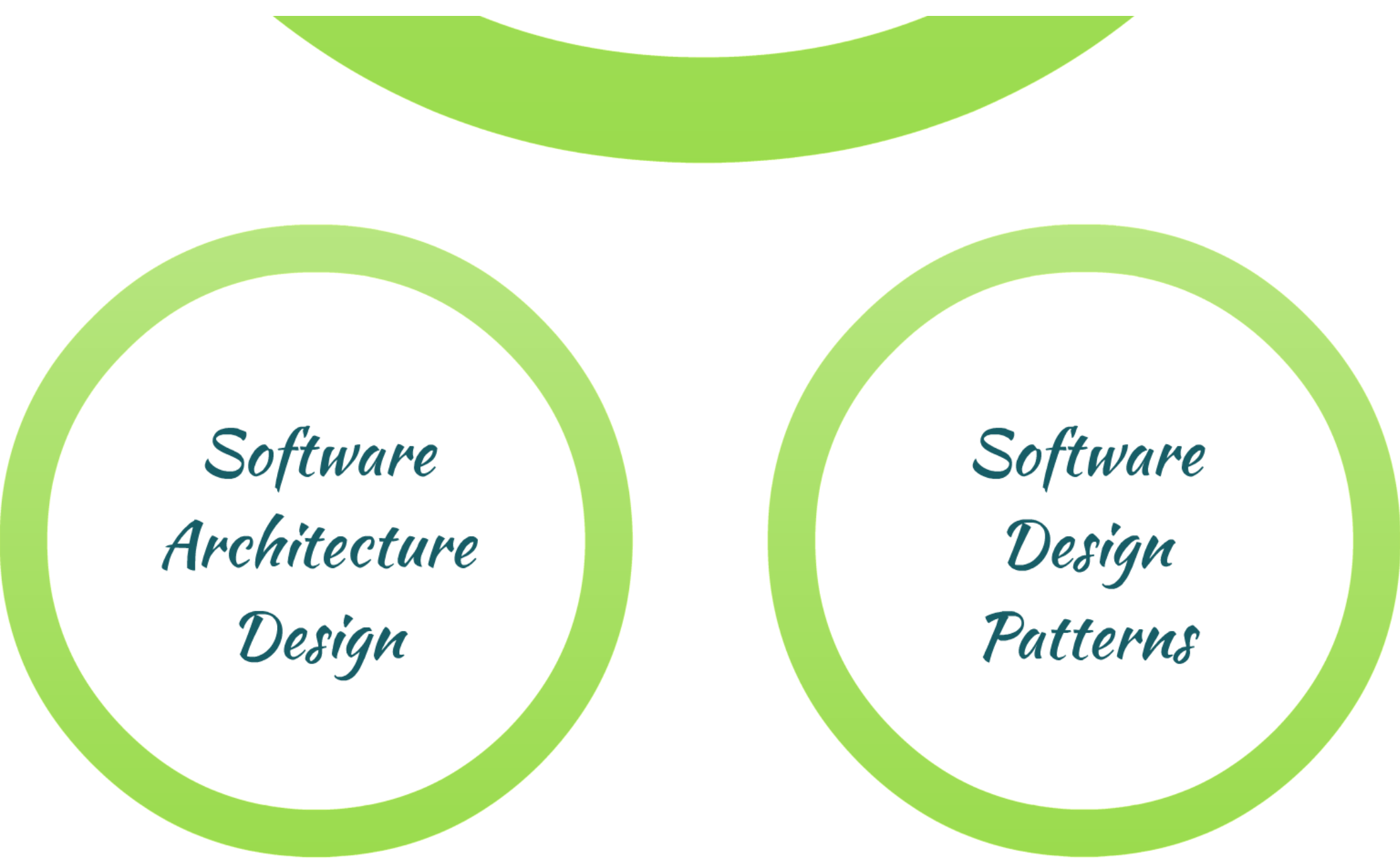
软件系统设计

Software Systems Design


Software Architecture Design
Software Design Patterns

Course Logistics

Who am I?

Lecturer in Charge:

2023本科生系统设计课
群号：316378602

张贺，He (Jason) Zhang

Office: 费彝民楼B座812室
Phone: 83621360/61/62/63 (812)
Email: hezhang@nju.edu.cn
Web: https://softeng.nju.edu.cn/faculty/hezhang/

Why Study Software Design & Architecture?

- Software (IT) systems are everywhere
- Every software intensive system has a software design and architecture
- Software design and architecture are an increasingly important area of practice, education, and research
- As a profession: Software Architect
- As a research area:
  - Originally started around 1960
  - Attracting major attention since 1990
- This course is about
  - Concepts, principles, methods, and patterns of software design and architecture
  - State-of-the-art practices of software design and architecture

No.1 in 2010
No.3 in 2012
No.3 in 2013
No.1 in 2015

Best Jobs in America

2016

CNNMoney/PayScale's top 100 careers with big growth, great pay and satisfying work.

1 of 100

NEXT

1. Software Architect

Median pay: $124,000
Top pay: $169,000
10-year job growth: 23%

PHOTO: JULIE FELPEL

Christopher Felpel, software architect/consultant

In the same way an architect designs a house, software architects lay out a design plan for new programs. That usually means leading a team of developers and engineers, and making sure all the pieces come together to make fully-functioning software.

What's great: New problems come up all the time and new technologies arise, making each day different, and keeping professionals in demand. "I'm pinged at least once or twice a week for new opportunities," said software architect Christopher Felpel. "There's just a lot of work out there, and that doesn't surprise me." *--Jillian Eugenios*

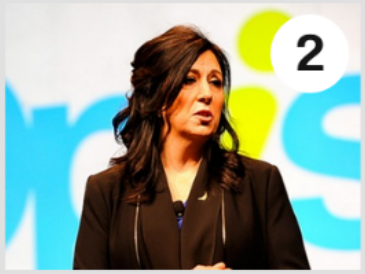
PHOTO: SHUTTERSTOCK

http://money.cnn.com/pf/best-jobs/

Top Ten Jobs

See the full list

Mobile App Developer



Risk Management Director

Landman

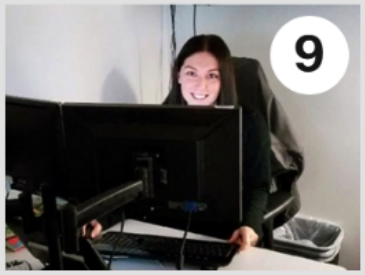
Product Analyst

Info Assurance Analyst

QA Coordinator

Clinical Applications Specialist

Hospital Administrator



Database Analyst

Director, Finance & Administration

Learning Objectives

- Understand concepts and principles of software design and architecture
- Create software architecture by taking requirements or through reverse architecting
- Apply patterns, styles, middleware technologies and frameworks in creating software architecture and design
- Analyze software design and evaluate software architecture systematically
- Understand state-of-the-art methods applied in software design and architecture
- Understand relationships between software design and software architecture, and other software engineering topic areas

Course Design 1

- Sources
  - SECS (Software Engineering Curriculum Standard)
    - Section 4.11: Software Modelling and Analysis
    - Section 4.12: Software Design
    - Strong relationships with other areas
  - SWEBOK (Software Engineering Body of Knowledge)
    - Chapter 3: Software Design
- Previous Courses
  - UNSW COMP9117
  - USYD INFO3220
  - ITU & SMU
  - NJU 2014-2022
- 2023
  - 2023 April (Week 9- Week 11): Basics of Software Architecture and Design Patterns
  - 2019 May (Week 13 - Week 16): Software Architecture Methods, Advanced Topics of Software Architecture

Course Design 2

- Lectures
  - Interaction is preferred in the class
- Assignments
  - "Apply" the principles and methods you have learned
  - Conduct independent research on "technologies"
  - Leverage your programming and technical background
  - All assignments will be plagiarism-checked!
- Exercises, Projects and Presentations
  - Students will be assigned to groups for assignments and project
  - These sessions are meant to reinforce learning about the lecturing material and help you to gain understanding and skills
  - For group project, you have to create and evaluate a software architecure with partial design systematically

Final Mark

- Final exam composition:
  - 50% from architecture design
  - 50% from design patterns
- Final mark = final exam * 80% + assignments * 20%
- Final exam comprising a mixture of
  - Concise questions
    - Knowledge: recall concepts, principles and methods
    - Comprehension: interpret, compare, contrast ...
  - Design and evaluation exercises:
    - Comprehension: translate knowledge to new context, infer causes ...
    - Application: apply what you have learned in new concrete situation ...
  - Necessary reference material will be supplied

References: (no official textbooks)

- L. Bass, P. Clements and R. Kazman (2013/2022) Software Architecture in Practice, 3rd/4th Edition, Addison-Wesley
- R.N. Taylor, N. Medvidovic and E.M. Dashofy (2010) Software Architecture: Foundations, Theory, and Practice, John Wiley & Sons
- A. Shalloway and J. Trott (2004) Design Patterns Explained: A New Perspective on Object Oriented Design, 2nd Edition, Addison-Welsey

- N. Rozanshi and E. Woods (2011) Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, 2nd Edition, Addison-Wesley
- P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford (2011) Documenting Software Architectures: Views and Beyond, 2nd Edition, Addison-Wesley
- F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal (1996) Pattern-Oriented Software Architecture Volume 1: A System of Patterns, John Wiley & Sons

References: (no official textbooks)

- L. Bass, P. Clements and R. Kazman (2013/2022) Software Architecture in Practice, 3rd/4th Edition, Addison-Wesley
- R.N. Taylor, N. Medvidovic and E.M. Dashofy (2010) Software Architecture: Foundations, Theory, and Practice, John Wiley & Sons
- A. Shalloway and J. Trott (2004) Design Patterns Explained: A New Perspective on Object Oriented Design, 2nd Edition, Addison-Wesley

- N. Rozanshi and E. Woods (2011) Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, 2nd Edition, Addison-Wesley
- P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford (2011) Documenting Software Architectures: Views and Beyond, 2nd Edition, Addison-Wesley
- F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal (1996) Pattern-Oriented Software Architecture Volume 1: A System of Patterns, John Wiley & Sons

Who am I?

Lecturer in Charge:

张贺，He (Jason) Zhang

2023本科生系统设计课

Office: 费彝民楼B座812室
Phone: 83621360/61/62/63 (812)
Email: hezhang@nju.edu.cn
Web: https://softeng.nju.edu.cn/faculty/hezhang/

Why Study Software Design & Architecture?

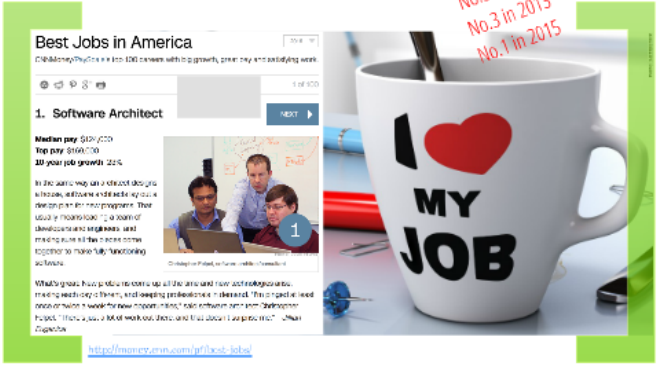
- Software (IT) systems are everywhere
- Every software intensive system has a software design and architecture
- Software design and architecture are an increasingly important area of practice, education, and research
- As a profession: Software Architect
- As a research area:
  - Originally started around 1960
  - Attracting major attention since 1990
- This course is about
  - Concepts, principles, methods, and patterns of software design and architecture
  - State-of-the-art practices of software design and architecture

Final Mark

- Final exam composition:
  - 50% from architecture design
  - 50% from design patterns
- Final mark = final exam * 80% + assignments * 20%
- Final exam comprising a mixture of
  - Concise questions
    - Knowledge: recall concepts, principles and methods
    - Comprehension: interpret, compare, contrast ...
  - Design and evaluation exercises:
    - Comprehension: translate knowledge to new context, infer causes ...
    - Application: apply what you have learned in new concrete situation ...
  - Necessary reference material will be supplied

Course Logistics

No.1 in 2010
No.3 in 2012
No.3 in 2013
No.1 in 2015



Course Design 2

- Lectures
  - Interaction is preferred in the class

- Assignments
  - "Apply" the principles and methods you have learned
  - Conduct independent research on "technologies"
  - Leverage your programming and technical background
  - All assignments will be plagiarism-checked!

- Exercises, Projects and Presentations
  - Students will be assigned to groups for assignments and project
  - These sessions are meant to reinforce learning about the lecturing material and help you to gain understanding and skills
  - For group project, you have to create and evaluate a software architecure with partial design systematically

Course Design 1

- Sources
  - SECS (Software Engineering Curriculum Standard)
    - Section 4.11: Software Modelling and Analysis
    - Section 4.12: Software Design
    - Strong relationships with other areas
  - SWEBOK (Software Engineering Body of Knowledge)
    - Chapter 3: Software Design
- Previous Courses
  - UNSW COMP9117
  - USYD INFO3220
  - ITU & SMU
  - NJU 2014-2022
- 2023
  - 2023 April (Week 9- Week 11): Basics of Software Architecture and Design Patterns
  - 2019 May (Week 13 - Week 16): Software Architecture Methods, Advanced Topics of Software Architecture

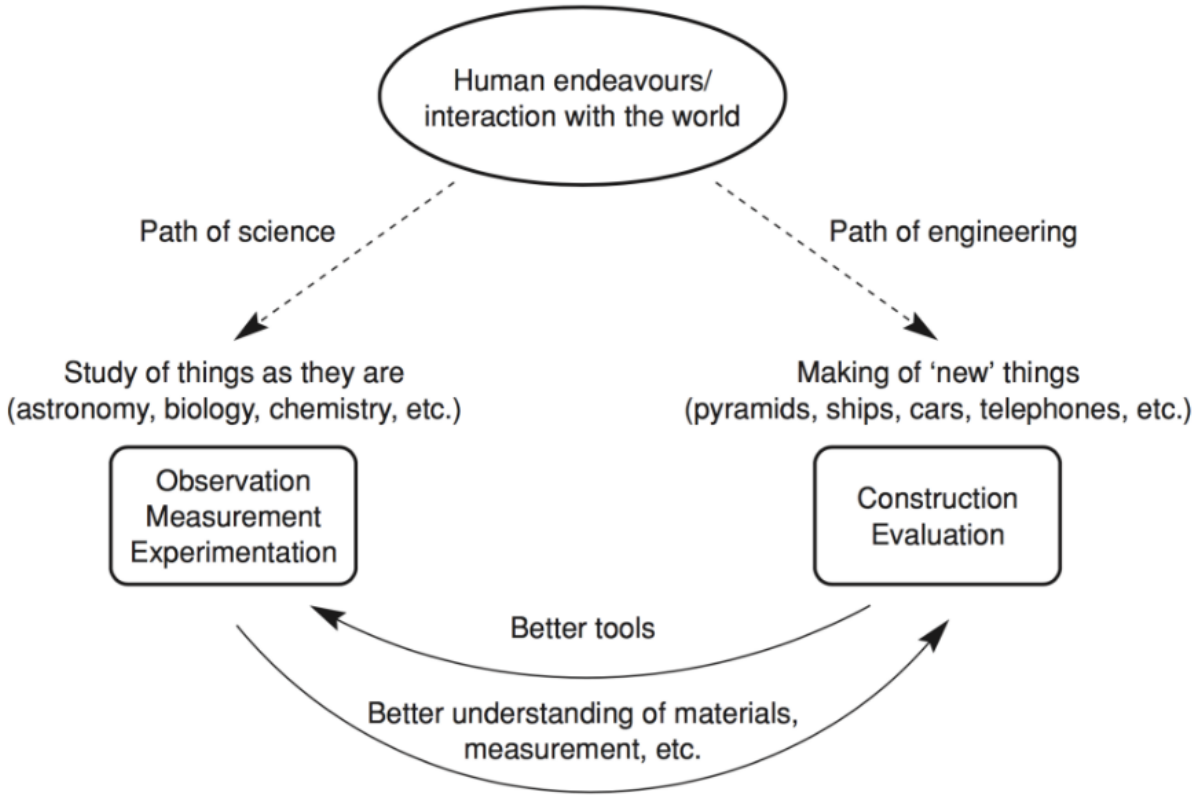
Learning Objectives

- Understand concepts and principles of software design and architecture
- Create software architecture by taking requirements or through reverse architecting
- Apply patterns, styles, middleware technologies and frameworks in creating software architecture and design
- Analyze software design and evaluate software architecture systematically
- Understand state-of-the-art methods applied in software design and architecture
- Understand relationships between software design and software architecture, and other software engineering topic areas

Introduction

Understanding Software Engineering

- Software | Engieering
  - Software vs. ?
  - Science vs. Engineering

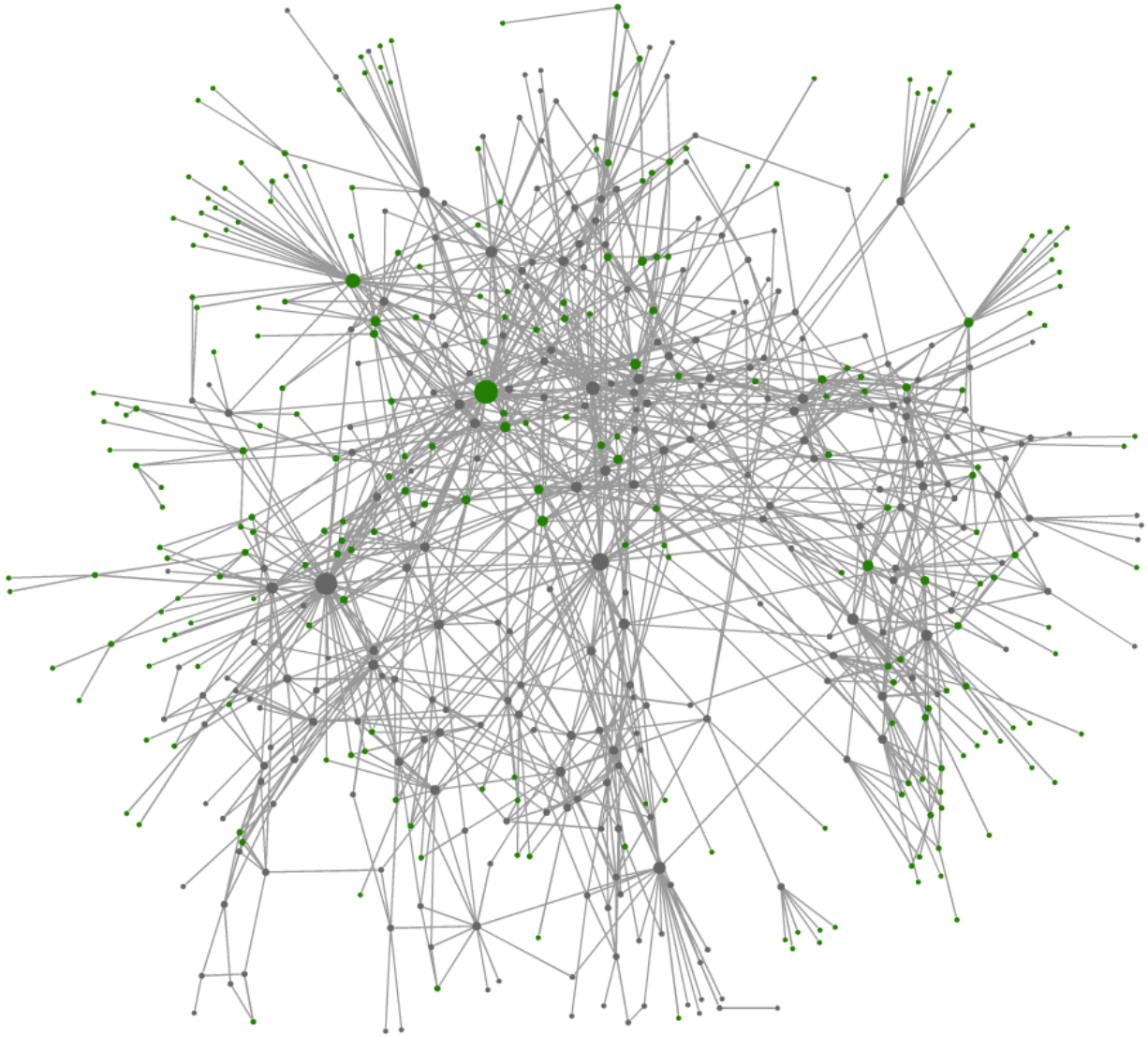


What is Software Architecture?

- Definition 1: "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 among them." [Software Engineering Institute (SEI)]
- Definition 2: "The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution." [IEEE 1471 -2000 "Recommended Practice for Architectural Description of Software-Intensive Systems"]

Architecture vs. Design

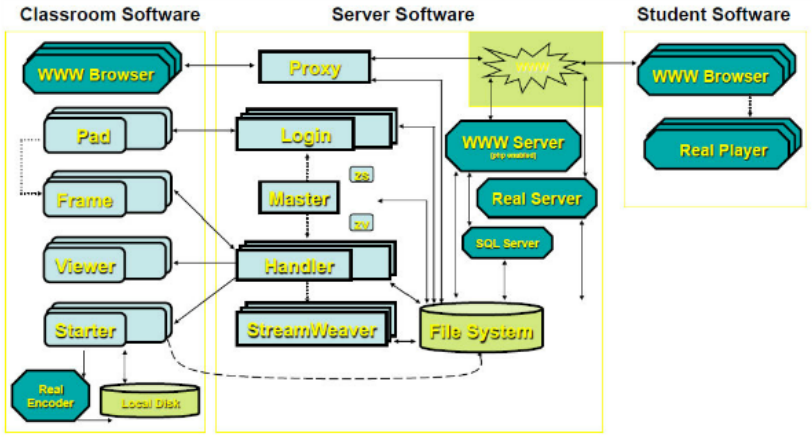
- It's about software design
  - All architecture is software design, but not all design is software architecture
  - "Architecting" is part of the design process
- Other views:
  - High-level designs
  - A set of design decisions
- Structure/Organization of the system
  - Elements: components & connectors
  - Relationships: static & dynamic relationships
- Properties: elements, groups of elements & overall system

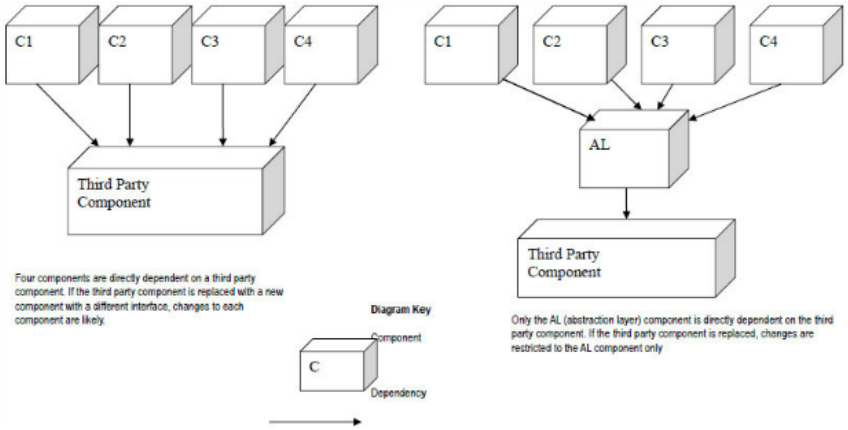


Architecture vs. Structure

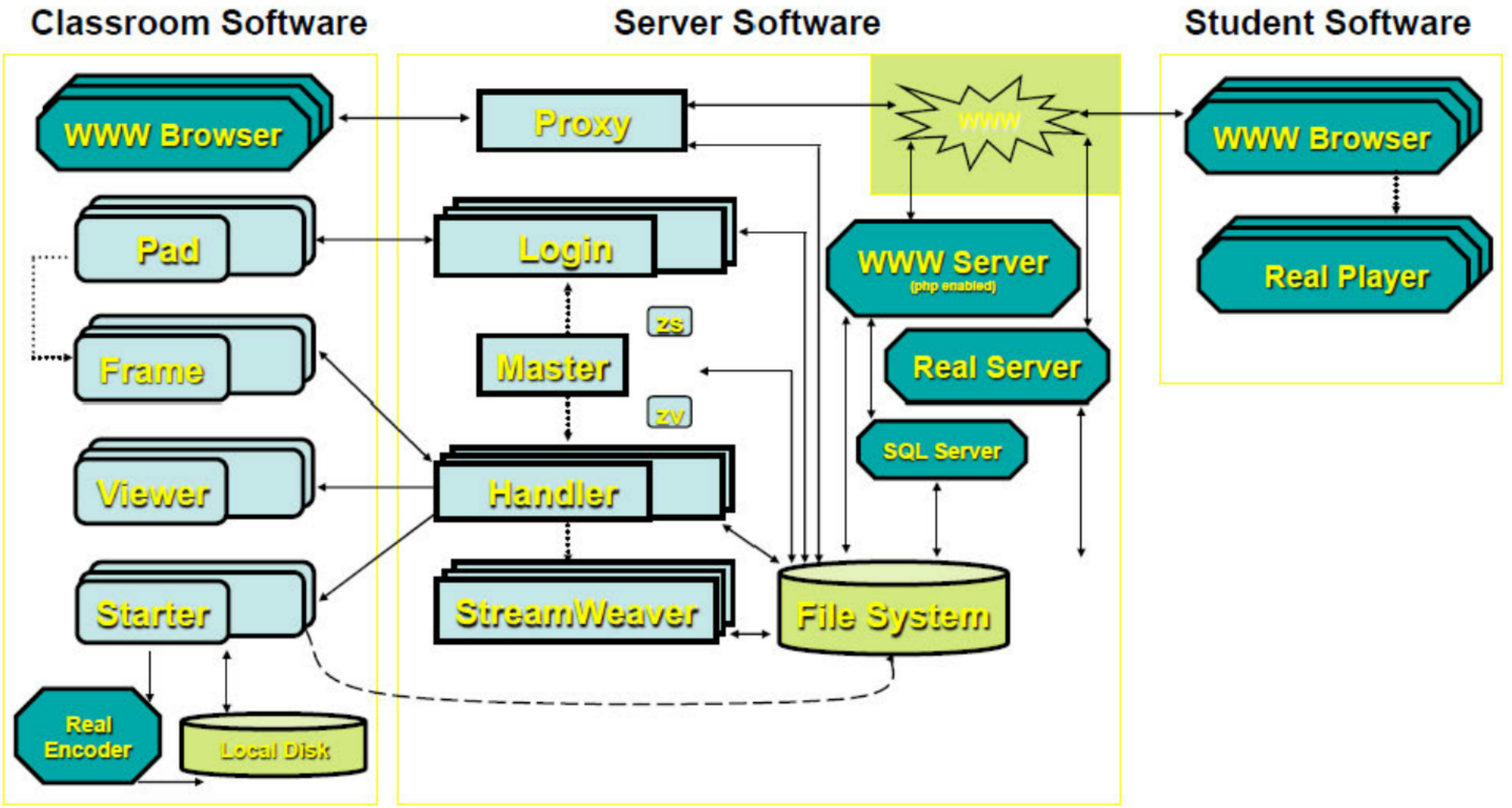
- Decomposition of system into components/modules/ subsystems
- Architecture defines:
  - Component interfaces
    - What a component can do?
  - Component communications and dependencies
    - How components communicate?
  - Component responsibilities
    - Precisely what a component will do when you ask it?

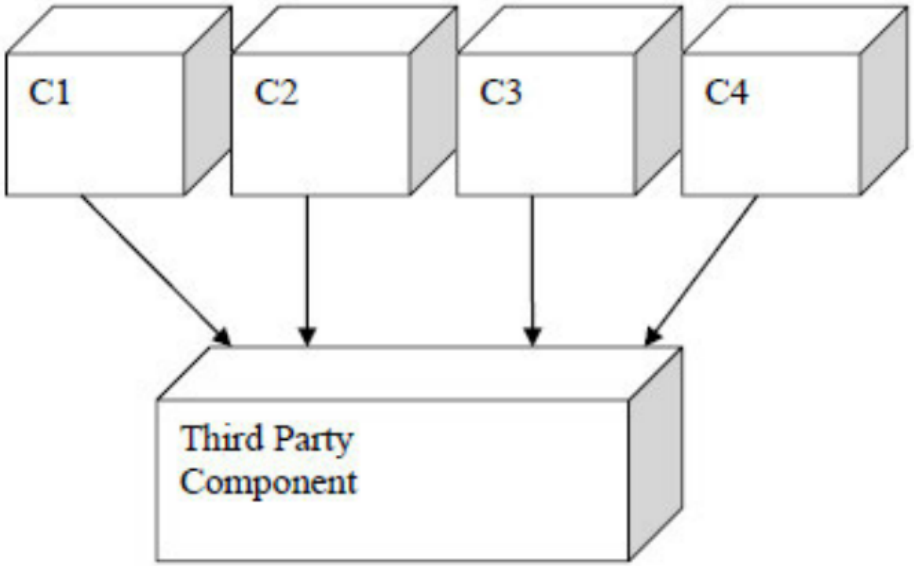
Structure and Architecture



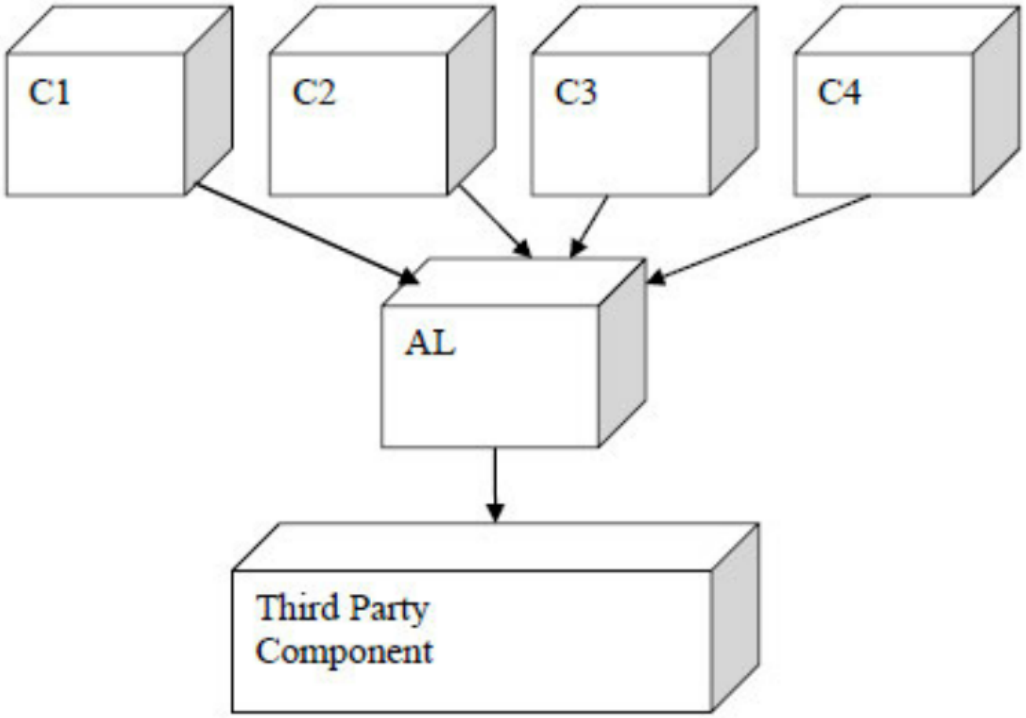


ucture and Architectu



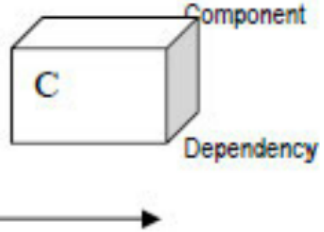


Four components are directly dependent on a third party component. If the third party component is replaced with a new component with a different interface, changes to each component are likely.



Only the AL (abstraction layer) component is directly dependent on the third party component. If the third party component is replaced, changes are restricted to the AL component only

Diagram Key



Architecture Specifies Communication

- Communication involves:
  - Data passing mechanisms, for example:
    - Function call
    - Remote method invocation
    - Asynchronous message
  - Control flow
    - Flow of messages between components to achieve required functionality
    - Sequential
    - Concurrent/parallel
    - Synchronization

Architecture Addresses NFRs

- Non-functional requirements (NFRs) define "how well a system works?"
- NFRs rarely captured in functional requirements
  - Aka. architecture requirements
  - Must be elicited by architect
- NFRs include:
  - Technical constraints
  - Business constraints
  - Quality attributes
- Discussion: A list of quality attributes?

How to Develop a Design?

Generic Design Strategies:

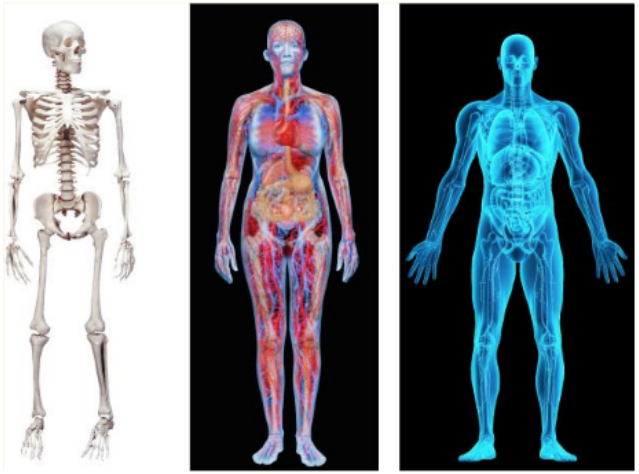
- Decomposition
- Abstraction
- Stepwise: 'Divid and Conquer'
- Generate and Test
- Iteration: Incremental Refinement
- Reuseable elements

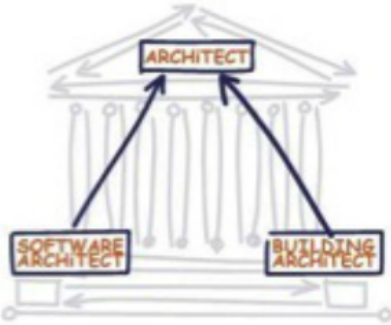
Design is an Abstraction

- Architecture provides an higher level abstract view of a design
  - Hides complexity and implementation of design
  - May or may not be a direct mapping between architecture elements and software elements
- Blackbox design and Whitebox design
  - Informal depiction of system's structure and interactions.
  - Portray the design philosophies embodied in the architecture
- Discussion: Why abstraction in design?

Architecture Views

- A software architecture represents a complex design artifact
- Many possible 'views' of the architecture
  - Analogy with buildings – floor plan, external, electrical, plumbing, air-conditioning





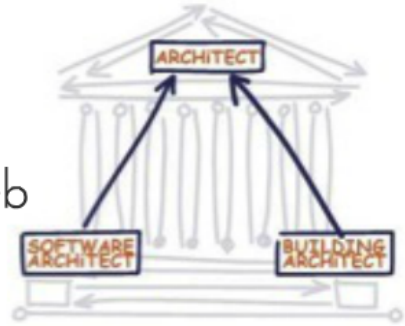
P. Krutchen's 4+1 View Model

- Logical view: describes architecturally significant elements of the architecture and the relationships between them.
- Process view: describes the concurrency and communications elements of an architecture.
- Physical view: depicts how the major processes and components are mapped on to the applications hardware.
- Development view: captures the internal organization of the software components as held in e.g., a configuration management tool.
- Architecture use cases: capture the requirements for the architecture; related to more than one particular view

Architect & Software Architect

"Architects design structures to meet human needs."- James Fitch, 1972

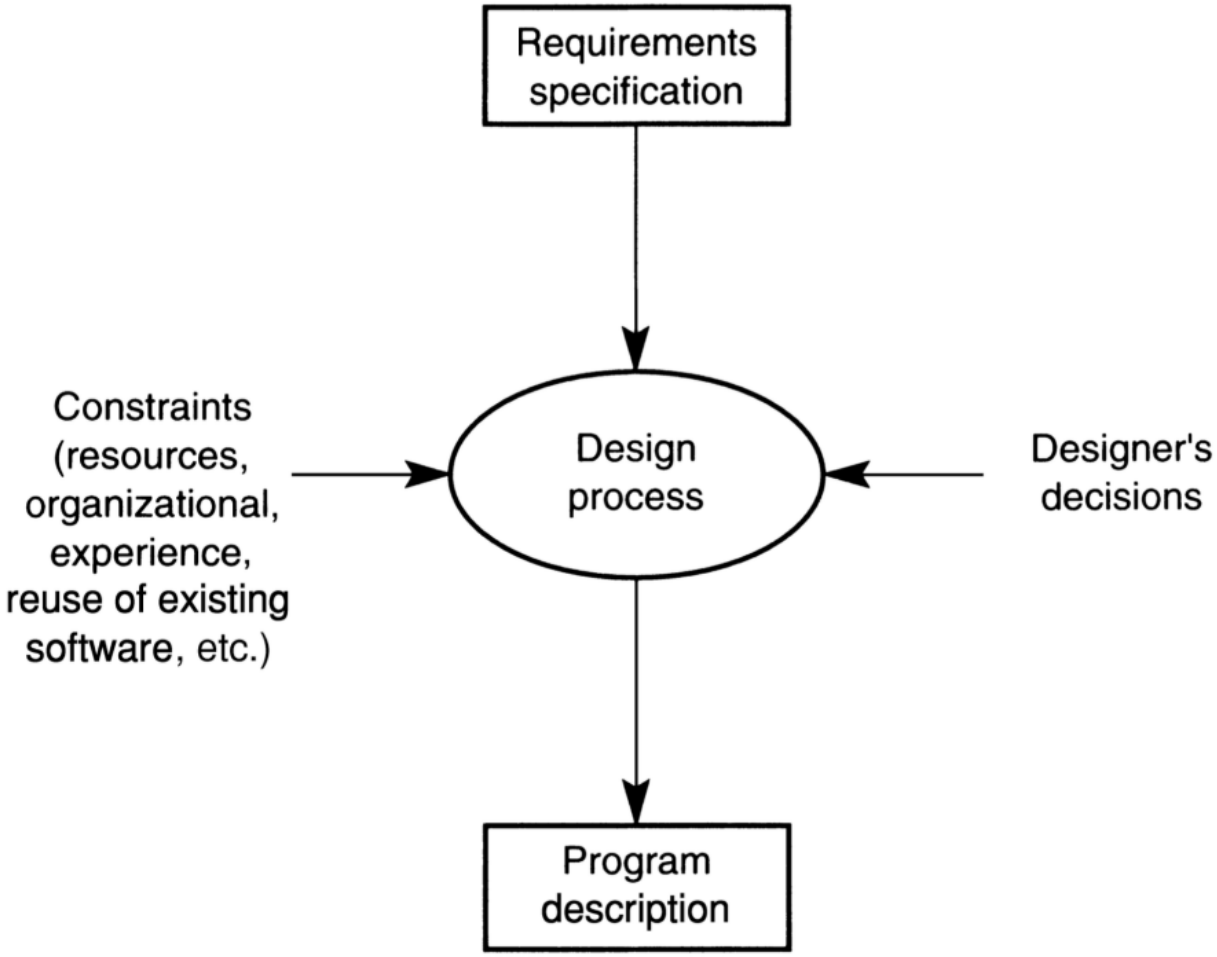
- The role of the architect remains the same
  - Listening to clients, understanding the totality of needs
  - Scrutinizing feasibilities
  - Forming a practical vision of a structure and creating a blueprint
  - Overseeing construction and ensuring compliance to the plan
  - Guiding the vision through the tempest of design changes, crises and ambiguities
- Software architects oversee software construction professionals
  - Programmers, Engineers, Designers
- Famous software architects:
  - Bill Gates: Chief Software Architect of Microsoft
  - Tim Bernrs-Lee: Inventor and Chief Architect of World Wide Web
  - Roy Fielding: Representational State Transfer (REST)



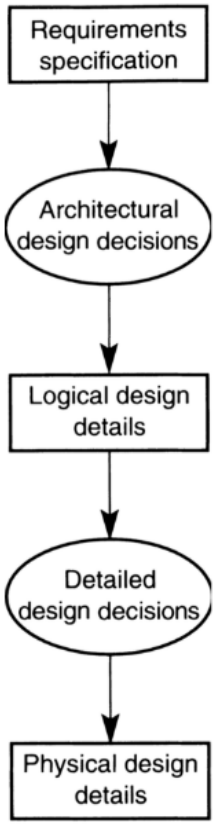
What Does a Software Architect Do?

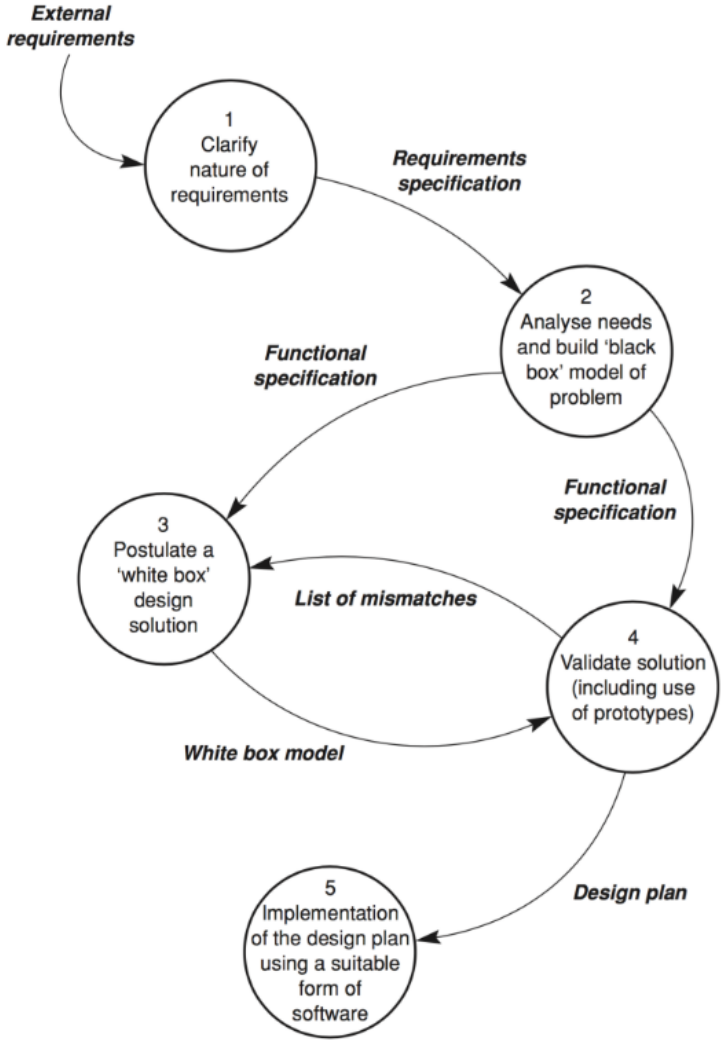
- Liaison
  - Among clients, technical team and business/ requirements analysts
  - With management or marketing
- Software Engineering
  - Software engineering best practices
- Technology Knowledge
  - Deep understanding of technology domain
- Risk Management
  - Risks associated with the design, technology choices
  - More?

A General Design Model



Software Design Process

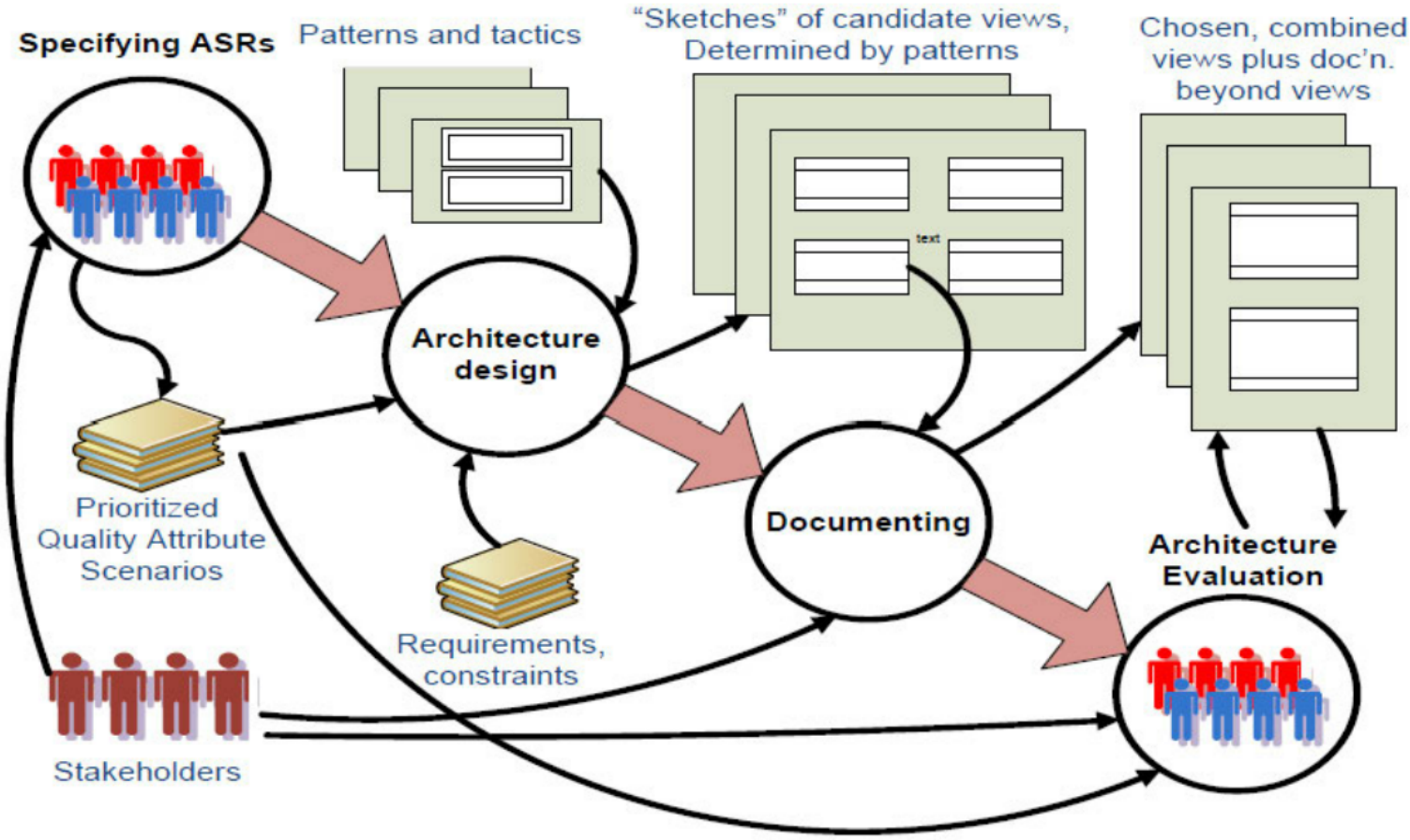




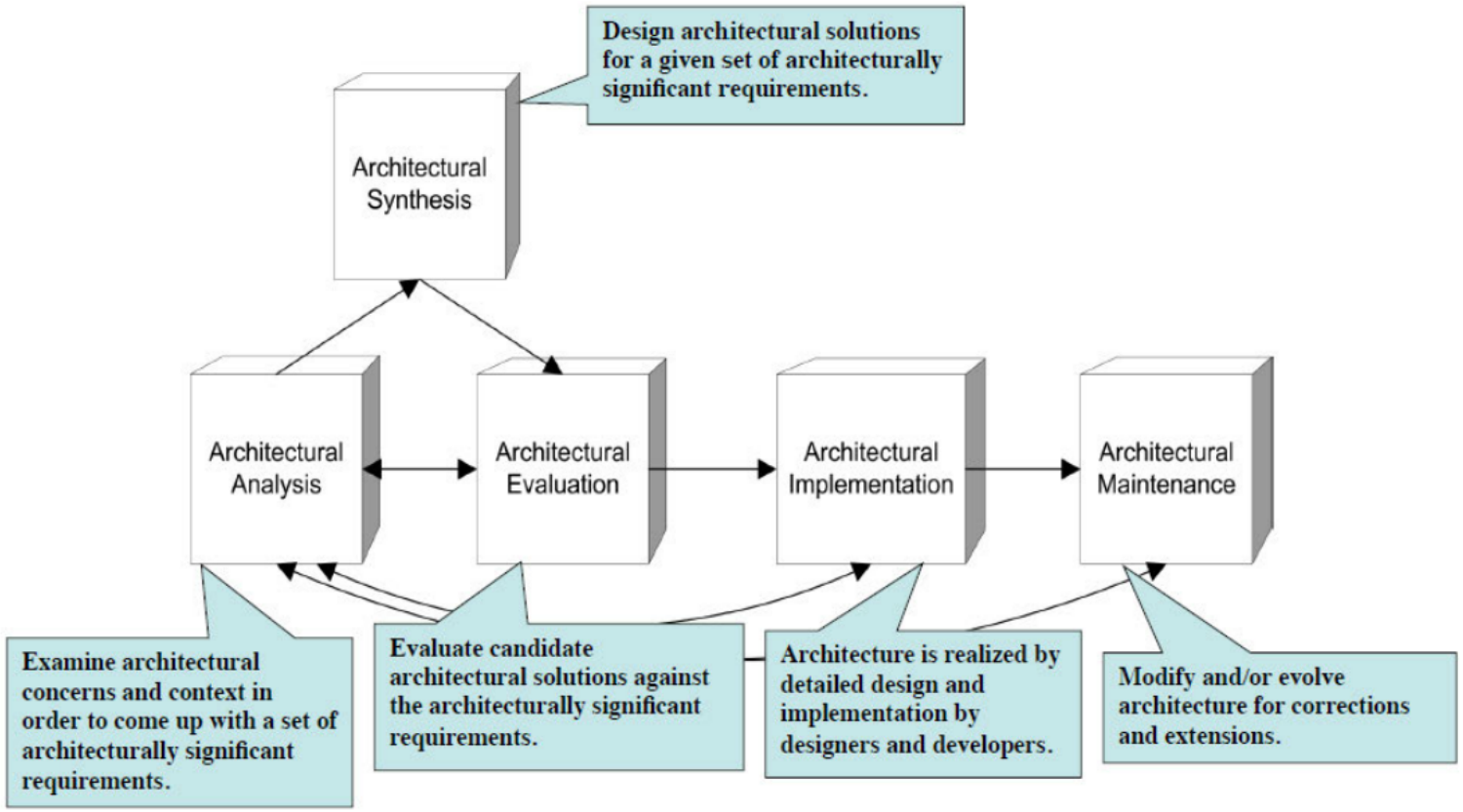
Architecture Activities

- Creating the business case for the System
- Understanding the requirements
- Creating and selecting architecture
- Communicating the architecture (stakeholders including developers)
- Analysing or evaluating the architecture
  - Overall methodologies
  - Quality specific techniques
- Implementing the architecture
- Ensuring conformance to an architecture

Software Architecture Process



Architecture Lifecycle



Software Design & Architecture Knowledge Areas

- Software Design Basic Concepts
  - General design concepts
  - Context: software development life cycle - requirements, design, construction and testing
  - Design process (role, activity, work product)
  - Enabling techniques for software design
- Key Issues (technical): concurrency, control and handling of events, distribution, exception handling, interactive systems, persistence
- Software Structure and Architecture
  - Architecture Structures and viewpoints
  - Architectural styles and patterns (macro-architecture)
  - Design patterns (micro-architecture)
- Software Design Methods
  - Architecture Methods (e.g., Attribute-Driven Design)
  - Design Methods (e.g., Dynamic System Development Method)

Software Design & Architecture Knowledge Areas

- Software Design Quality Analysis and Evaluation
  - Quality attributes
  - Quality analysis and evaluation methods, techniques and tools
    - Design reviews (e.g. SEI's Architecture Trade-off Analysis Method)
    - Static analysis and dynamic analysis
    - Simulation and prototyping
  - Measures:
    - Metrics: Architecture level
    - Technique specific measures
- Design Modeling and Representation
  - Architecture and Design Notations (Architecture Description Languages (ADL))
  - Unified Modelling Language (UML)
  - Design Documentation (Views & Beyond)
  - Others: differ in ability, focus and domain (e.g. ACME, Rapide)

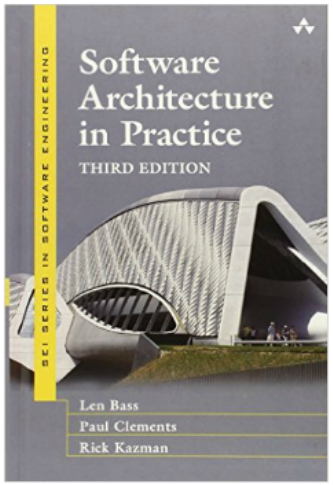
Lectures Preview

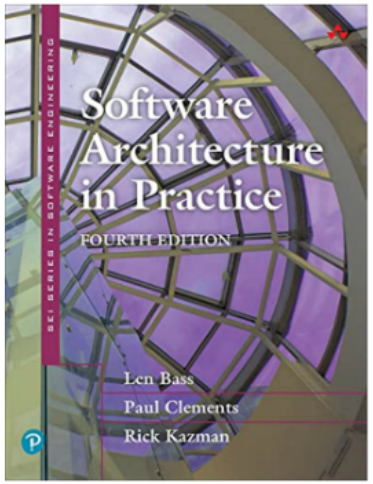
- Lecture 1: Introdcution
  - Concepts of software architecture
- Lecture 2-4: Quality Attributes and Architecture Tactics
  - Quality attributes, NFRs and scenarios
  - Tactics achieving quality attributes
  - Architecturally significant requirements
- Lecture 5/6: Architecture Patterns
  - Patterns in software design
  - Architecture styles and patterns
- Lecture 7/8: Architecture Design Methods
  - Design strategies in general
  - Software design process and architecting process
  - ADD method: Attribute Driven Design

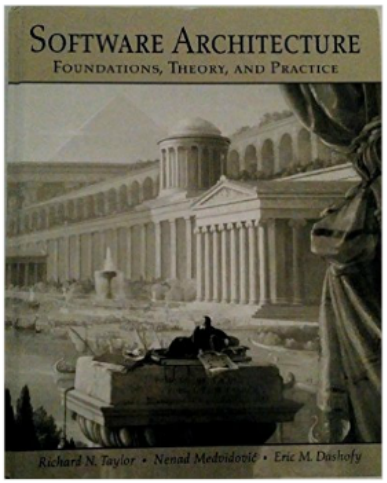
Lectures Preview

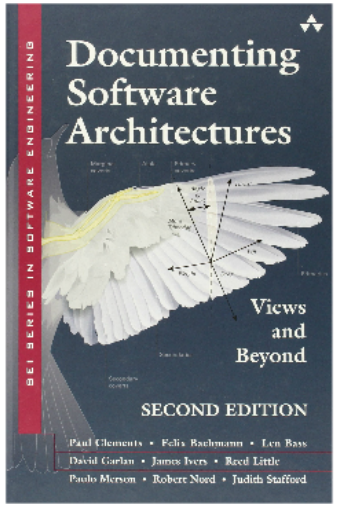
- Lecture 9: Guest Lecture
- Lecture 10: Documenting Software Architecture
  - Modelling notations, views and more
  - Documenting rationales behind your architecture
- Lecture 11: Evaluating Software Architecture
  - Single attribute specific techniques
  - Trade-off analysis among multiple quality attributes
- Lecture 12: Microservices Architecture
- Lecture 13: Course Review —— Softare Architecture
- Final Exam

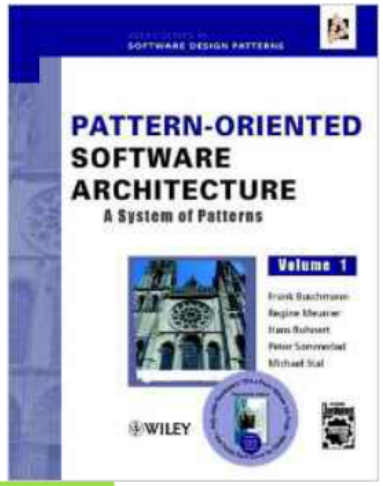
Reference Books

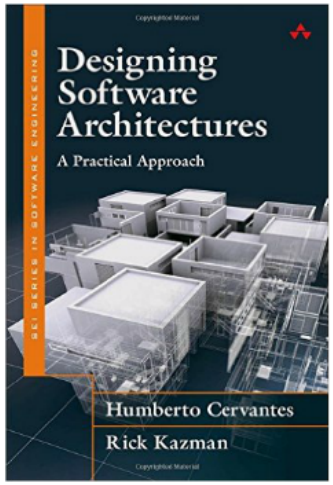


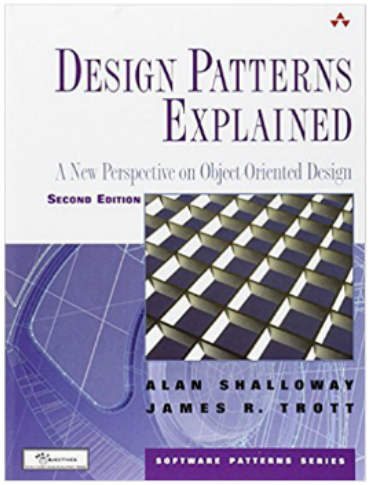


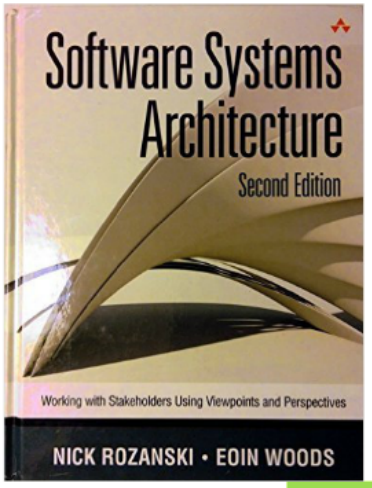












Discussion

- What is Difference between Science and Engineering?
- What is Difference between 'Software' and 'Hardware'?
- What is Difference between Architecture and Design?
- What is Difference between Architecture and Structure?

Discussion

- What is Difference between Science and Engineering?
- What is Difference between 'Software' and 'Hardware'?
- What is Difference between Architecture and Design?
- What is Difference between Architecture and Structure?

Architecture Views

- A software architecture represents a complex design artifact
- Many possible 'views' of the architecture
  - Analogy with buildings – floor plan, external, electrical, plumbing, air-conditioning

How to Develop a Design?

Generic Design Strategies:

- Decomposition
- Abstraction
- Stepwise: 'Divid and Conquer'
- Generate and Test
- Iteration: Incremental Refinement
- Reuseable elements

P. Krutchen's 4+1 View Model

- Logical view: describes architecturally significant elements of the architecture and the relationships between them.
- Process view: describes the concurrency and communications elements of an architecture.
- Physical view: depicts how the major processes and components are mapped on to the applications hardware.
- Development view: captures the internal organization of the software components as held in e.g., a configuration management tool.
- Architecture use cases: capture the requirements for the architecture; related to more than one particular view

Architect & Software Architect

"Architects design structures to meet human needs." - James Fitch, 1972

- The role of the architect remains the same
  - Listening to clients, understanding the totality of needs
  - Scrutinizing feasibilities
  - Forming a practical vision of a structure and creating a blueprint
  - Overseeing construction and ensuring compliance to the plan
  - Guiding the vision through the tempest of design changes, crises and ambiguities
- Software architects oversee software construction professionals
  - Programmers, Engineers, Designers
- Famous software architects:
  - Bill Gates: Chief Software Architect of Microsoft
  - Tim Berners-Lee: Inventor and Chief Architect of World Wide Web
  - Roy Fielding: Representational State Transfer (REST)

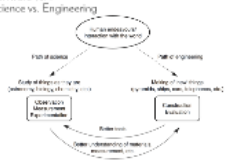
Reference Books

Design is an Abstraction

- Architecture provides an higher level abstract view of a design
  - Hides complexity and implementation of design
  - May or may not be a direct mapping between architecture elements and software elements
- Blackbox design and Whitebox design
  - Informal depiction of system's structure and interactions.
  - Portray the design philosophies embodied in the architecture
- Discussion: Why abstraction in design?

Understanding Software Engineering

- Software | Engineering
  - Software vs. ?
  - Science vs. Engineering



What is Software Architecture?

- Definition 1: "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 among them." [Software Engineering Institute (SEI)]
- Definition 2: "The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution." [IEEE 1471 -2000 "Recommended Practice for Architectural Description of Software-Intensive Systems"]

What Does a Software Architect Do?

- Liaison
  - Among clients, technical team and business/ requirements analysts
  - With management or marketing
- Software Engineering
  - Software engineering best practices
- Technology Knowledge
  - Deep understanding of technology domain
- Risk Management
  - Risks associated with the design, technology choices
  - More?

Lectures Preview

- Lecture 9: Guest Lecture
- Lecture 10: Documenting Software Architecture
  - Modelling notations, views and more
  - Documenting rationales behind your architecture
- Lecture 11: Evaluating Software Architecture
  - Single attribute specific techniques
  - Trade-off analysis among multiple quality attributes
- Lecture 12: Microservices Architecture
- Lecture 13: Course Review — Software Architecture
- Final Exam

Architecture Addresses NFRs

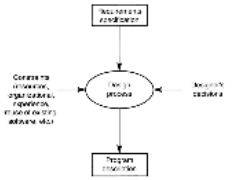
- Non-functional requirements (NFRs) define "how well a system works?"
- NFRs rarely captured in functional requirements
  - Aka. architecture requirements
  - Must be elicited by architect
- NFRs include:
  - Technical constraints
  - Business constraints
  - Quality attributes
- Discussion: A list of quality attributes?

Introduction

Architecture vs. Design

- It's about software design
  - All architecture is software design, but not all design is software architecture
  - "Architecting" is part of the design process
- Other views:
  - High-level designs
  - A set of design decisions
- Structure/Organization of the system
  - Elements: components & connectors
  - Relationships: static & dynamic relationships
- Properties: elements, groups of elements & overall system

A General Design Model



Lectures Preview

- Lecture 1: Introduction
  - Concepts of software architecture
- Lecture 2-4: Quality Attributes and Architecture Tactics
  - Quality attributes, NFRs and scenarios
  - Tactics achieving quality attributes
  - Architecturally significant requirements
- Lecture 5/6: Architecture Patterns
  - Patterns in software design
  - Architecture styles and patterns
- Lecture 7/8: Architecture Design Methods
  - Design strategies in general
  - Software design process and architecting process
  - ADD method: Attribute Driven Design

Architecture Specifies Communication

- Communication involves:
  - Data passing mechanisms, for example:
    - Function call
    - Remote method invocation
    - Asynchronous message
  - Control flow
    - Flow of messages between components to achieve required functionality
    - Sequential
    - Concurrent/parallel
    - Synchronization

Structure and Architecture

Architecture vs. Structure

- Decomposition of system into components/modules/ subsystems
- Architecture defines:
  - Component interfaces
    - What a component can do?
  - Component communications and dependencies
    - How components communicate?
  - Component responsibilities
    - Precisely what a component will do when you ask it?

Software Design Process

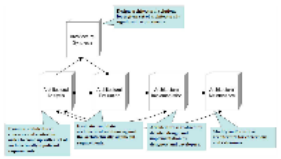
Software Design & Architecture Knowledge Areas

- Software Design: Quality Analysis and Evaluation
  - Quality attributes
  - Quality analysis and evaluation methods, techniques and tools
    - Design reviews (e.g. SEI's Architecture Trade-off Analysis Method)
    - Static analysis and dynamic analysis
    - Simulation and prototyping
  - Measures
    - Metrics: Architecture level
    - Technique specific measures
- Design Modeling and Representation
  - Architecture and Design Notations (Architecture Description Languages (ADL))
  - Unified Modeling Language (UML)
  - Design Documentation (Views & Beyond)
  - Others: differ in ability, focus and domain (e.g. ACME, Rapide)

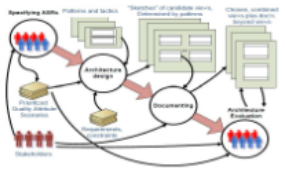
Software Design & Architecture Knowledge Areas

- Software Design Basic Concepts
  - General design concepts
  - Context: software development life cycle - requirements, design, construction and testing
  - Design process (role, activity, work product)
  - Enabling techniques for software design
- Key Issues (technical): concurrency, control and handling of events, distribution, exception handling, interactive systems, persistence
- Software Structure and Architecture
  - Architecture Structures and viewpoints
  - Architectural styles and patterns (macro-architecture)
  - Design patterns (micro-architecture)
- Software Design Methods
  - Architecture Methods (e.g., Attribute-Driven Design)
  - Design Methods (e.g., Dynamic System Development Method)

Architecture Lifecycle



Software Architecture Process



Architecture Activities

- Creating the business case for the System
- Understanding the requirements
- Creating and selecting architecture
- Communicating the architecture (stakeholders including developers)
- Analysing or evaluating the architecture
  - Overall methodologies
  - Quality specific techniques
- Implementing the architecture
- Ensuring conformance to an architecture