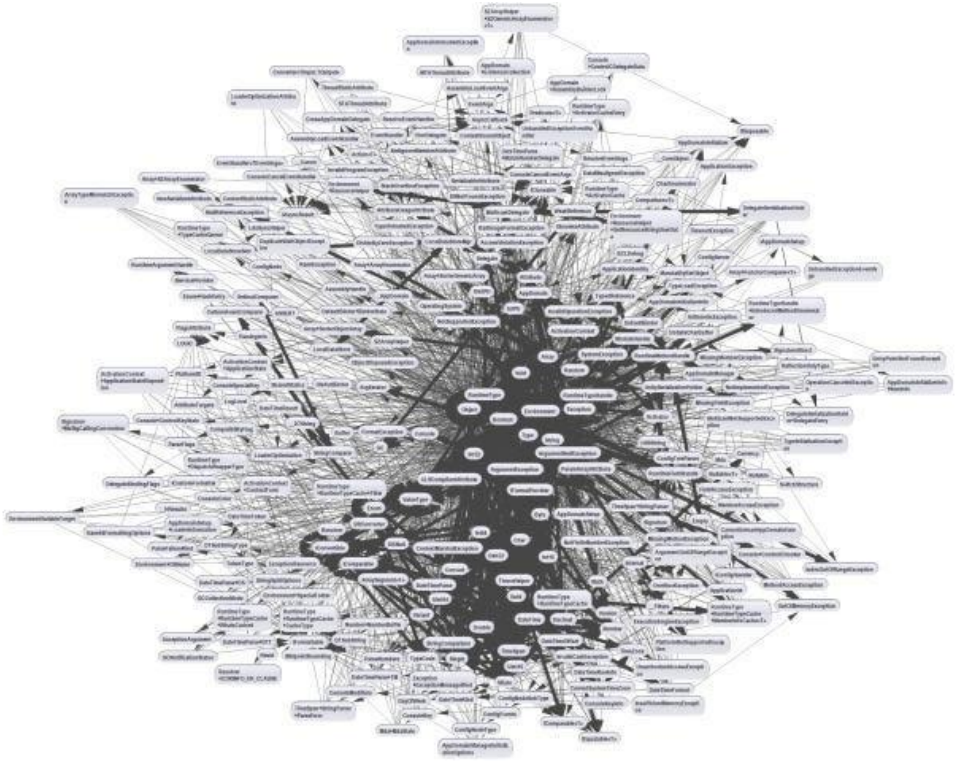
CS3213: Foundations of Software Engineering

Architecture Overview and Approaching Architecture

Coding After Gaining the Requirements

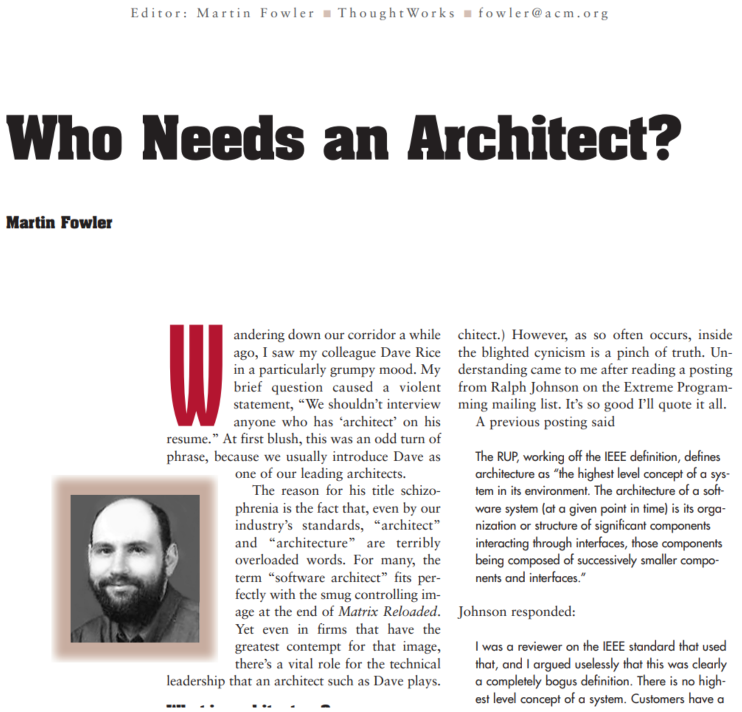
Image by Gemini

Big Ball of Mud



https://medium.com/@e0310443/the-big-ball-of-mud-amplifying-the-importance-of-software-engineering-principles-6e2eac5979cb

What’s Architecture?



Editor: Martin Fowler ■ ThoughtWorks ■ fowler@acm.org

Who Needs an Architect?

Martin Fowler

Wandering down our corridor a while ago, I saw my colleague Dave Rice in a particularly grumpy mood. My brief question caused a violent statement, “We shouldn’t interview anyone who has ‘architect’ on his resume.” At first blush, this was an odd turn of phrase, because we usually introduce Dave as one of our leading architects.

The reason for his title schizophrenia is the fact that, even by our industry’s standards, “architect” and “architecture” are terribly overloaded words. For many, the term “software architect” fits perfectly with the smug controlling image at the end of *Matrix Reloaded*. Yet even in firms that have the greatest contempt for that image, there’s a vital role for the technical leadership that an architect such as Dave plays.

chitect.) However, as so often occurs, inside the blighted cynicism is a pinch of truth. Understanding came to me after reading a posting from Ralph Johnson on the Extreme Programming mailing list. It’s so good I’ll quote it all.

A previous posting said

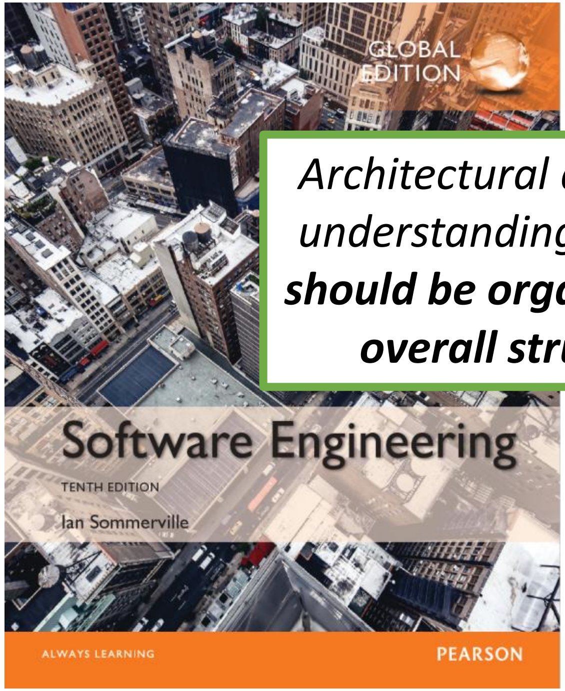
> The RUP, working off the IEEE definition, defines architecture as “the highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces.”

Johnson responded:

> I was a reviewer on the IEEE standard that used that, and I argued uselessly that this was clearly a completely bogus definition. There is no highest level concept of a system. Customers have a

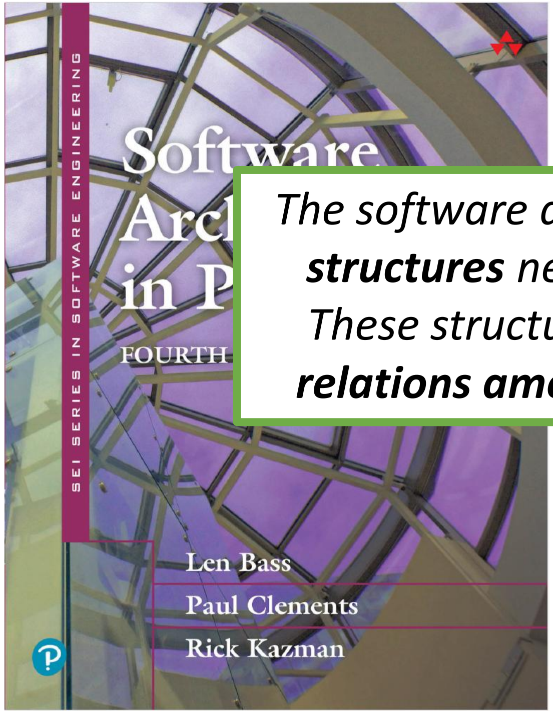
https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

What’s Architecture?



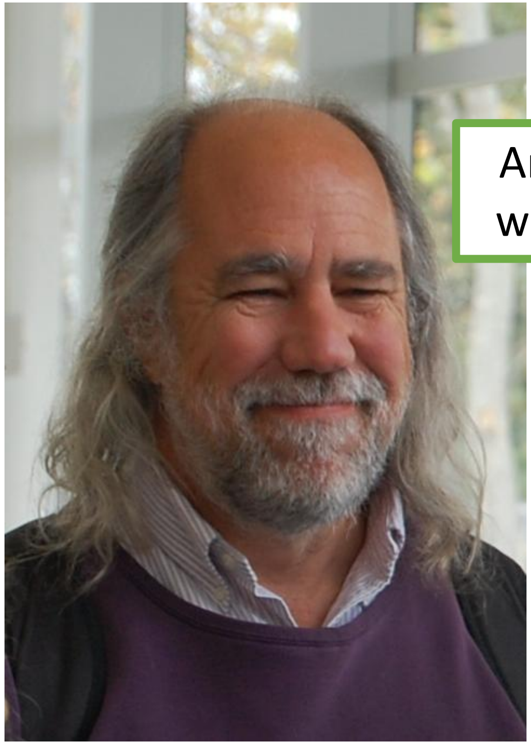
Architectural design is concerned with understanding how a software system ***should be organized*** *and* ***designing the overall structure*** *of that system*

What’s Architecture?



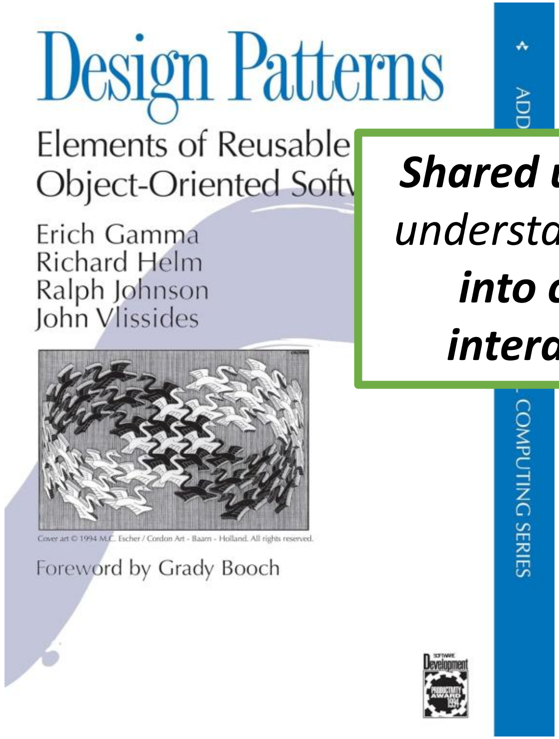
The software architecture of a system is the ***set of structures*** *needed to reason about the system. These structures* ***comprise software elements, relations among them, and properties of both.***

What’s Architecture?



Architecture represents the **significant decisions**, where significance is measured **by cost of change**

What's Architecture?



***Shared understanding of the system design**. This understanding includes **how the system is divided into components** and how the **components interact through interfaces**. (Ralph Johnson)*

What’s Architecture?

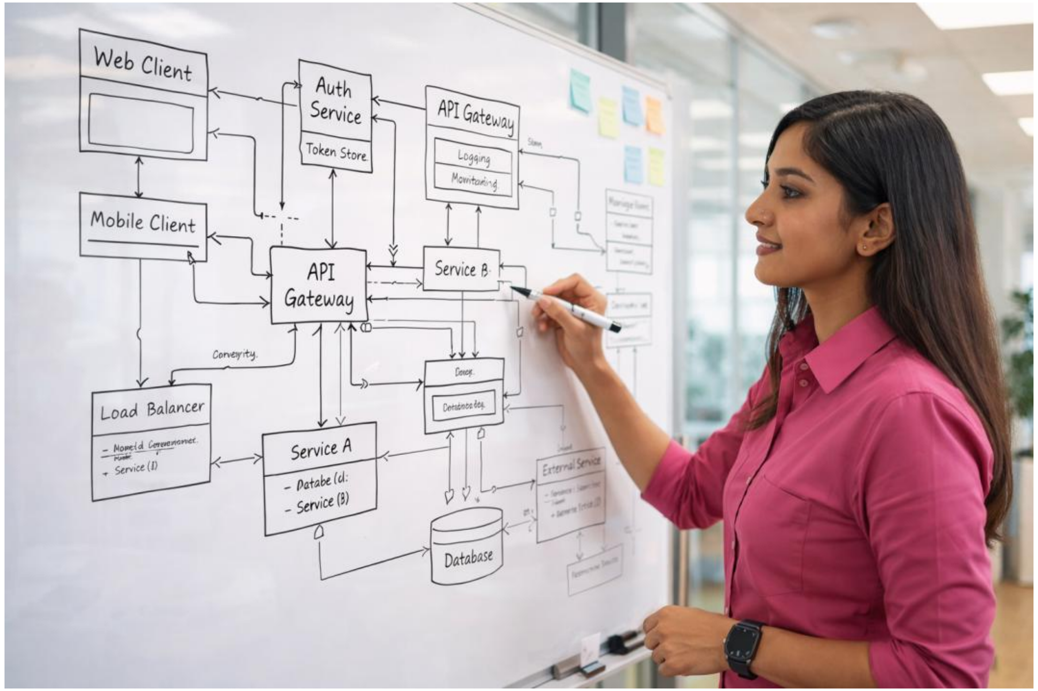


Image by ChatGPT

Three Kind of Structures

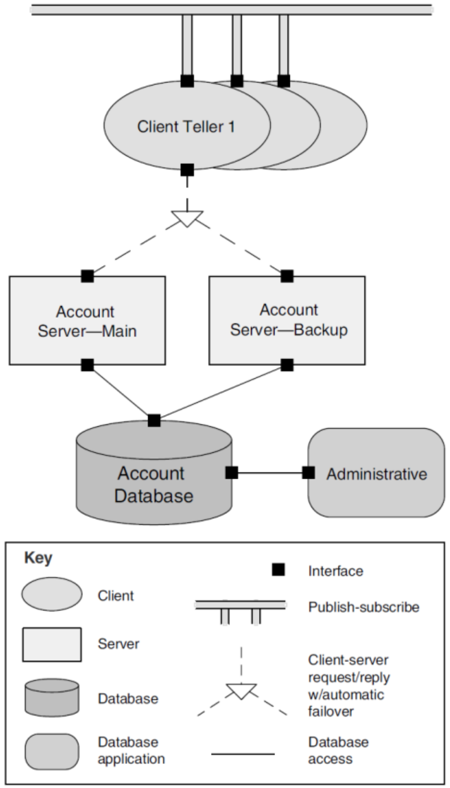
- *Component-and-connector (C&C) structures*
- *Module structures*
- *Allocation structures*

Three Kind of Structures

- *Component-and-connector (C&C) structures*
- *Module structures*
- *Allocation structures*

Focus on the way components interact with each other at run time

Component-and-connector (C&C)

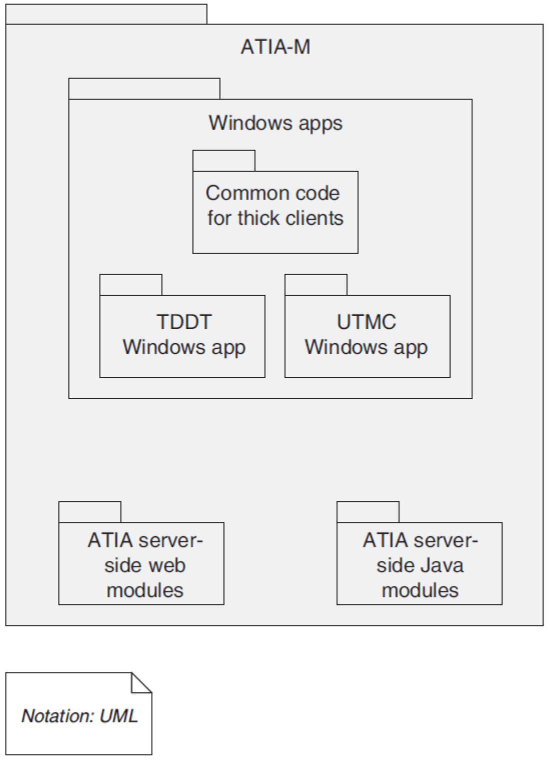


Three Kind of Structures

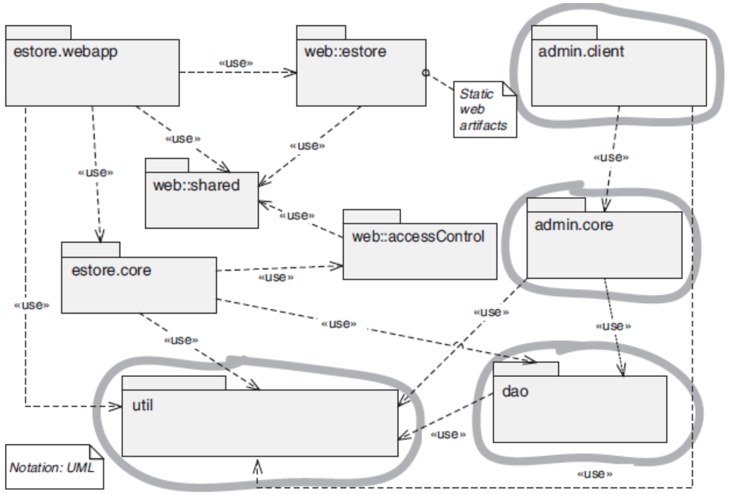
- *Component-and-connector (C&C) structures*
- *Module structures*
- *Allocation structures*

Partition systems into implementation units (modules)

Module Structures: Decomposition Structure


ATIA-M
Windows apps
Common code for thick clients
TDDT Windows app
UTMC Windows app
ATIA server-side web modules
ATIA server-side Java modules
Notation: UML

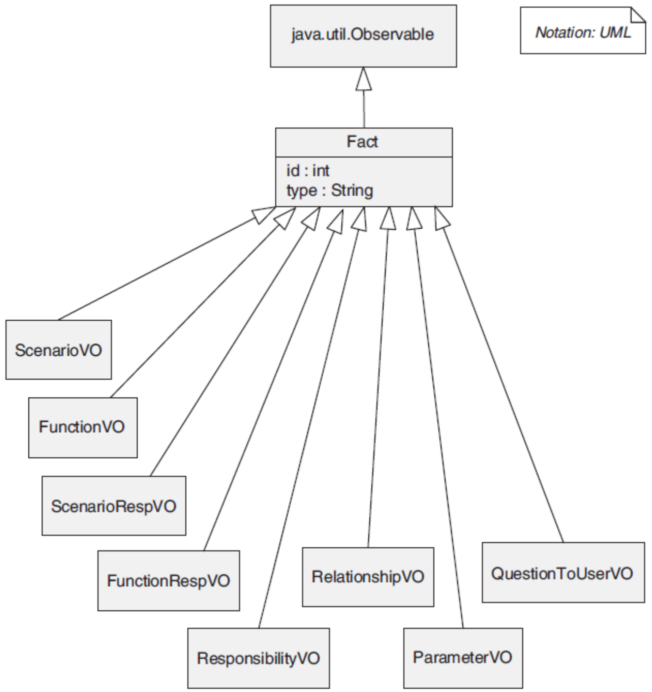
Module Structures: Uses Structure



Conway’s Law and Architecture

Conway’s law: [O]rganizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.

Module Structures: Generalization Structure

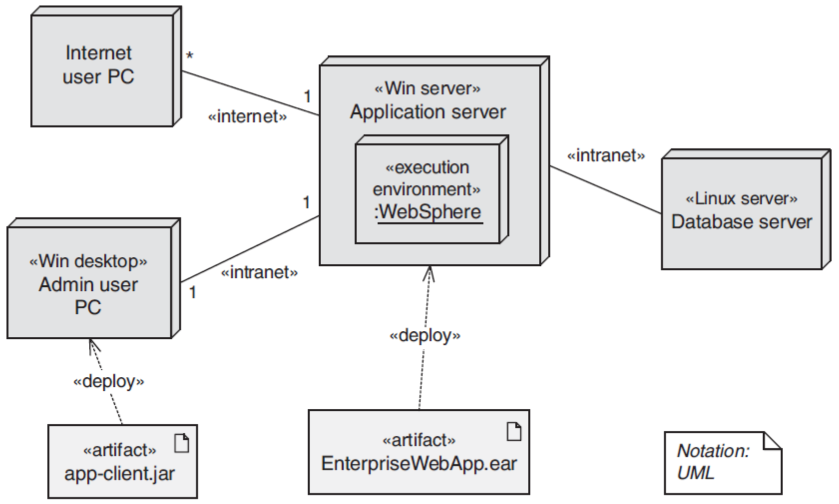


Three Kind of Structures

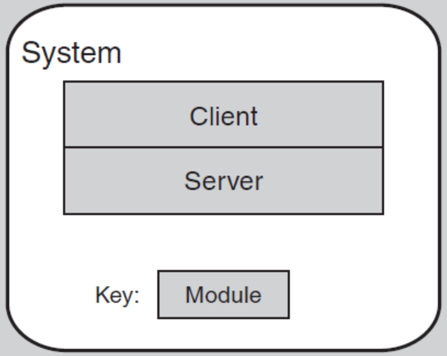
- *Component-and-connector (C&C) structures*
- *Module structures*
- *Allocation structures*

How software is assigned to hardware and communication pathways

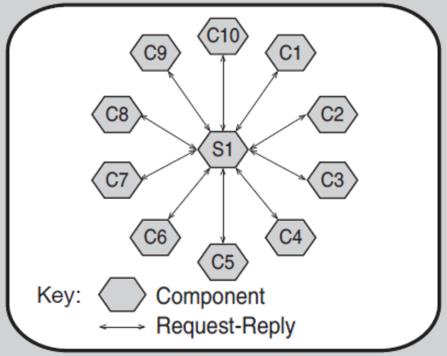
Allocation Structures: Deployment Structure


Internet user PC
*
1
«internet»
«Win server» Application server
«execution environment» :WebSphere
«intranet»
«Linux server» Database server
1
«Win desktop» Admin user PC
«intranet»
1
«deploy»
«deploy»
«artifact» app-client.jar
«artifact» EnterpriseWebApp.ear
Notation: UML

Which Structure(s) to Model?

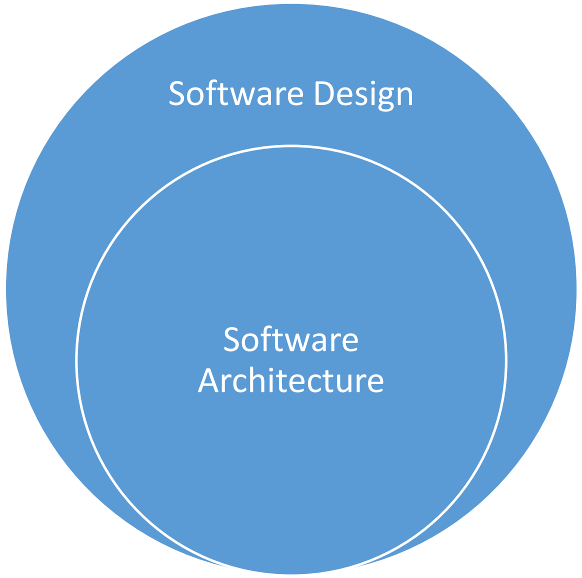


Decomposition View

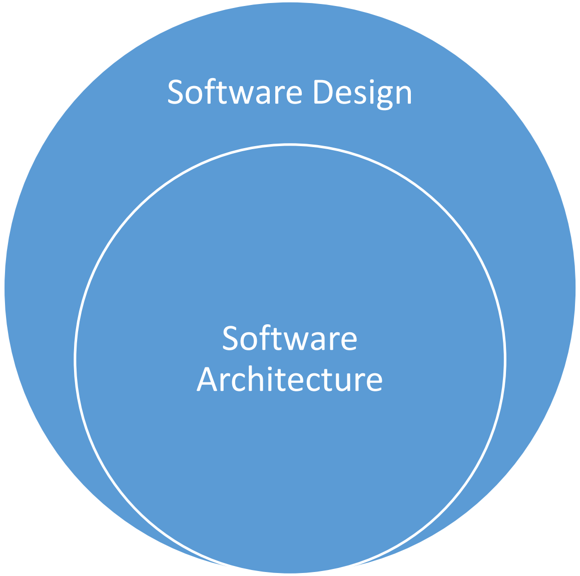


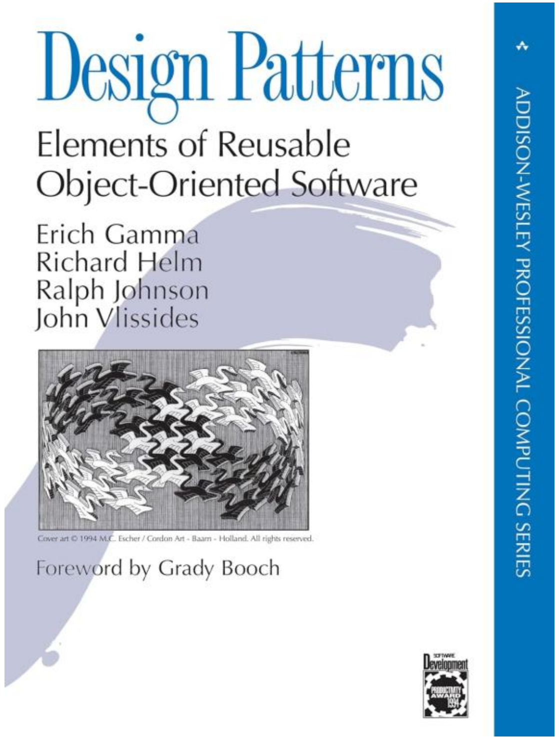
Client-Server View

Architecture vs. Design



Architecture vs. Design




Design Patterns

Elements of Reusable Object-Oriented Software

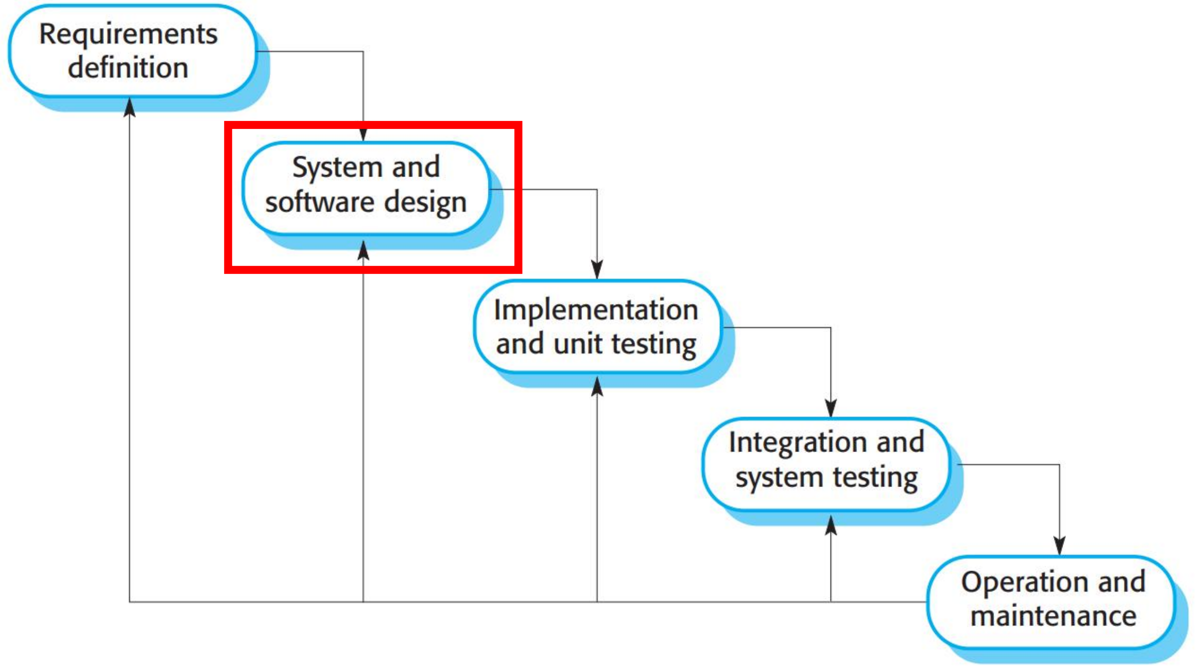
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

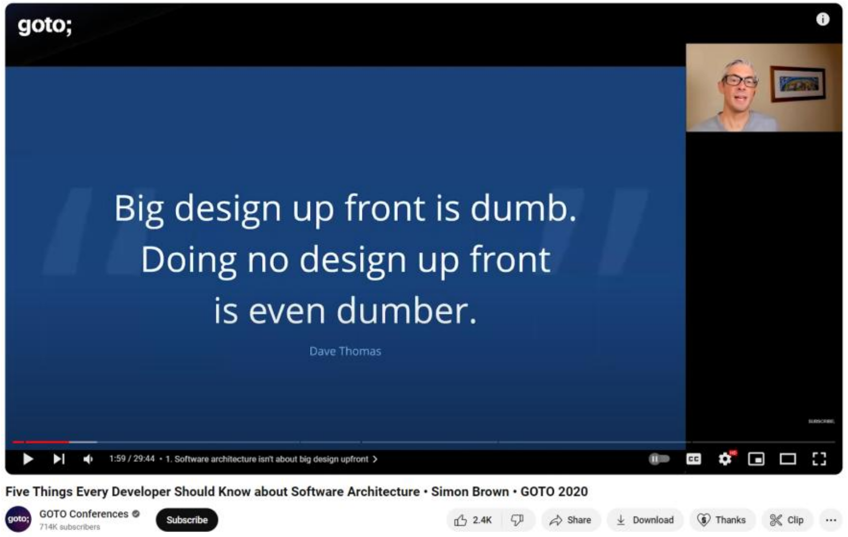
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Architecture in the Waterfall Model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

System Architecture in Agile Methods

- Agile mindset: architectural design in early stages
- Common perspective: **mitigate** highest priority **risks**



Big design up front is dumb.
Doing no design up front
is even dumber.

1:59 / 29:44 • 1. Software architecture isn't about big design upfront

Five Things Every Developer Should Know about Software Architecture • Simon Brown • GOTO 2020

https://www.youtube.com/watch?v=9Az0q2XHtH8

Architecturally Significant Requirements (ASRs)

“Temperature should be displayed in Celsius not Fahrenheit on this webpage”

Architecturally Significant Requirements (ASRs)

“the system should provide five nines (99.999 percent) availability”

Why Architecture?

Reasons: Quality Attributes

- Whether or not a system meets its quality attributes (i.e., non-functional requirements) is significantly influenced by its architecture
- Example 1: **High performance** requires managing time-based behavior of components and their access to shared resources
- Example 2: **Modifiability** requires assigning responsibilities to components and limit their interaction, so that a change ideally affects only a single component
- Example 3: **Safe and secure** systems require safeguards and recovery mechanisms

Reasons: Modifiability and Reasoning about Change

- **Modifiability** is a quality attribute, but one of the **most common and important** ones
- Three categories of changes
  - **Local change**: only single component is affected (e.g., adding a new rule to a pricing module)
  - **Non-local change**: multiple elements are affected (e.g., besides the rule, database and UI needs to be changed)
  - **Architectural change**: affects how the elements interact and might require changes over the whole system (e.g., changing a system from single-threaded to multi-threaded)

Reasons: Communication Among Stakeholders

- Common abstraction of the system
- Can be used as a basis for mutual understanding, negotiating, forming consensus, and communicating with each other
- Also non-technical people are likely to understand the architecture to the extent they need to

How to Approach Architecture?

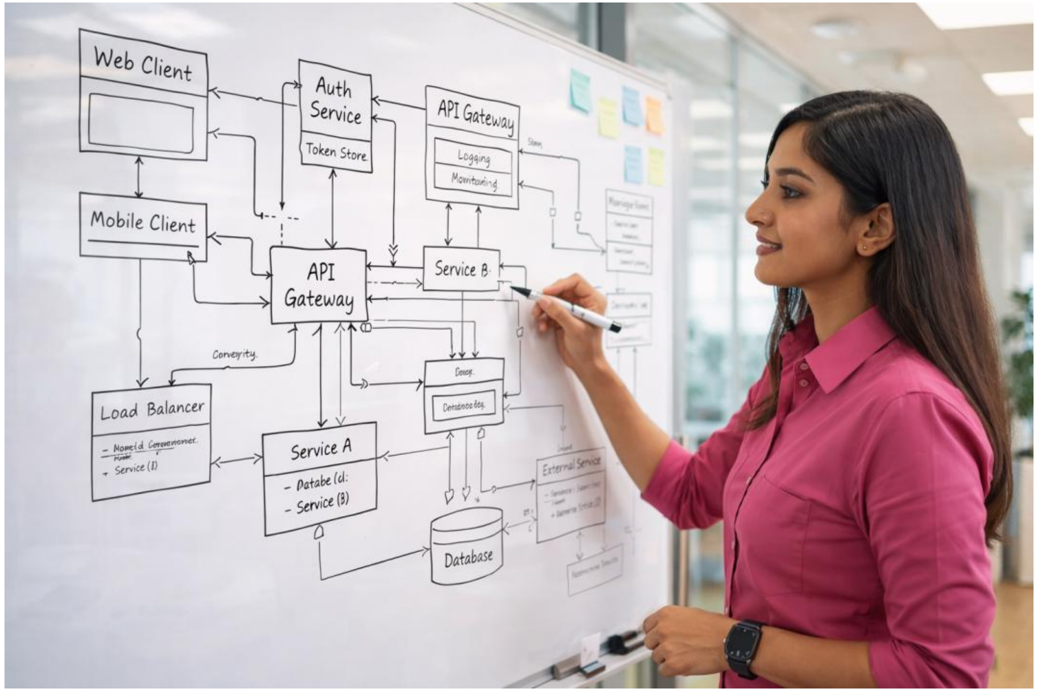
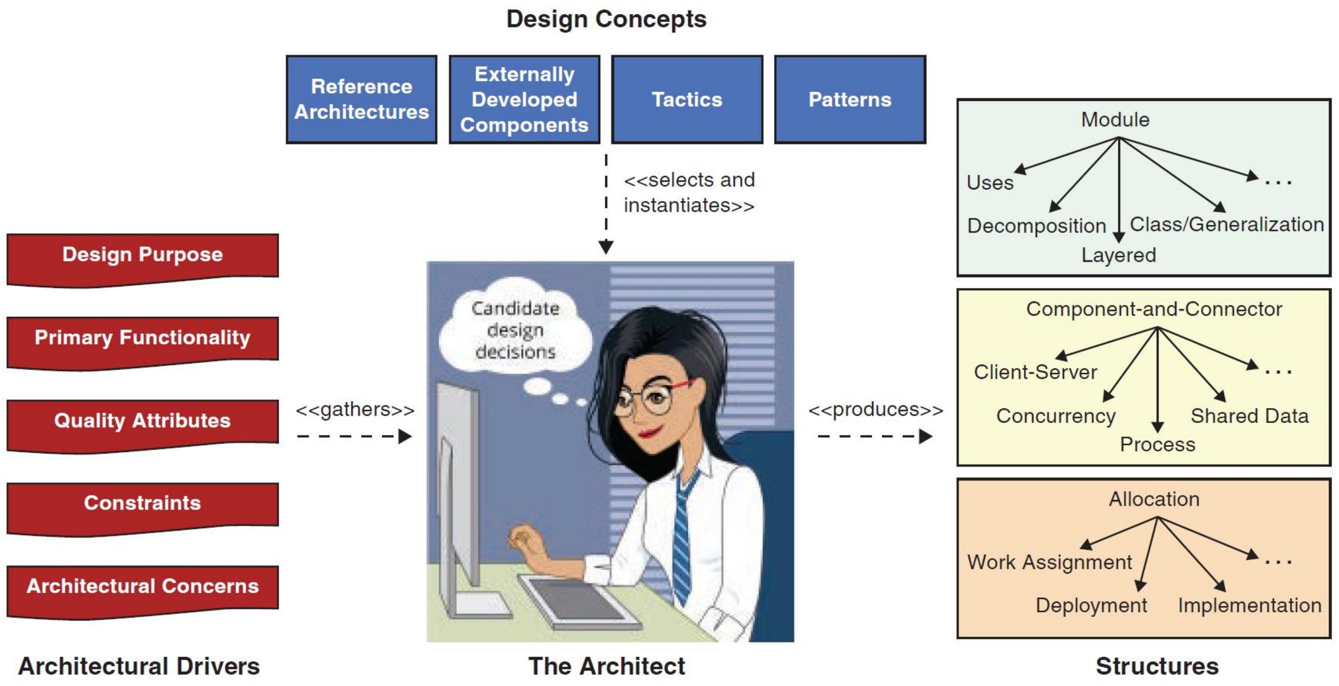
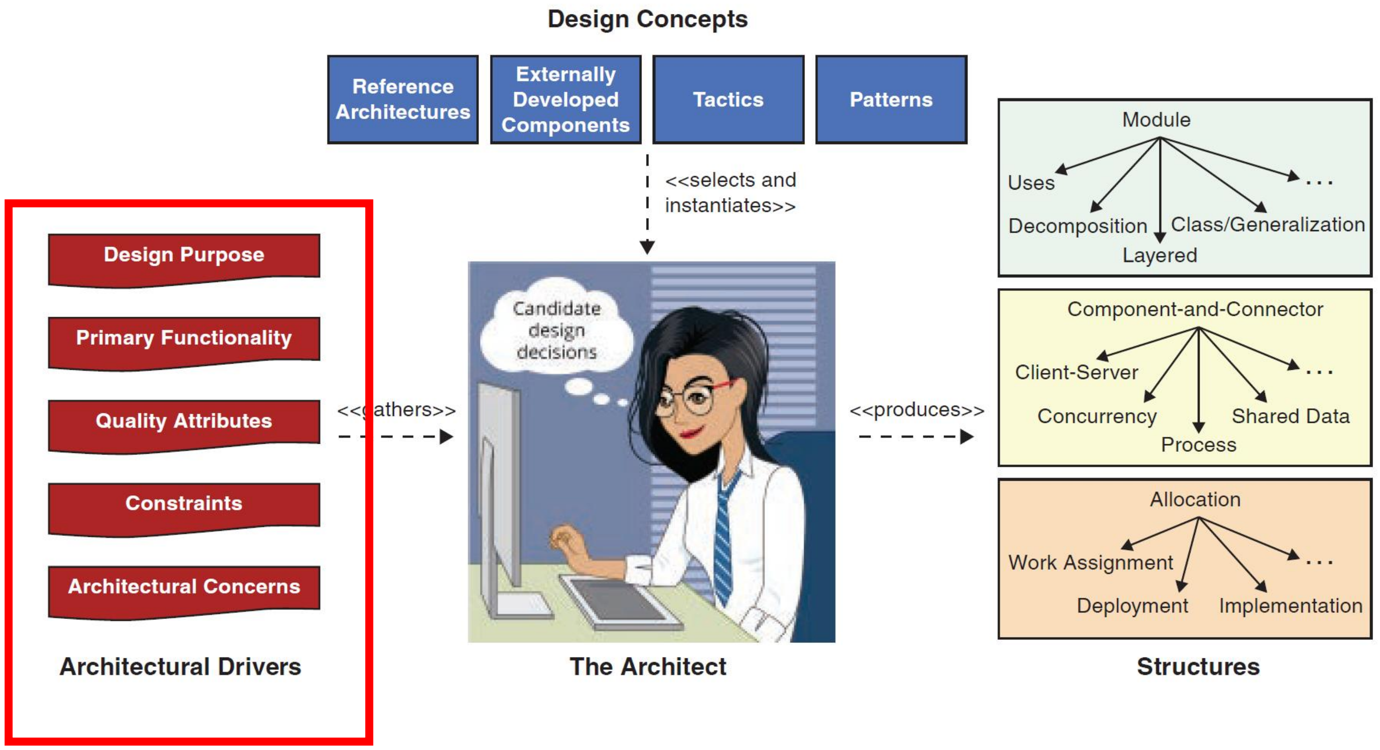


Image by ChatGPT

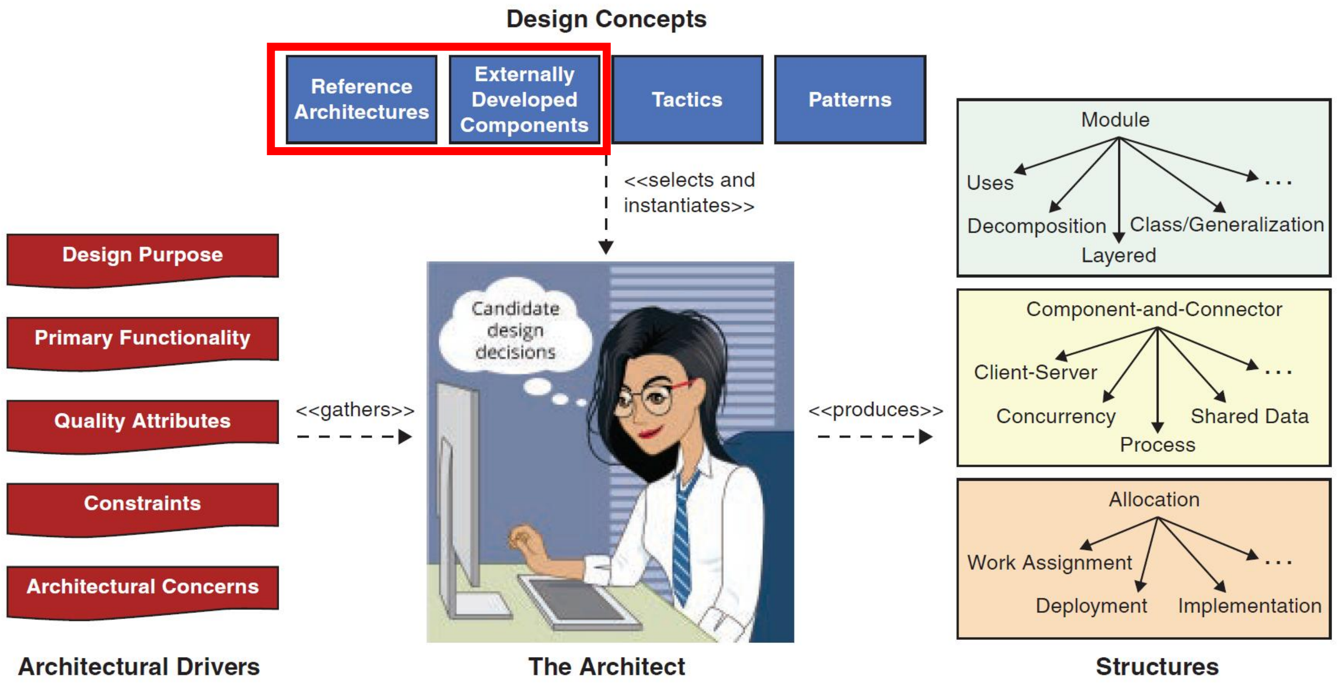
Approaching Architecture



Approaching Architecture

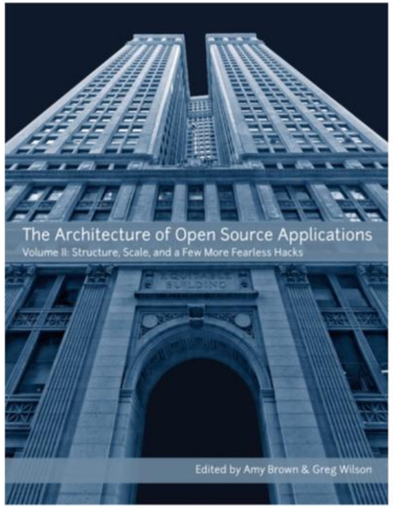


Approaching Architecture



The Architecture of Open Source Applications

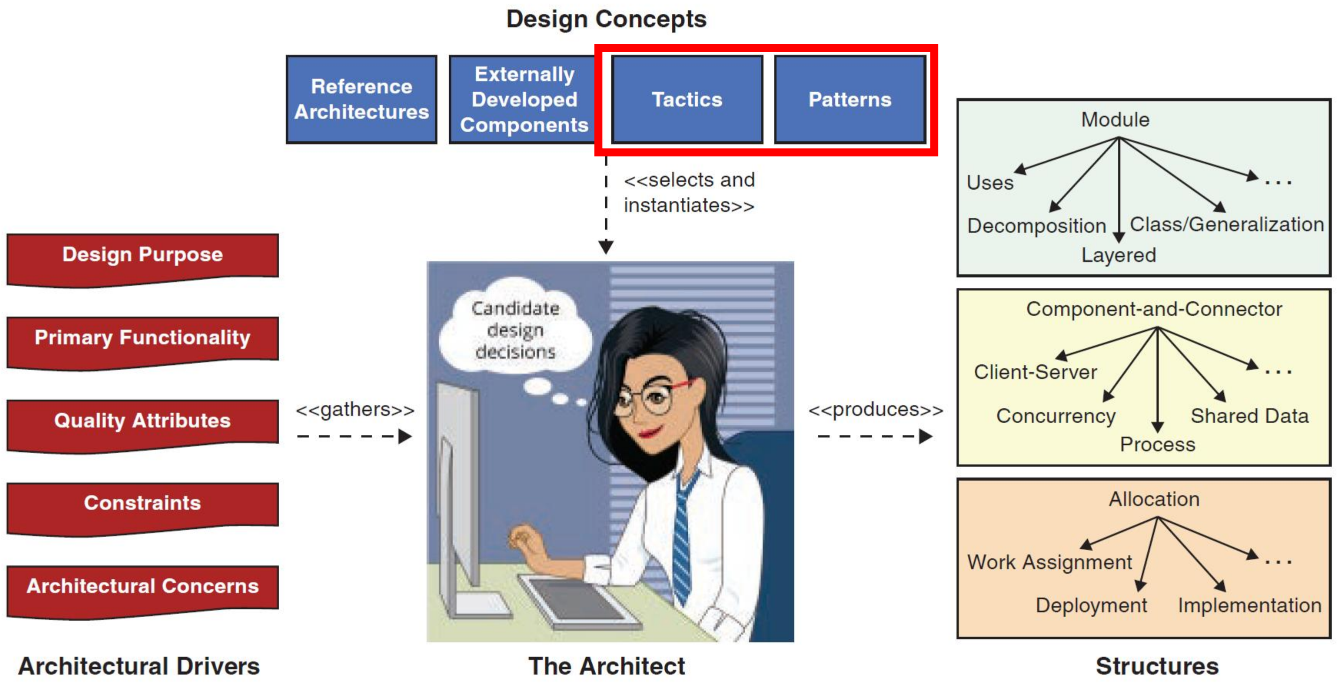
AOSA Volume 2


The Architecture of Open Source Applications
Volume II: Structure, Scale, and a Few More Fearless Hacks
Edited by Amy Brown & Greg Wilson

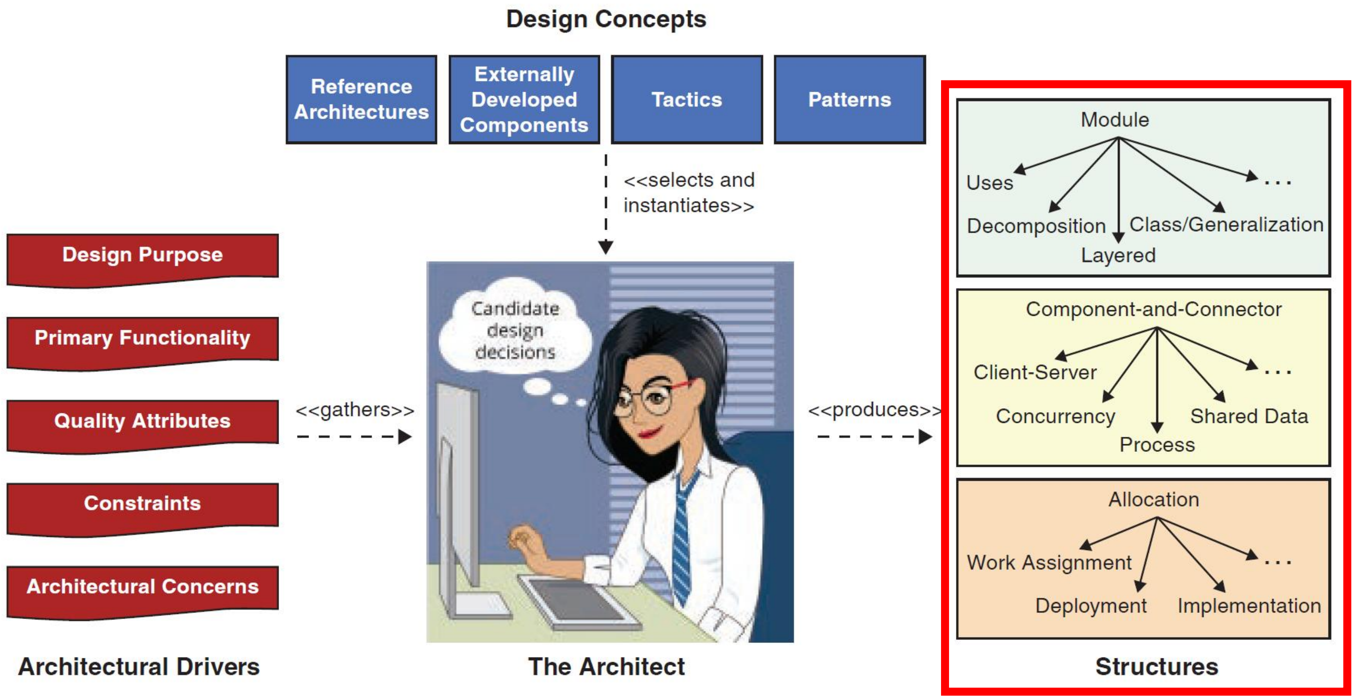
Buy Volume II

https://aosabook.org/en/

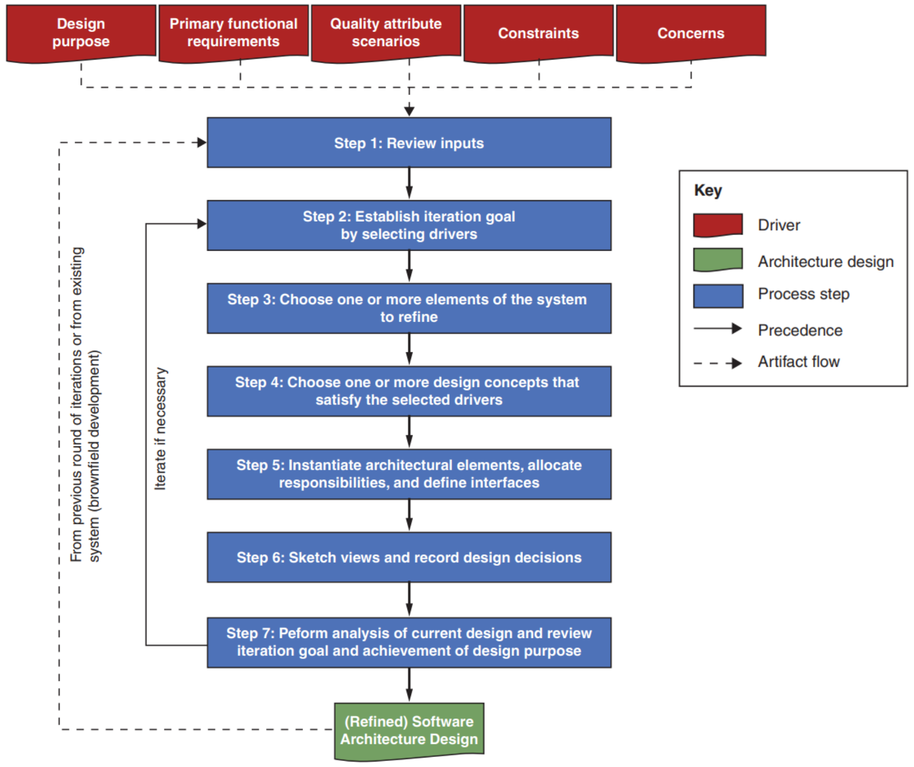
Approaching Architecture



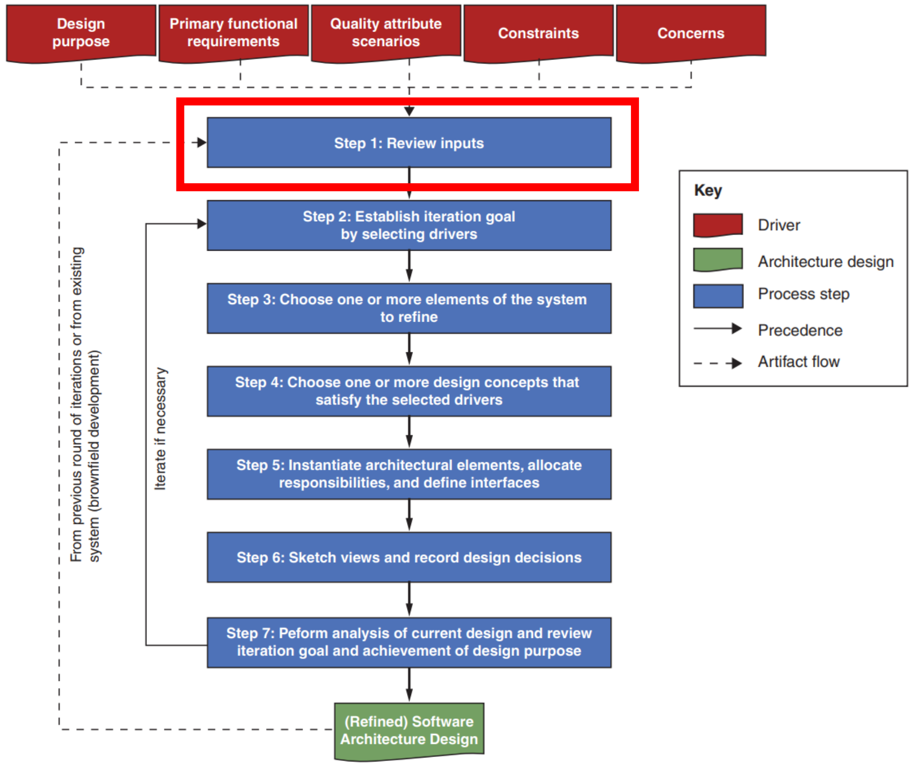
Approaching Architecture



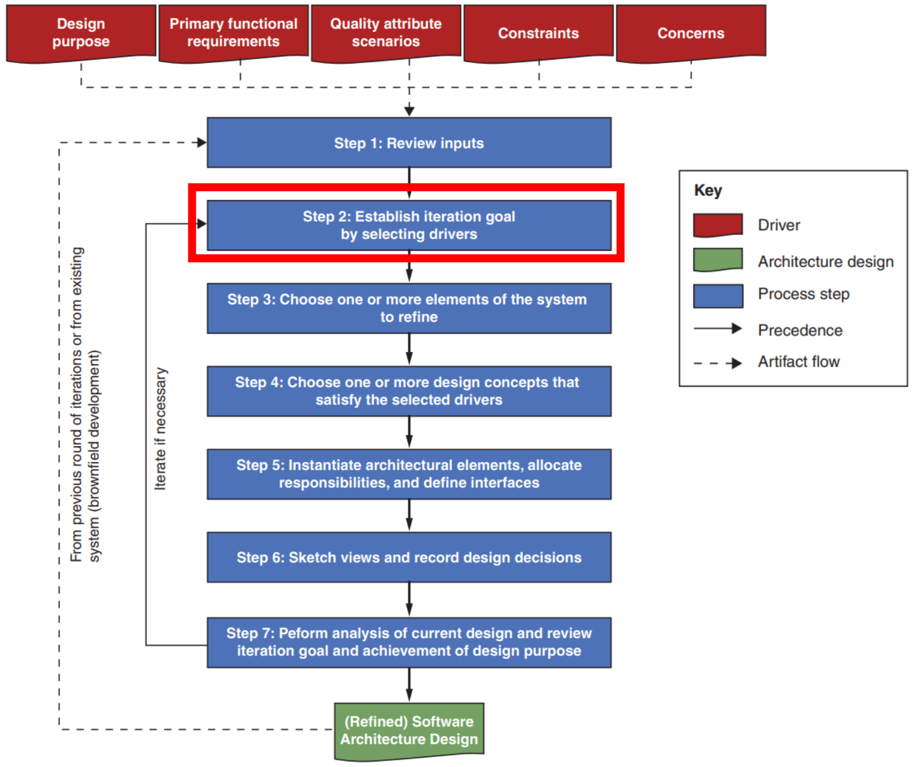
Attribute-Driven Design (ADD)


Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
From previous round of iterations or from existing system (brownfield development)
Iterate if necessary
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

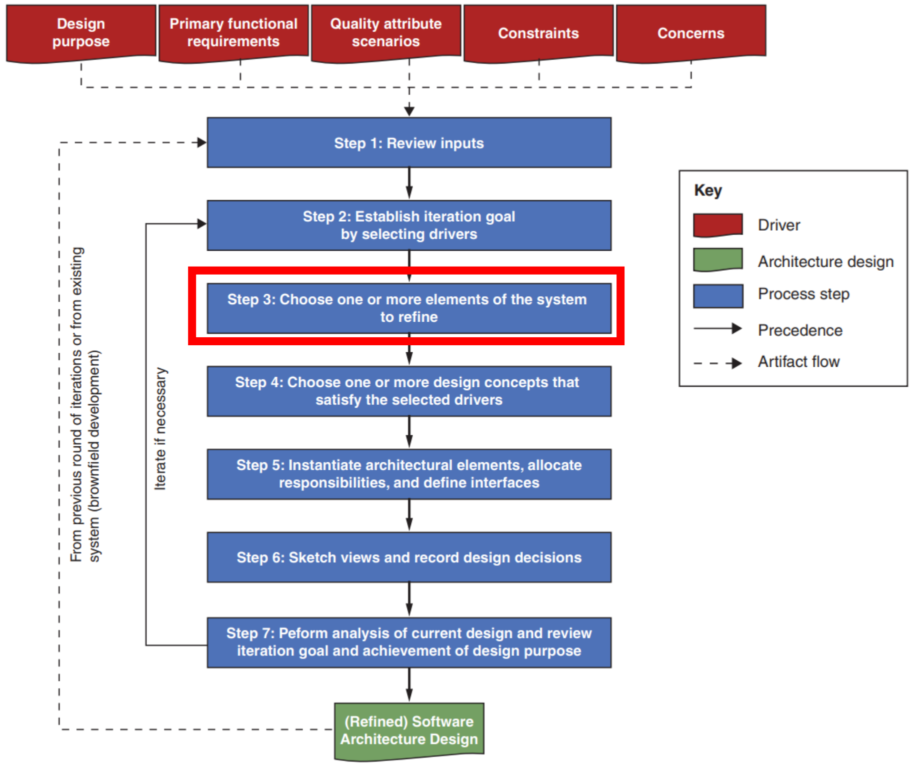
Attribute-Driven Design (ADD)


Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
From previous round of iterations or from existing system (brownfield development)
Iterate if necessary
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

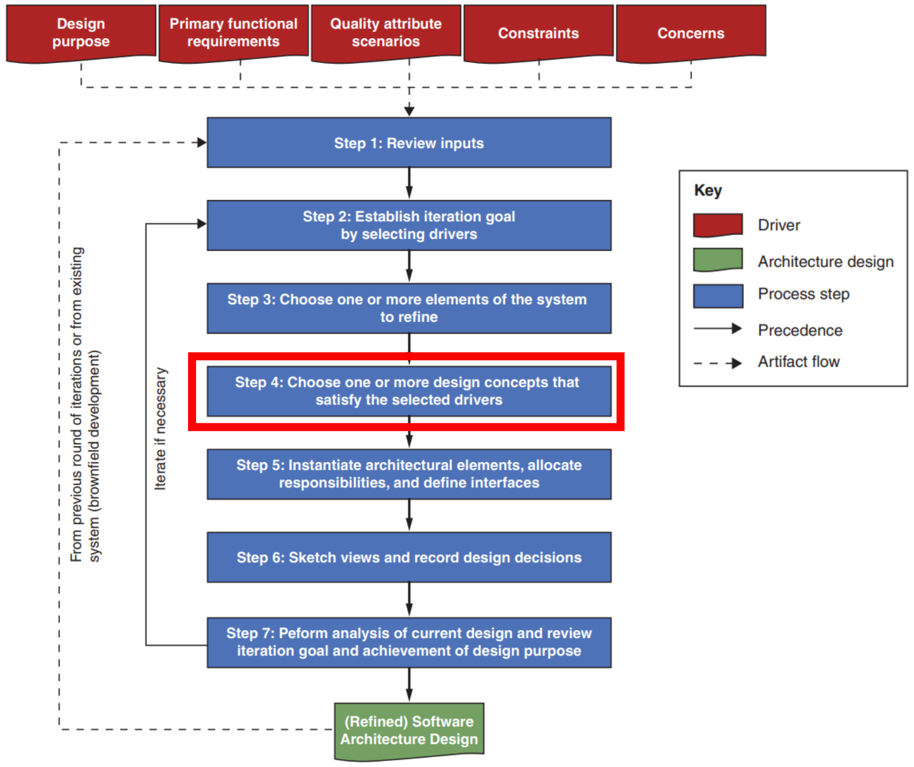
Attribute-Driven Design (ADD)


Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
From previous round of iterations or from existing system (brownfield development)
Iterate if necessary
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

Attribute-Driven Design (ADD)

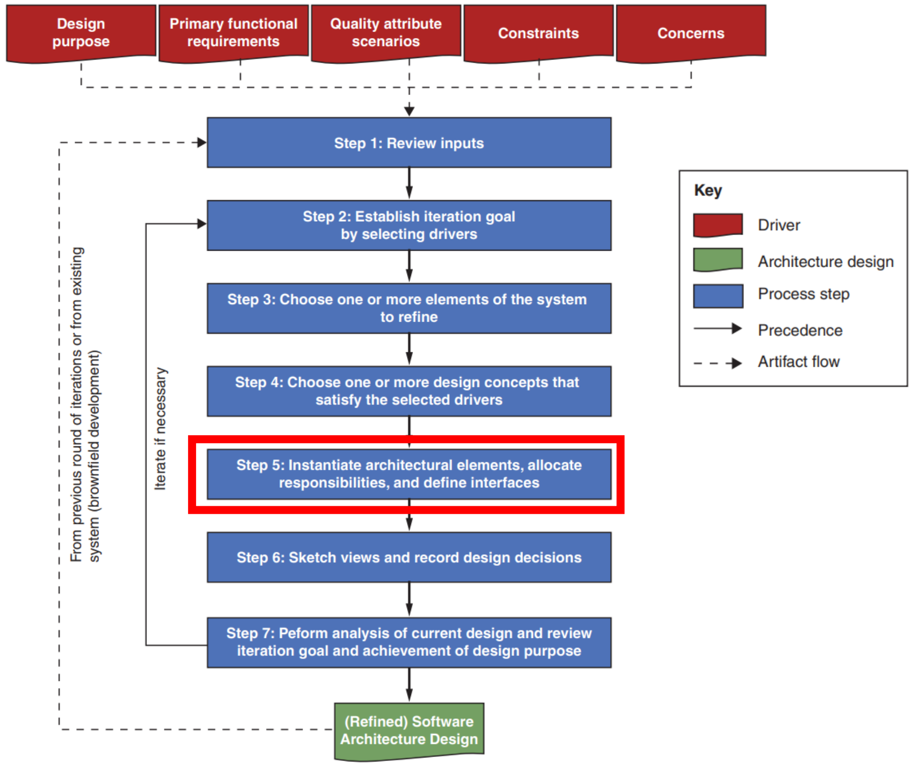

Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
From previous round of iterations or from existing system (brownfield development)
Iterate if necessary
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

Attribute-Driven Design (ADD)

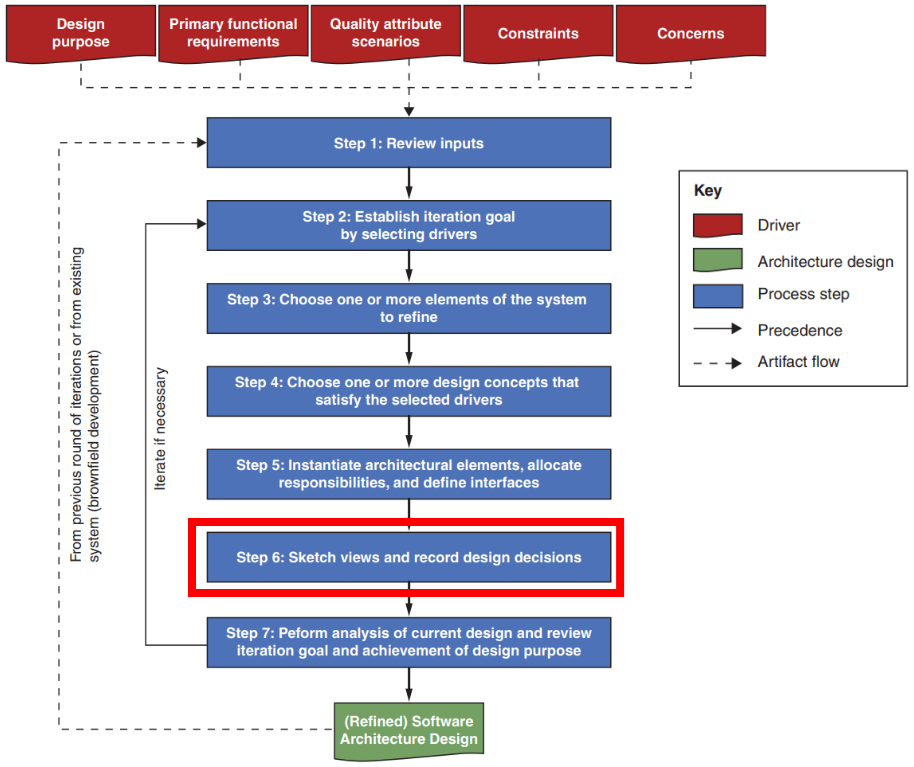


Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
Iterate if necessary
From previous round of iterations or from existing system (brownfield development)
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

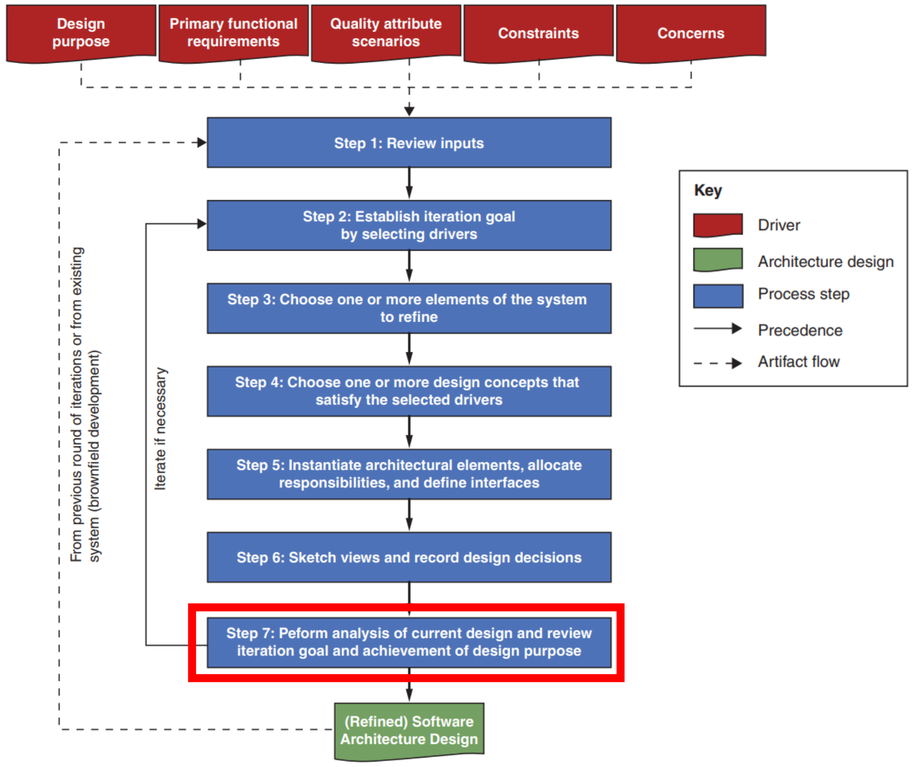
Attribute-Driven Design (ADD)


Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
From previous round of iterations or from existing system (brownfield development)
Iterate if necessary
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

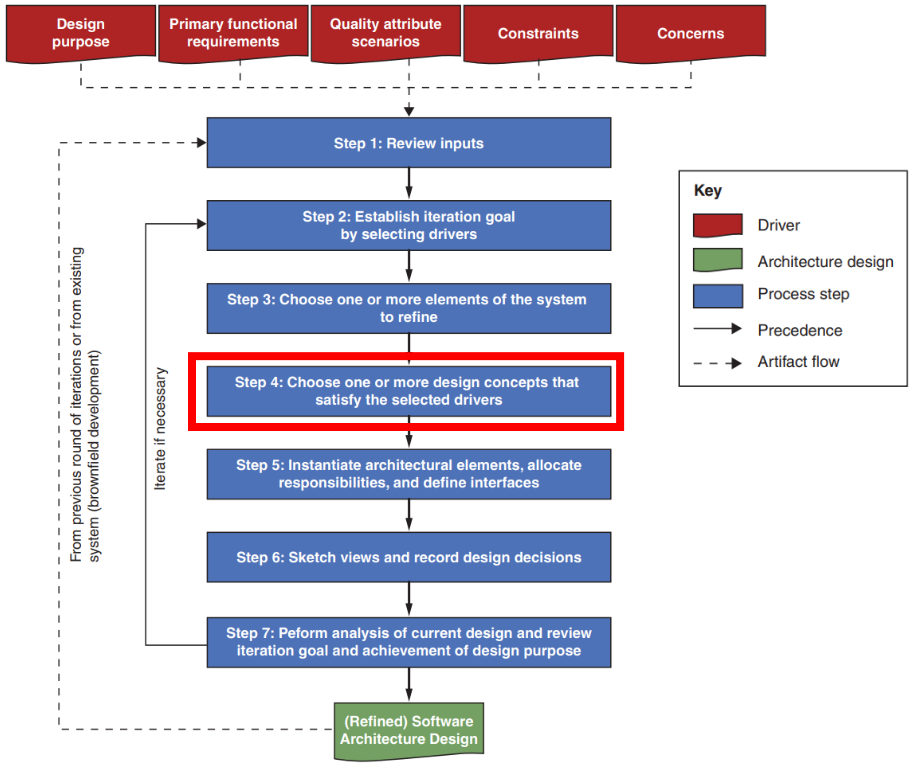
Attribute-Driven Design (ADD)


Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
From previous round of iterations or from existing system (brownfield development)
Iterate if necessary
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

Attribute-Driven Design (ADD)


Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
Iterate if necessary
From previous round of iterations or from existing system (brownfield development)
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

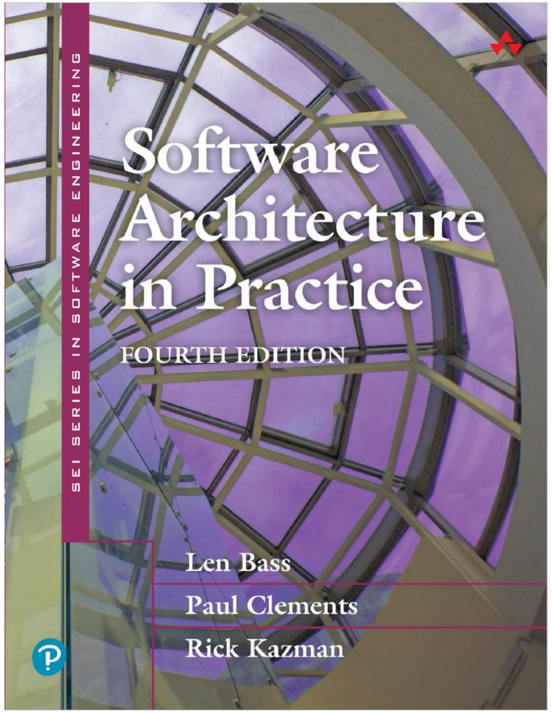
Attribute-Driven Design (ADD)

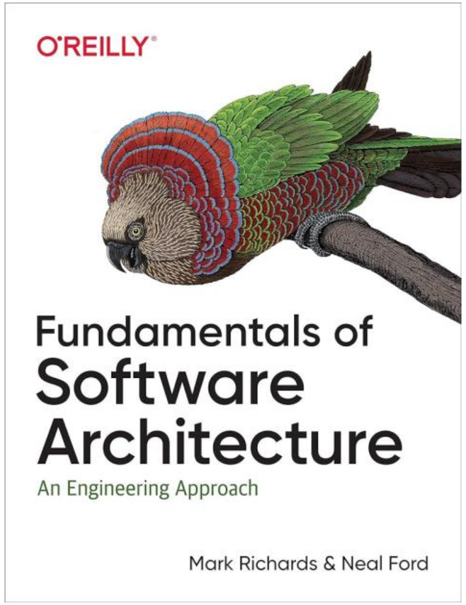

Design purpose
Primary functional requirements
Quality attribute scenarios
Constraints
Concerns
Step 1: Review inputs
Step 2: Establish iteration goal by selecting drivers
Step 3: Choose one or more elements of the system to refine
Step 4: Choose one or more design concepts that satisfy the selected drivers
Step 5: Instantiate architectural elements, allocate responsibilities, and define interfaces
Step 6: Sketch views and record design decisions
Step 7: Peform analysis of current design and review iteration goal and achievement of design purpose
(Refined) Software Architecture Design
Iterate if necessary
From previous round of iterations or from existing system (brownfield development)
Key
Driver
Architecture design
Process step
Precedence
Artifact flow

Summary and Key Points

- What is architecture? Why architecture?
- Different structures
- Approaching architecture
- Attribute-Driven Design (ADD)
- Conway’s Law

Architecture: Sources and References


SEI SERIES IN SOFTWARE ENGINEERING
Software Architecture in Practice
FOURTH EDITION
Len Bass
Paul Clements
Rick Kazman


O'REILLY
Fundamentals of Software Architecture
An Engineering Approach
Mark Richards & Neal Ford