage risks, and failed to manage his customers (especially the king!)

10

The architecture life-cycle

- Create the business case.
- Understand the requirements.
- Create or select the architecture.
- Document and communicate the architecture.
- Analyze or evaluate the architecture.
- Implement the system based on the architecture.
- Ensure that the implemented system conforms to the architecture.

11

A second look at the definition of SA

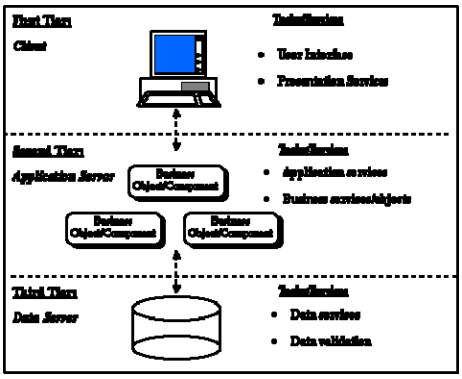
- Software Elements – the Software architecture defines **how elements interact**, and suppresses details that are purely internal to an element.
- Architectural structure is represented through **multiple views**.
- **Every software system has an architecture** (whether formally documented or not)
- The **behavior of each element** is part of the architecture.
- There are good and bad architectures!

  See examples

12

Architectural Patterns, reference models, & reference architectures

- An architectural pattern is a description of **element** and **element types** together with a set of **constraints** on how they are used.



Example: Client server, blackboard, pipe-and-filter.

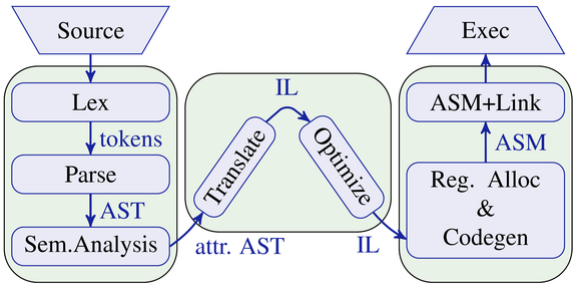
Architectural patterns exhibit known quality attributes.

Also referred to as architectural style.

13

Architectural Patterns, reference models, & reference architectures

- A reference model provides a decomposition of functionality together with the data flow.
- Found in mature domains.



Example: Compiler or DBMS

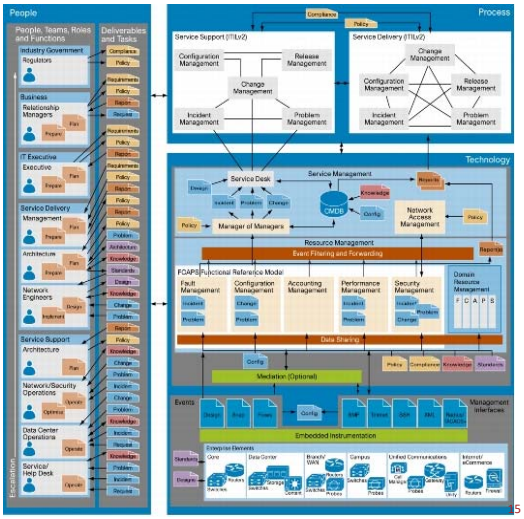
No need to recreate the architecture from the ground up.

14

Architectural Patterns, reference models, & reference architectures

INTRODUCTION

A reference architecture maps a reference model onto software elements.



15

Why is Software Architecture Important?

INTRODUCTION

Communication between stakeholders (although there are other non-Software Architecture-centric techniques for communicating too).

16

What makes a good architecture?

- The architecture should be the product of a **single architect** or a small group of architects.
- The architect should have the **functional requirements** for the system and an articulated, prioritized, **list of quality attributes** that the architecture is expected to satisfy.
- The architecture should be **well documented** with at least one static view and one dynamic view (using understandable, agreed upon notation).
- The architecture should lend itself to **incremental implementation** via the creation of a 'skeletal' system in which communication paths are exercised but which starts out with minimal functionality.

Stories from the trenches: https://www.infoq.com/ebay

17

QUALITIES OF A GOOD ARCHITECTURE

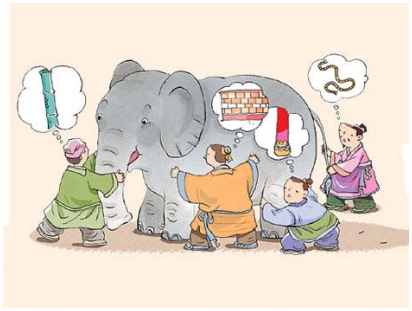
What makes a good architecture?

- Well defined modules (information hiding and separation of concerns).
- Well defined interface for each module.
- Quality concerns achieved using well-known architectural tactics.
- Architecture should be independent of specific versions of commercial products.
- Producers and consumers of data should be separated from each other.

18

QUALITIES OF A GOOD ARCHITECTURE

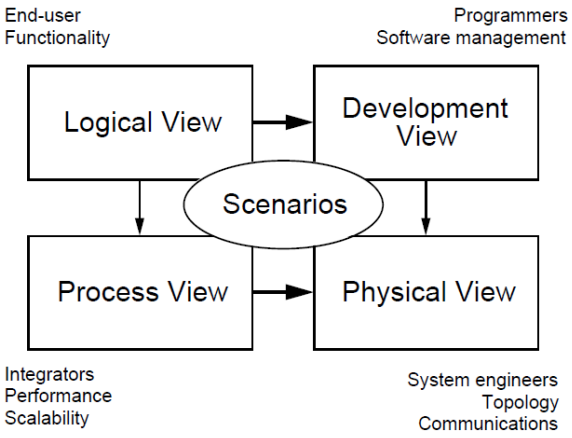
Multiple views



19

ARCHITECTURAL STRUCTURES

4+1 Approach



Software architecture = {Elements, Forms, Rationale/ Constraints}

Formula applied separately to each view.

Different styles can be applied to each view.

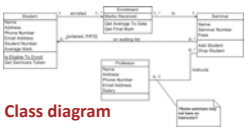
Philippe Kruchten: The 4+1 View Model of Architecture. IEEE Software 12(6): 42-50 (1995)

20

ARCHITECTURAL VIEWS

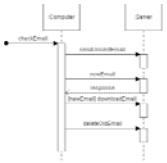
4+1 Approach: Logical view

- Primarily supports **functional requirements** i.e. what the system should provide to its users.
- System decomposed into a set of **key abstractions** (primarily from problem domain) in the form of objects or object classes.
- Exploits abstraction, encapsulation, inheritance.
- Representations: Class diagrams, E-R Diagrams, State transition diagrams, state charts.



Class diagram

Communication Diagram

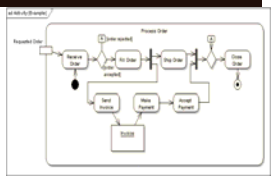


Sequence Diagram

Note: Diagrams can be used for Architecture or design.

21

4+1 Approach: Process view



Activity Diagram

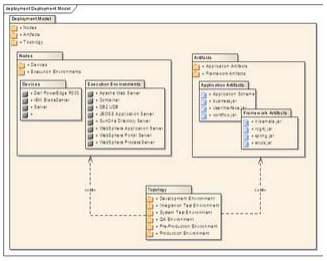
- Models NFRs such as performance, availability, & fault-tolerance.
- A process **groups tasks into an executable unit**, which can be scheduled, started, recovered, reconfigured, shut down, or replicated.
- Major tasks communicate via a set of well-defined **inter-task communication mechanisms**: synchronous and asynchronous message-based communication services, remote procedure calls, etc.
- Minor tasks communicate via shared memory and rendezvous etc.

22

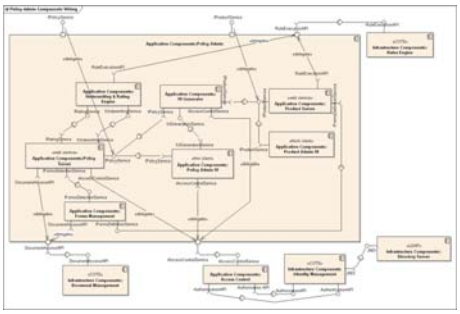
4+1 Approach: Development View

ARCHITECTURAL VIEWS

- Also referred to as **implementation view**.
- Shows organization of software modules, libraries, subsystems, and units of development.
- Serves as an allocation view.



Package Diagram



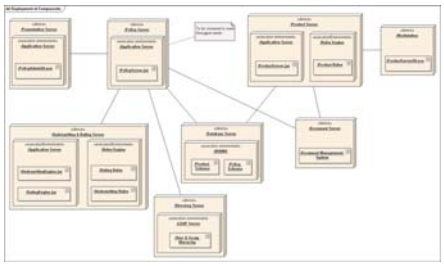
Component Diagram

23

4+1 Approach: Physical View

ARCHITECTURAL VIEWS

- Also referred to as **deployment view**.
- Takes into account NFRs such as availability, reliability (fault-tolerance), performance (throughput), and scalability.
- Various **elements** i.e. processes, tasks, and objects—need to be **mapped onto various nodes**.
- Different configurations for development and testing, deployment etc. Mapping needs to be flexible.



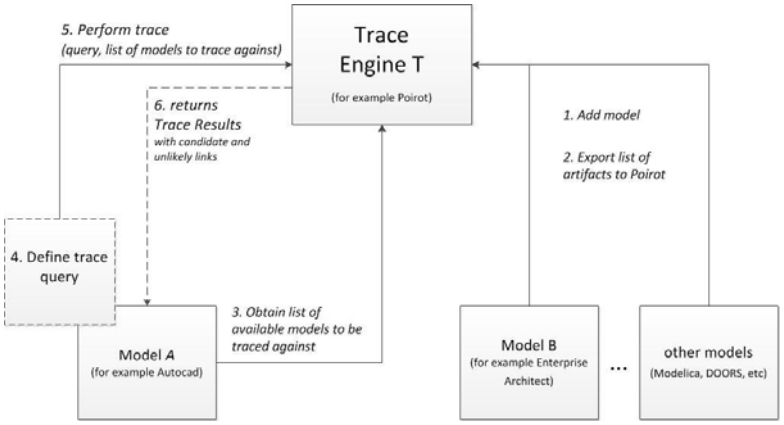
Deployment Diagram

24

The good, the bad, and the ugly…

EXTRA MATERIALS

Push Architecture

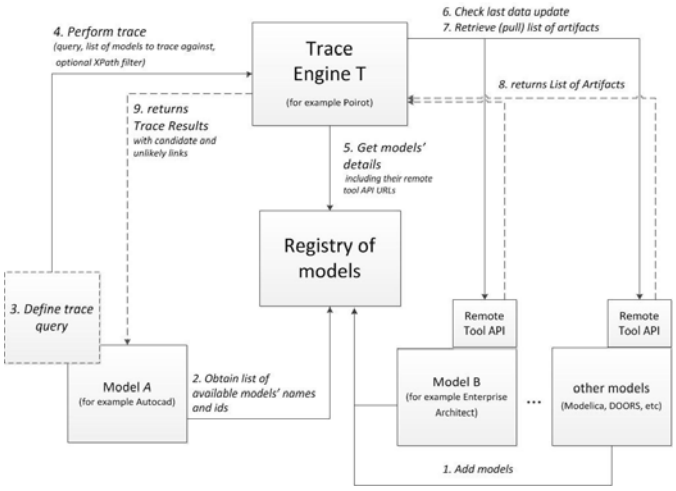


25

The good, the bad, and the ugly…

EXTRA MATERIALS

Pull Architecture

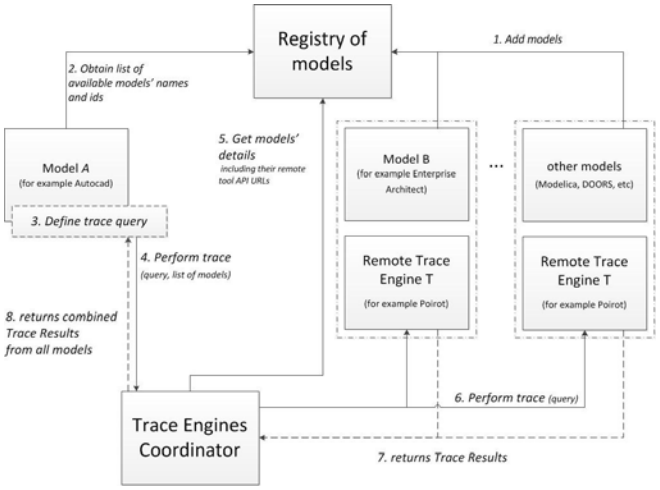


26

The good, the bad, and the ugly…

EXTRA MATERIALS

Distributed Architecture with Central Coordination



Return to slide sequence

27