Software Architecture

CSC 326



img

© NC State Software Engineering Faculty

1

Overview

- Architecture is a subset of Design
  - Architecture deals with the external properties of the parts of a system
  - Design deals with both the external and internal properties of the parts

2

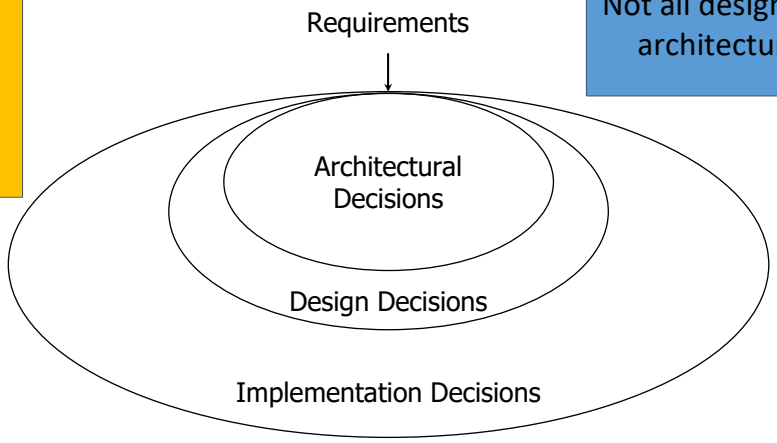
2

All Architectural Decisions are Design Decisions

Architecture is a subset of Design

All architecture decisions are design decisions.

Not all design decisions are architecture decisions



Abrahamsson, Ali Babar, and Kruchten [2010]

3

3

Intuitive Notion of Architecture

- Like software, architecture is invisible; we work with descriptions
- Descriptions are called views
- A view identifies
  - (a) the parts or elements of a system and
  - (b) how the elements work together
- Views are used to understand and reason about some aspect of a system

4

4

Example (6.1 in text): Spell Checker

- *Spell-check* is responsible for reading a stream of characters that contain punctuation, numbers, and formatting commands, It produces a stream of words that are **not** in a dictionary. (here, a word is a contiguous sequence of characters)
- *Split* is responsible for removing unwanted characters. It produces a stream of words.
- *Lookup* is responsible for answering true or false to whether a word is in the dictionary

Spell-check	uses	*Split* to get words
Spell-check	uses	*Lookup* to check if words are in the dictionary

What can we learn from this about the relationship between Split and Lookup?

5

5

What is good architecture?

There are no hard and fast rules. Only guidelines.

- **Functional**: Will it meet the requirements? Will it be worth the cost?
- **Quality Attributes**: Will the system do its job well?
- **Elegance**: is the architecture easy to understand and update?

6

6

Design vs. Architecture

Design Questions

- How do I add a menu item in Eclipse?
- How can I make it easy to add menu items in Eclipse?
- What lock protects this data?
- How does Google rank pages?
- What encoder should I use for secure communication?
- What is the interface between objects?

Architectural Questions

- How do I extend Eclipse with a plugin?
- What threads exist and how do they coordinate?
- How does Google scale to billions of hits per day?
- Where should I put my firewalls?
- What is the interface between subsystems?

7

Slide adapted from Rohan Padhye and Michael Hilton

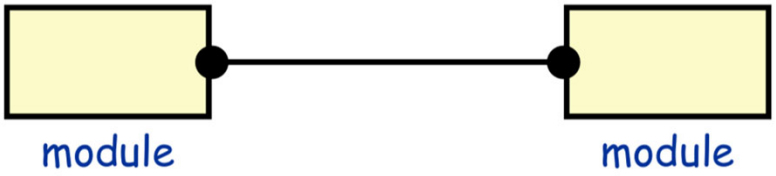
7

Module Design: Coupling and Cohesion

8

8

Architectural Building blocks: Modules



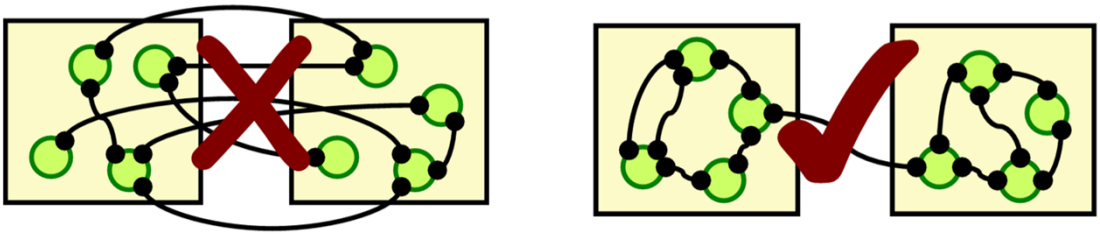
- What are your **criteria for module design**?
- A module is a grouping of program elements with
  - A specific responsibility
  - A well-defined interface

9

9

A good architecture

- **Minimizes coupling** between modules:
  - Goal: modules don't need to know much about one another to interact
  - Low coupling makes future change easier
- **Maximizes cohesion** of each module
  - Goal: the contents of each module are strongly interrelated
  - High cohesion makes a module easier to understand



from Steve Easterbrook, University of Toronto

10

10

Coupling: Aim for Loose Coupling

- Coupling is the degree to which modules are inter-related
  - Loosely coupled if interaction is only through interfaces
  - Tightly coupled if the implementation of one module depends on the implementation of another

11

11

Forms of Coupling

In order of increasing coupling

- Message: Send messages through interfaces
- Subclass: Inherit methods and data from a superclass
- Global: Two modules share global data
- Content: One module relies on the implementation of another

12

12

Again, we want to **minimize coupling!**



img

13

13

Information or Implementation Hiding

- Modularize to make a system easy to understand, build, test, and modify
- Put key design decisions in separate modules
  - So they can be changed independently
- Keep each module simple enough
  - So it can be understood fully
- Plan for change
  - So likely changes don't affect module interfaces

14

14

Cohesion: Aim for High Cohesion

- Cohesion is the degree to which the elements of a module belong together
- High cohesion if a module has one purpose and all element relate to the purpose
- Low cohesion if the module has multiple roles or elements that do not relate to a single purpose

Stevens, Myers, and Constantine [1974]

15

15

Forms of Cohesion

- **Functional**: elements relate to a single purpose
- **Sequential**: group elements based on processing steps
- **Data**: group based on data that is manipulated
- **Temporal**: group elements based on ordering (may not be for related activities)
- **Coincidental**: no reason for the elements to be grouped together

16

16

Again, we want to **maximize cohesion!**



img

17

17

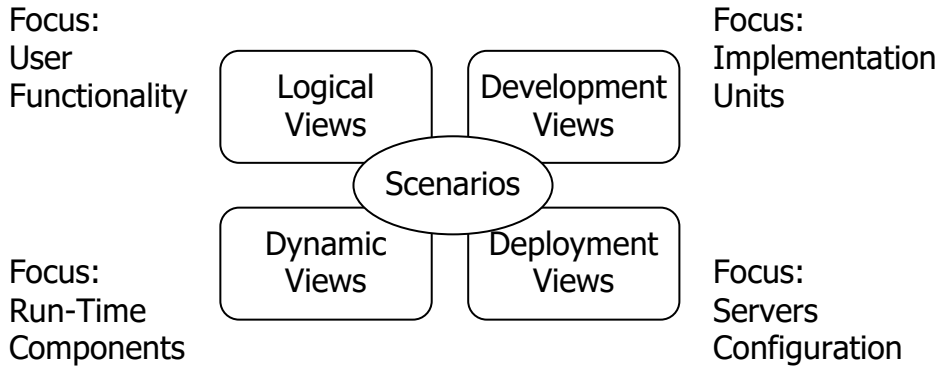
Architectural Views

18

18

4+1 Grouping of Views

Scenarios tie the views together



Kruchten [1995]

19

19

High level

- There is no “right” view of an architecture
- Choose a view to suit the intended audience and the purpose of the view
- The 4+1 grouping of views offers 4 kinds of views for consideration

20

20

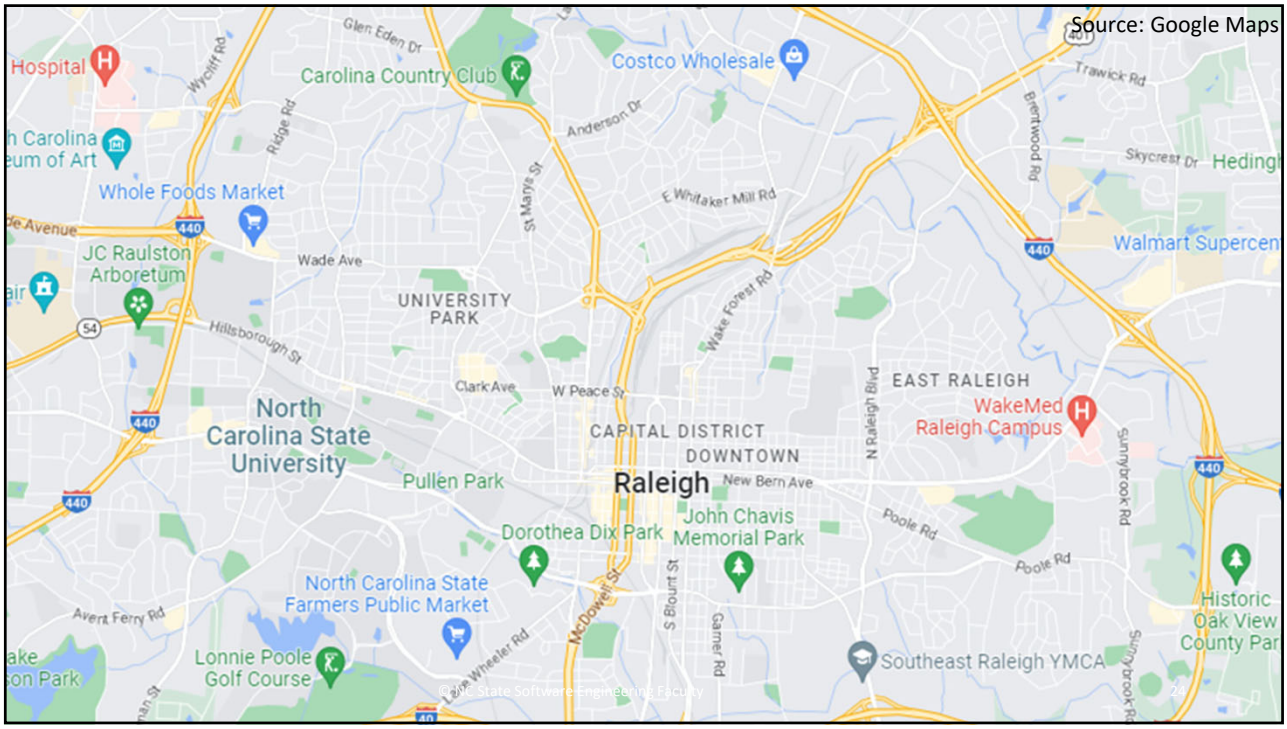
Views of an Architecture

One System, Multiple Views

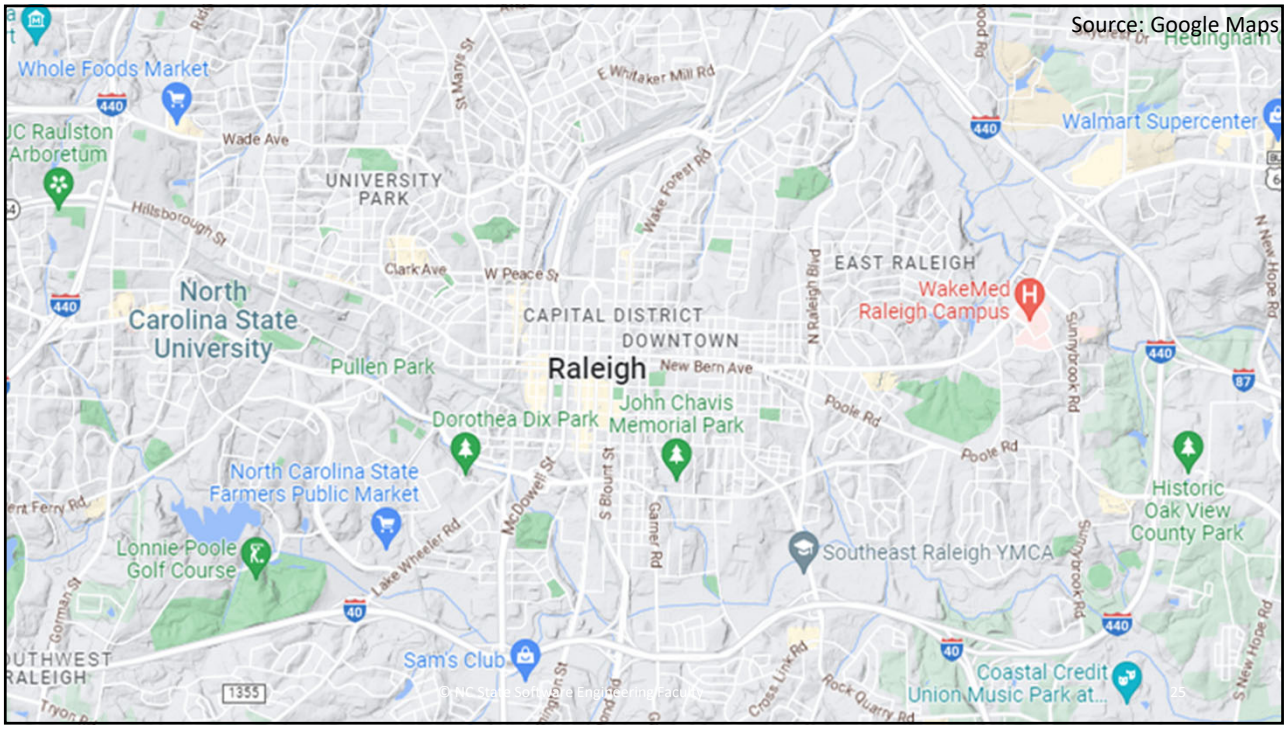
- A single system can have multiple architectures
  - There can be multiple views of each architecture
- Keep views simple: use different views for different purposes
  - Views are useful for reason about the properties of a system
  - Example: reason about the modules support a scenario
- Example: reason about the deployment of software across virtual servers

A non-tech example of different views of the same thing

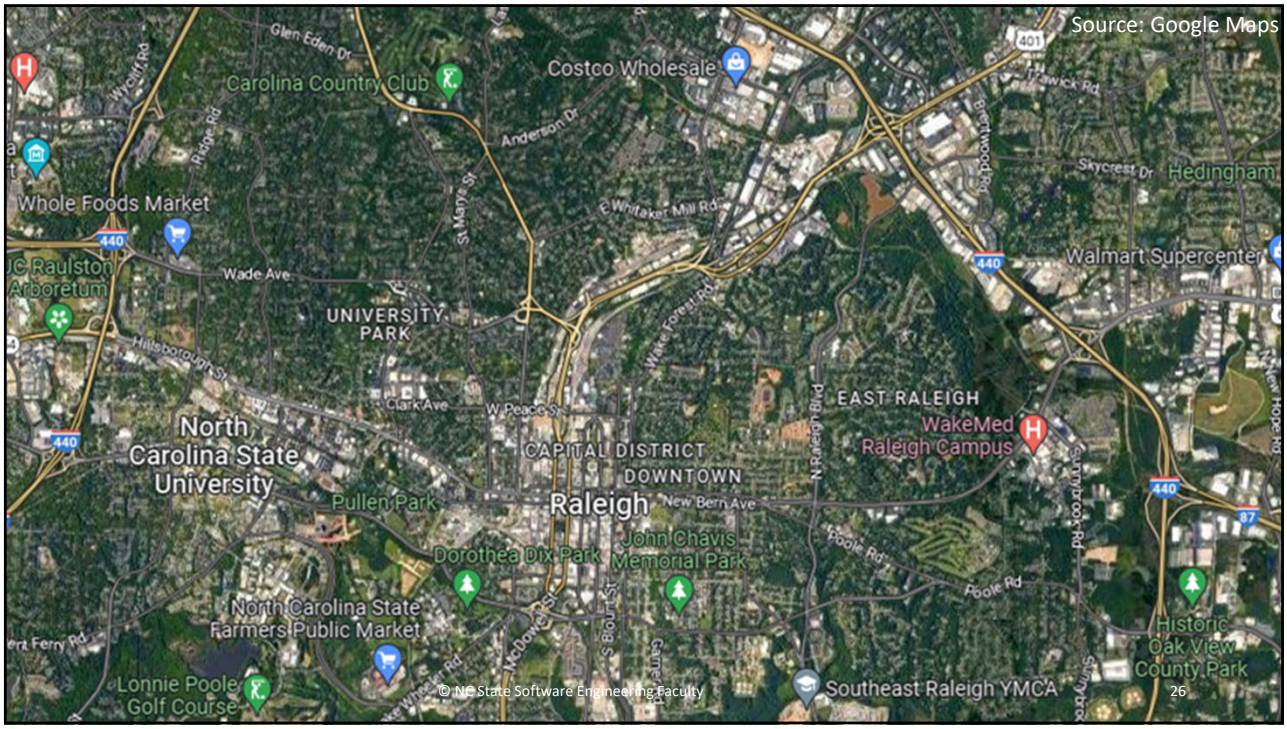
23



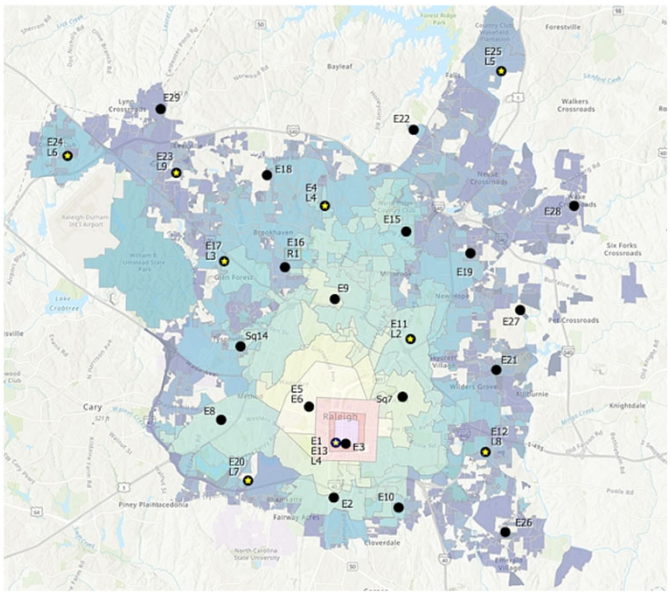
24


Source: Google Maps
Whole Foods Market
JC Raulston Arboretum
Wade Ave
UNIVERSITY PARK
Hillsborough St
Clark Ave
W Peace St
North Carolina State University
Pullen Park
CAPITAL DISTRICT
DOWNTOWN
Raleigh
New Bern Ave
Dorothea Dix Park
John Chavis Memorial Park
North Carolina State Farmers Public Market
Lonnie Poole Golf Course
Lake Wheeler Rd
McDowell St
S Blount St
Gamer Rd
Poole Rd
Sam's Club
Southeast Raleigh YMCA
EAST RALEIGH
WakeMed Raleigh Campus
Walmart Supercenter
Historic Oak View County Park
Coastal Credit Union Music Park at...
Sunnybrook Rd
N Raleigh Blvd
E Whitaker Mill Rd
Wake Forest Rd
St Marys St
Rock Quarry Rd
© NC State Software Engineering Faculty
25

25

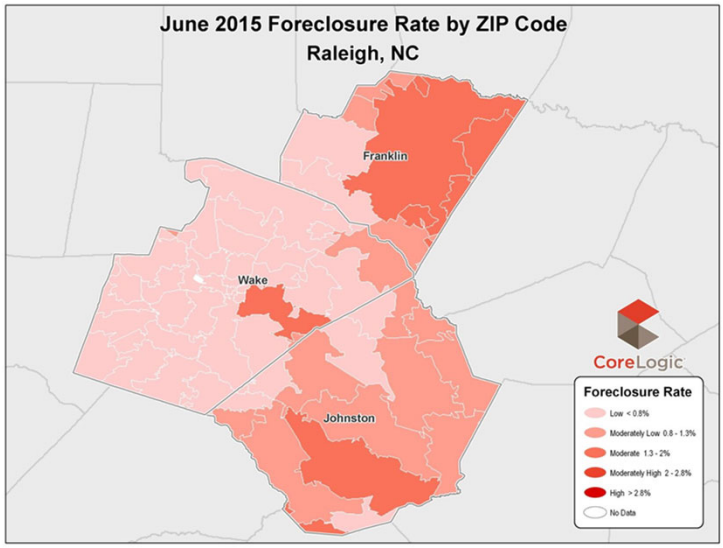

Source: Google Maps
Glen Eden Dr
Carolina Country Club
Costco Wholesale
Whole Foods Market
JC Raulston Arboretum
Wade Ave
UNIVERSITY PARK
Hillsborough St
Clark Ave
W Peace St
North Carolina State University
Pullen Park
CAPITAL DISTRICT
DOWNTOWN
Raleigh
New Bern Ave
Dorothea Dix Park
John Chavis Memorial Park
North Carolina State Farmers Public Market
Lonnie Poole Golf Course
Lake Wheeler Rd
McDowell St
S Blount St
Gamer Rd
Poole Rd
Southeast Raleigh YMCA
EAST RALEIGH
WakeMed Raleigh Campus
Walmart Supercenter
Hedingham
Skycrest Dr
Historic Oak View County Park
Sunnybrook Rd
N Raleigh Blvd
E Whitaker Mill Rd
Wake Forest Rd
St Marys St
Anderson Dr
Ridge Rd
Brentwood Rd
Trawick Rd
© NC State Software Engineering Faculty
26

26



© NC State Software Engineering Faculty

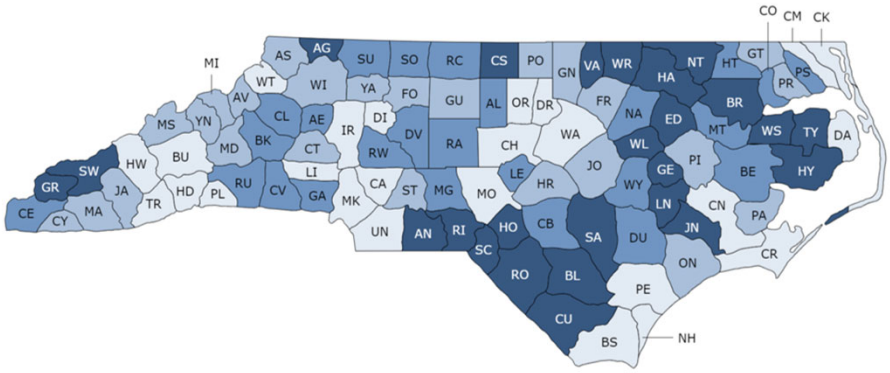
Source: https://legeros.com/ralwake/raleigh/history/timelines/2020-present.shtml



Source: https://trubluerealty.com/status-of-foreclosure-rates-in-nc/

© NC State Software Engineering Faculty

2022 Health Factors – North Carolina



Health Factor Ranks 1 to 25 26 to 50 51 to 75 76 to 100

Source: https://www.countyhealthrankings.org/explore-health-rankings/north-carolina/data-and-resources

© NC Sta

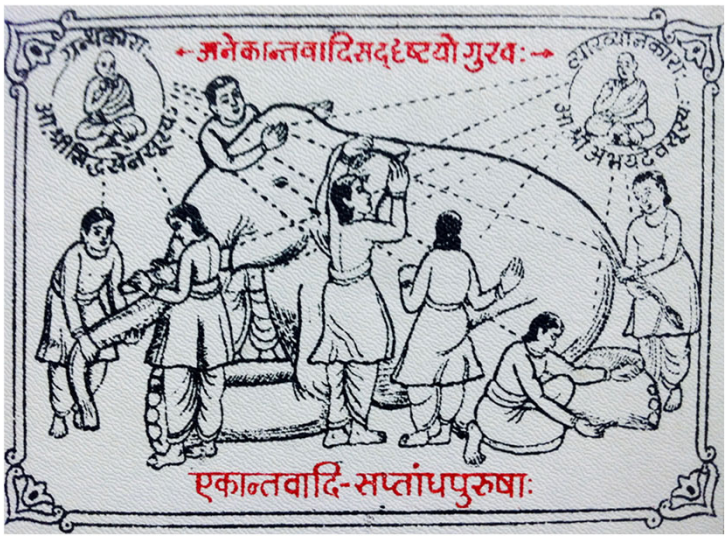
County Health Rankings & Roadmaps
Building a Culture of Health, County by County

29

29

Parable: One Elephant, Multiple Views

Blind Philosophers and an elephant



30

30

Parable: The Blind Philosophers and the Elephant

- The philosophers encounter an elephant for the first time
  - ”It’s like a thick snake,” says one, touching the moving trunk
  - “It’s like a fan,” argues another, touching the ear
  - ”It’s like a spear,” says the third, touching the tusk
- Each of them was right
  - And, none of them described the elephant as a whole

31

31

Back to tech

32

32

What are the “right” views?

33

33

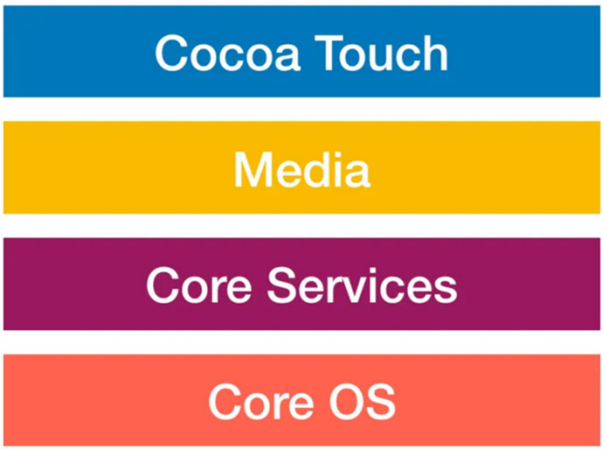
There is No “Right” View

- Match the view to the audience
  - What do they want to know about how the system works?
  - Include just enough detail to convey an intuitive understanding
  - Suppress details that do not relate to the purpose of the view
- Upcoming Examples:
  - The iOS overview is suitable for a broad audience
  - The Android Software Stack is suitable for developers

34

34

A View of iOS Suitable for a Broad Audience



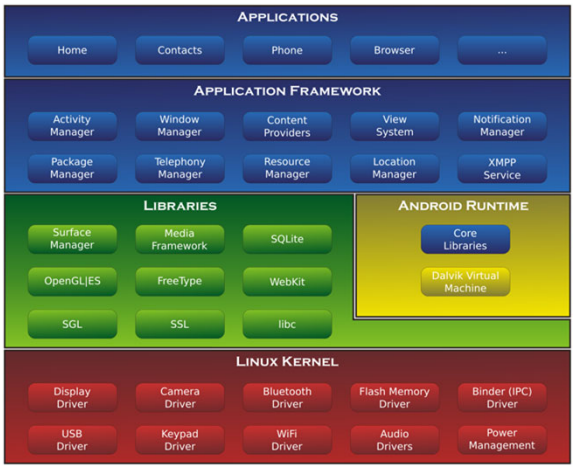
- Elements in the view
  - The four layers shown as boxes
- Relationships between elements:
  - An upper layer may use the layer just below

35

35

A View of Android Suitable for Developers

Google's diagram of the Android Software Stack



By Smieh - Anatomy Physiology of an Android, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20067152

36

36

Lessons from Classical Architecture for Buildings

- Architects have long used views to highlight properties of buildings
- The Roman architect Vitruvius (1st Century BCE) singled out 3 properties:
  - **Utility**: Does the building conveniently serve its intended purpose?
  - **Strength**: Will it stand? Foundations solid? Materials wisely chosen?
  - **Beauty**: Is the appearance pleasing and in good taste?



In your opinion, what are software counterparts of Vitruvius's 3 "virtues" (desirable properties) for buildings?

37

37

Desirable Properties of an Architecture

Counterparts of utility, strength, beauty

- Functionality
  - Meet user requirements? Be fit for purpose? Worth the cost?
- Quality Attributes
  - Support quality attributes like performance and scalability?
- Elegance
  - Understandable? Modifiable? Have conceptual integrity?
  - Will it lead to a clean implementation?

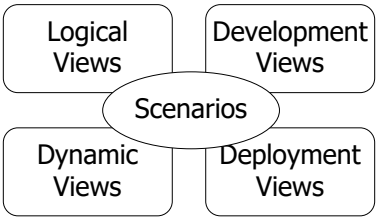
38

38

4+1 Grouping of Views

Views to consider for an an architecture description

- Logical Views focus on support for user functionality and requirements
- Development Views focus on modules in the source code
- Dynamic Views focus on run-time object/process, data flows
- Deployment Views focus on configuration, mapping of software onto servers, geographic distribution



Kruchten [1995]

39

39

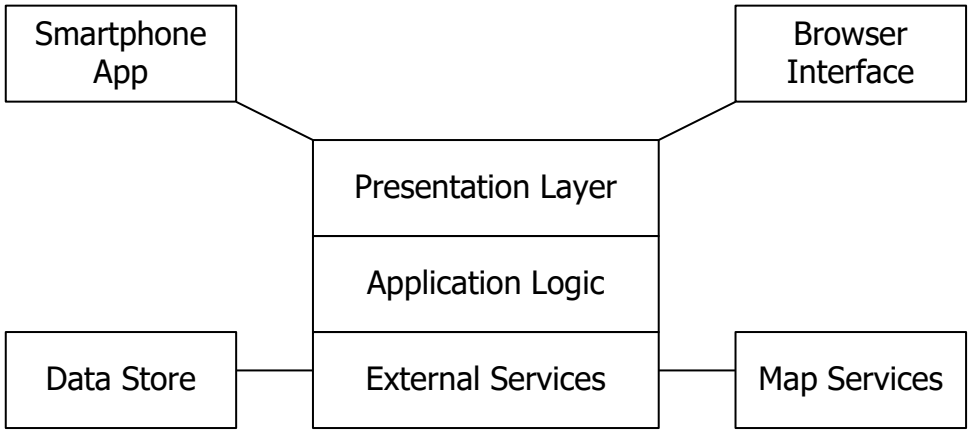
Examples of Views

40

40

A Logical View of a System

The system has **two user interfaces**: for smartphone and browser

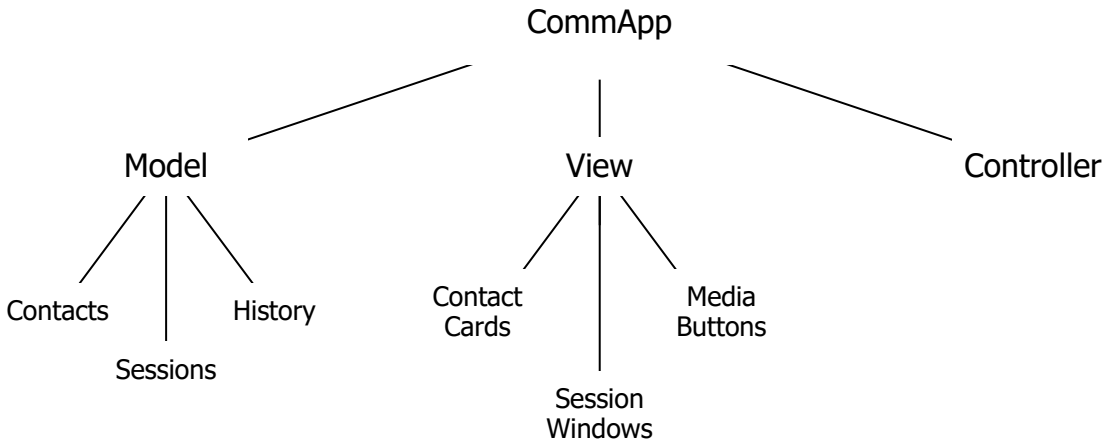


41

41

A Development View: A Module Hierarchy

This view shows module-submodule relationships

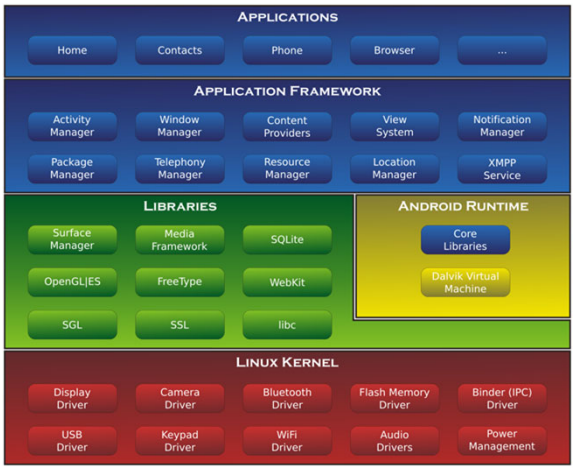


42

42

A Development View of the Android Stack

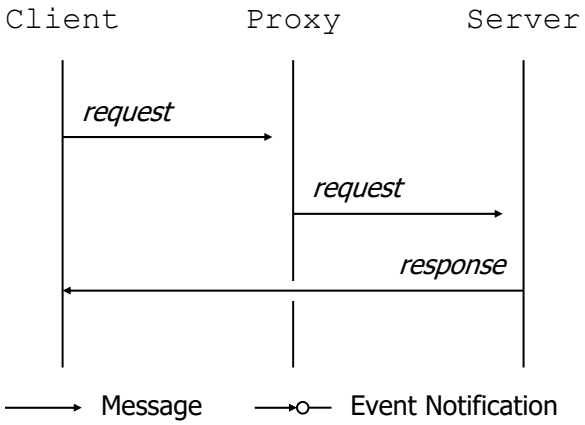
Google's diagram of the Android Software Stack



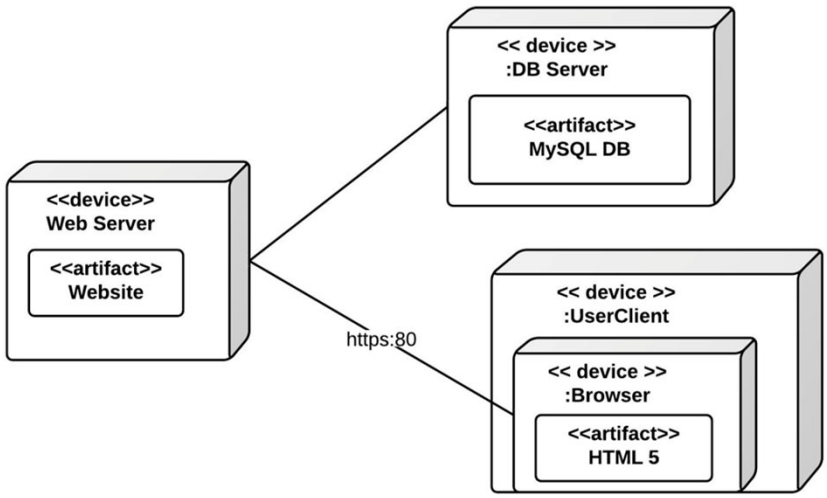
By Smieh - Anatomy Physiology of an Android, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20067152

A Dynamic View of Interactions

This view focuses on interactions rather than on the components



A Deployment View



45

45

Deployment View





46

46

Slide 46

1 added for fun because this is a cool factoid about the world
Kathryn Stolee, 2/7/2024

The "4+1" View Model of Software Architecture

	Logical	Process (a.k.a. Dynamic)	Development	Physical (a.k.a. Deployment)	Scenario
Description	Shows the component (Object) of system as well as their interaction	Shows the processes / Workflow rules of system and how those processes communicate, focuses on dynamic view of system	Gives building block views of system and describe static organization of the system modules	Shows the installation, configuration and deployment of software application	Shows the design is complete by performing validation and illustration
Viewer / Stake holder	End-User, Analysts and Designer	Integrators & developers	Programmer and software project managers	System engineer, operators, system administrators and system installers	All the views of their views and evaluators
Consider	Functional requirements	Non Functional Requirements	Software Module organization (Software management reuse, constraint of tools)	Nonfunctional requirement regarding to underlying hardware	System Consistency and validity
UML – Diagram	Class, State, Object, sequence, Communication Diagram	Activity Diagram	Component, Package diagram	Deployment diagram	Use case diagram

img

47

47

Architectural Patterns

48

48

Architectural Patterns

- Layered Systems
- Shared Data
- Observer
- Publish-Subscribe
- Model-View-Controller
- Dataflow
- Services
- Client-Server

49

Architectural Patterns

Pattern:	Good For:
• Layered Systems	• Portability and reuse
• Shared Data	• Persistent data
• Observer	• Event notifications
• Publish-Subscribe	• Broadcast events
• Model-View-Controller	• User interfaces
• Dataflow	• Operations on data streams
• Services	• Interoperability
• Client-Server	• Scalability of services

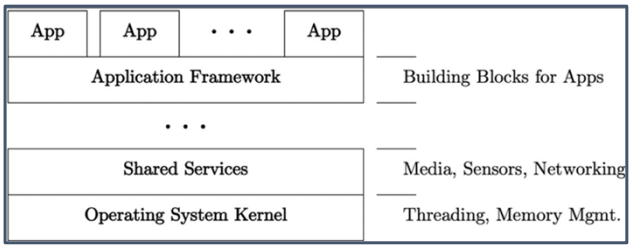
50

Textbook: Chapter 7, Table 7.1

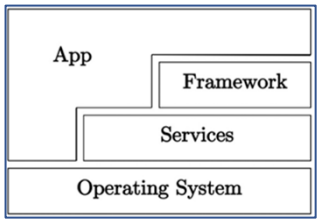
50

Layered Systems

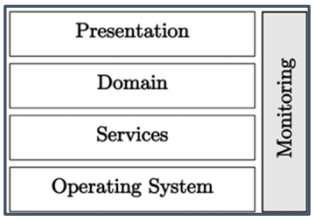
- **Problem**: design and build reusable and portable software
- **Solution**: break into related modules, called *layers*
  - Each layer has a responsibility
  - Each layer can interact with the layer below it
- **Layer Bridging**: a layer may access multiple layers below it
- **Layers with Sidecar**: cross-cutting layer used for monitoring, auditing, security, etc.
  - Ensure lower layers cannot access upper layers through the sidecar



Layered System



Layer Bridging

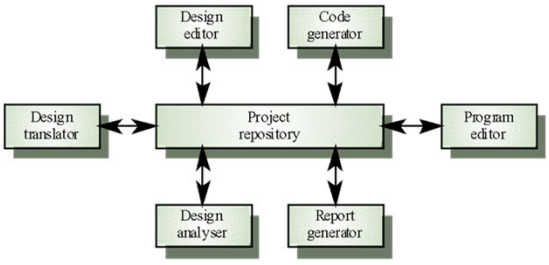


Layers with Sidecar

Textbook: Chapter 7, Figures 7.3 and 7.4 (a & b)

51

Shared Data / Repository



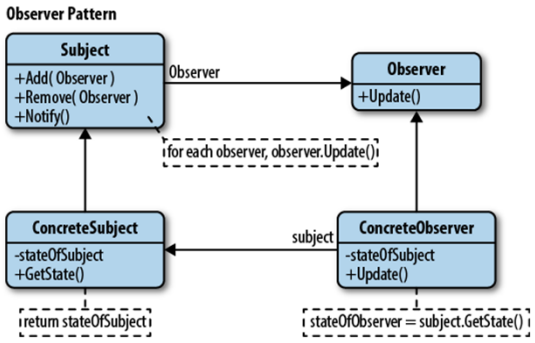
- **Problem**: let multiple components work with persistent data
- **Solution**:
  - A shared central database or repository can be accessed by all accessors
  - Accessors interact with each other through shared data store
- **Considerations:**
  - Reliability - backups
  - Performance - caching
  - Scalability - consistent copies

Pressman

52

Observer

- **Problem**: how to notify observers about events from publisher (subject)
- **Solution**: observer knows the publisher; publisher doesn't know about observers
  - Publisher has a set of observers
  - Observer registers with publisher
  - Publisher's set delivers events to all registered observers



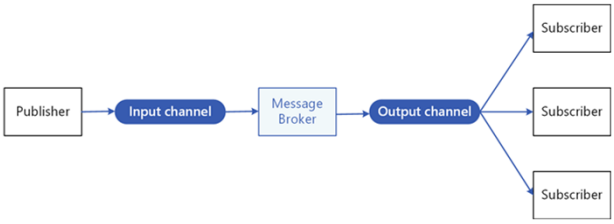
img

53

53

Publish-Subscribe

- **Problem**: notify subscribers of events when subscribes/publishers are unaware of each other
- **Solution**: level of indirection (e.g., message broker)
  - register publishers
  - register subscribers for specific types of events
  - publisher raises event
  - message broker notifies subscribers



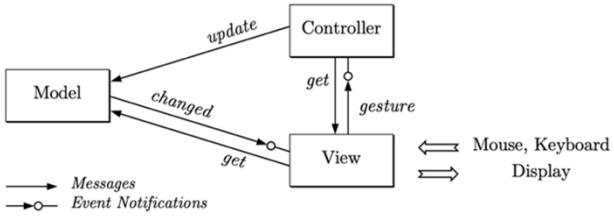
img

54

54

Model-View-Controller

- **Problem:** present common information in multiple views
  - observer pattern used to synchronize the model
- **Solution:** separate presentation from model
  - *Model*: domain logic and objects
  - *View*: displays information and gathers user interactions
  - *Controller*: interprets user activity and informs the model
  - View or controller *observes* the model

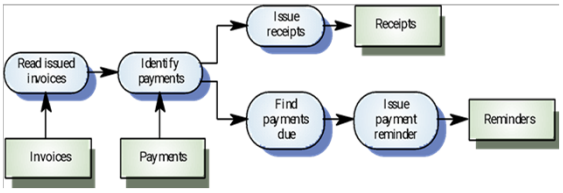


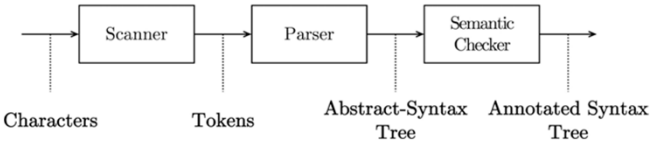
Textbook: Chapter 7, Figures 7.8

55

Dataflow

- **Problem:** transform data streams
- **Solution:** create independent filters that transform data
  - the output from one filter becomes the input to another filter or the end of the dataflow
  - if every filter has at most one input and one output, it's a *pipeline*
- **Considerations:**
  - Unbound data streams
    - Mapping and Reducing Transformations
    - Windowing
  - Big data
    - Parallelism
    - Reliability
    - Bandwidth

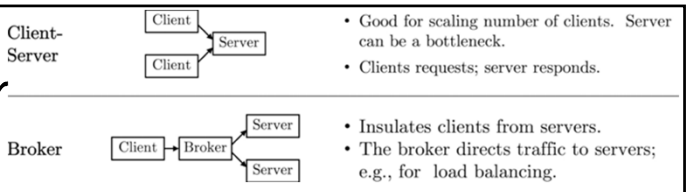




Textbook: Chapter 7, Figure 7.9 & Pressman

56

Client-server & Broker



Client Server

- **Problem:** connect multiple clients with a specific server
- **Solution:** many-to-one
  - Clients know server identity
  - Clients send request w/ info about client
  - Once the reply is sent, the connection is closed
- **Considerations:**
  - sync or async responses
  - name service
  - load balancing
  - assumption of shared data

Broker

- **Problem:** match client request to the appropriate service(s) (server(s))
- **Solution:** broker intermediary
  - name service and load balancing are logical functions of the broker
- **Considerations:**
  - performance bottlenecks
  - latency
  - single point of failure
  - security vulnerability
  - assumption of shared data

57

Exercise

Exercise 12.02

What architectural style would you use, and why, for the following:

- Software controlling an elevator
- Real time stock exchange
- Software automating a library's book and patron information and tasks such as check-in/check-out
- A company's payroll software

- Layered
- Shared Data
- Observer
- Publish-Subscribe
- Model-View-Controller
- Dataflow
- Client-Server

58

Design vs. Architecture

Design Questions

- How do I add a menu item in Eclipse?
- How can I make it easy to add menu items in Eclipse?
- What lock protects this data?
- How does Google rank pages?
- What encoder should I use for secure communication?
- What is the interface between objects?

Architectural Questions

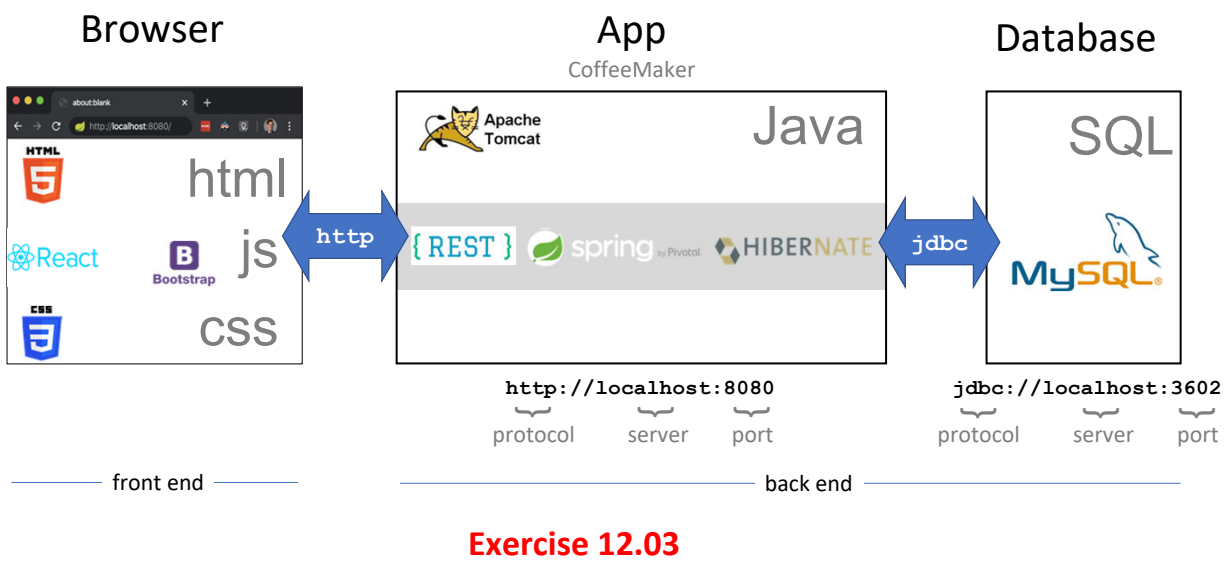
- How do I extend Eclipse with a plugin?
- What threads exist and how do they coordinate?
- How does Google scale to billions of hits per day?
- Where should I put my firewalls?
- What is the interface between subsystems?

59

Slide adapted from Rohan Padhye and Michael Hilton

59

High Level 326 App Architecture



Exercise 12.03

60

Document Architecture



img

61

Why Document Architecture?

- Blueprint for the system
  - Artifact for early analysis
  - Primary carrier of quality attributes
  - Key to post-deployment maintenance and enhancement
- Documentation speaks for the architect, today and 20 years from today
  - As long as the system is built, maintained, and evolved according to its documented architecture
- Support traceability.

Slide adapted from Rohan Padhye and Michael Hilton

62

Guidelines: Avoiding Ambiguity

- **Always include a legend**
- Define precisely what the boxes mean
- Define precisely what the lines mean
- Don't mix viewtypes unintentionally
- Supplement graphics with explanation
  - Very important: rationale (architectural intent)
- Do not try to do too much in one diagram
  - Each view of architecture should fit on a page
  - Use hierarchy

Slide adapted from Rohan Padhye and Michael Hilton