Architectural Design

CSCI 5040: Professional Master's Project (1 of 2)

Lecture 10

Learning Objectives

- Review best practices for architectural design
- Review design team activities/class schedule & deliverables

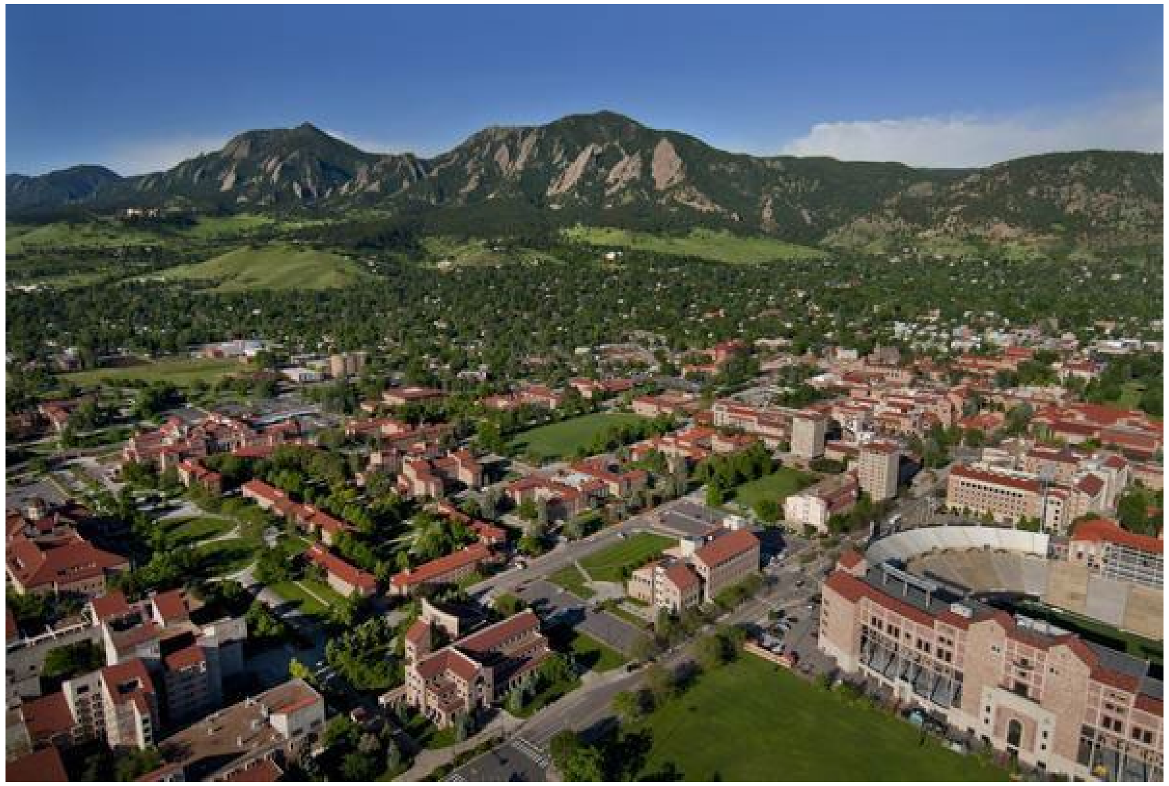
- Full disclosure
  - Taken in part from projects class architecture presentation by Trevor DiMartino, 2018
  - Also stealing a bit from my own OOAD and Embedded Interface Design lectures

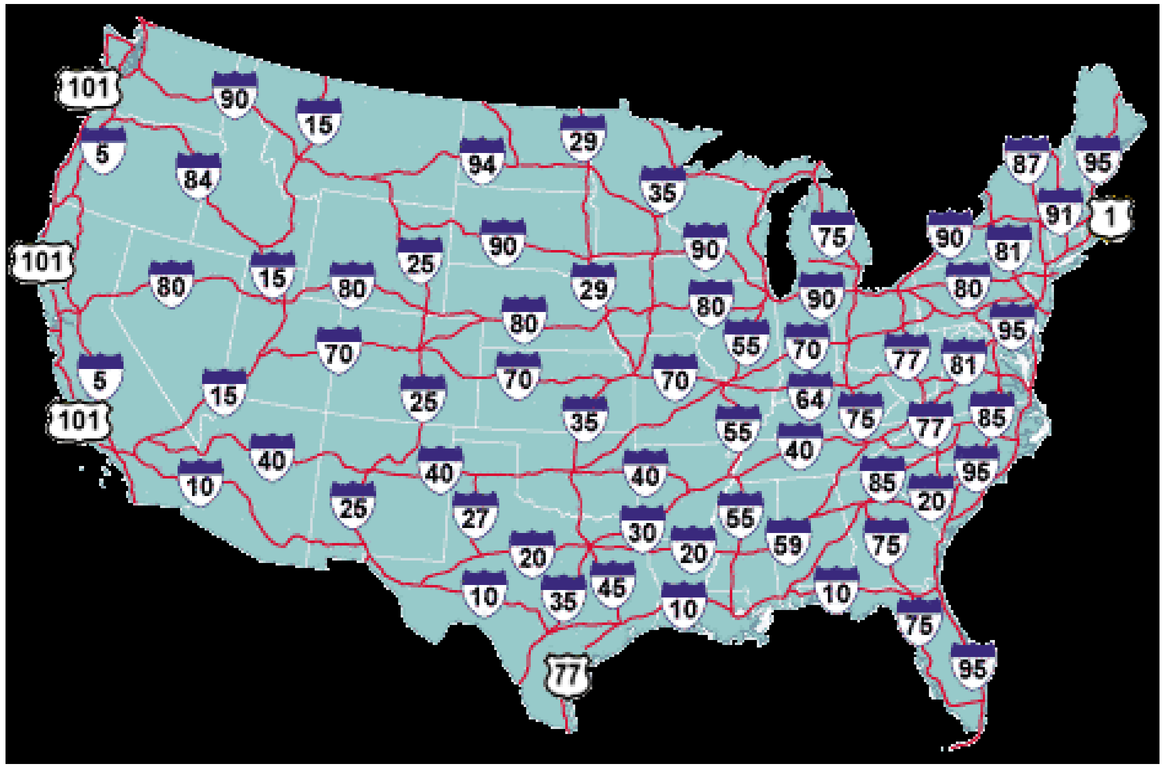
Architecture Topics

- Definitions
- Goals
- Best approaches
- Methods

Consider…

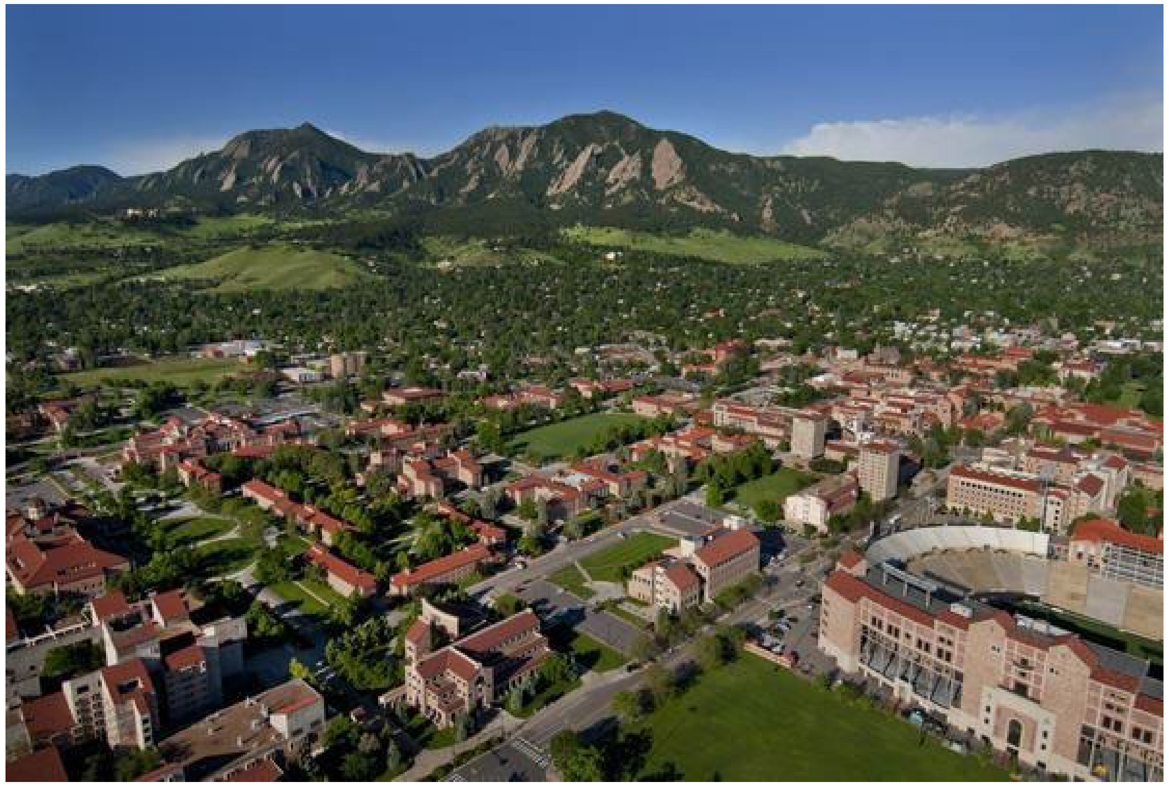
- The CU Campus
- The US Interstate System

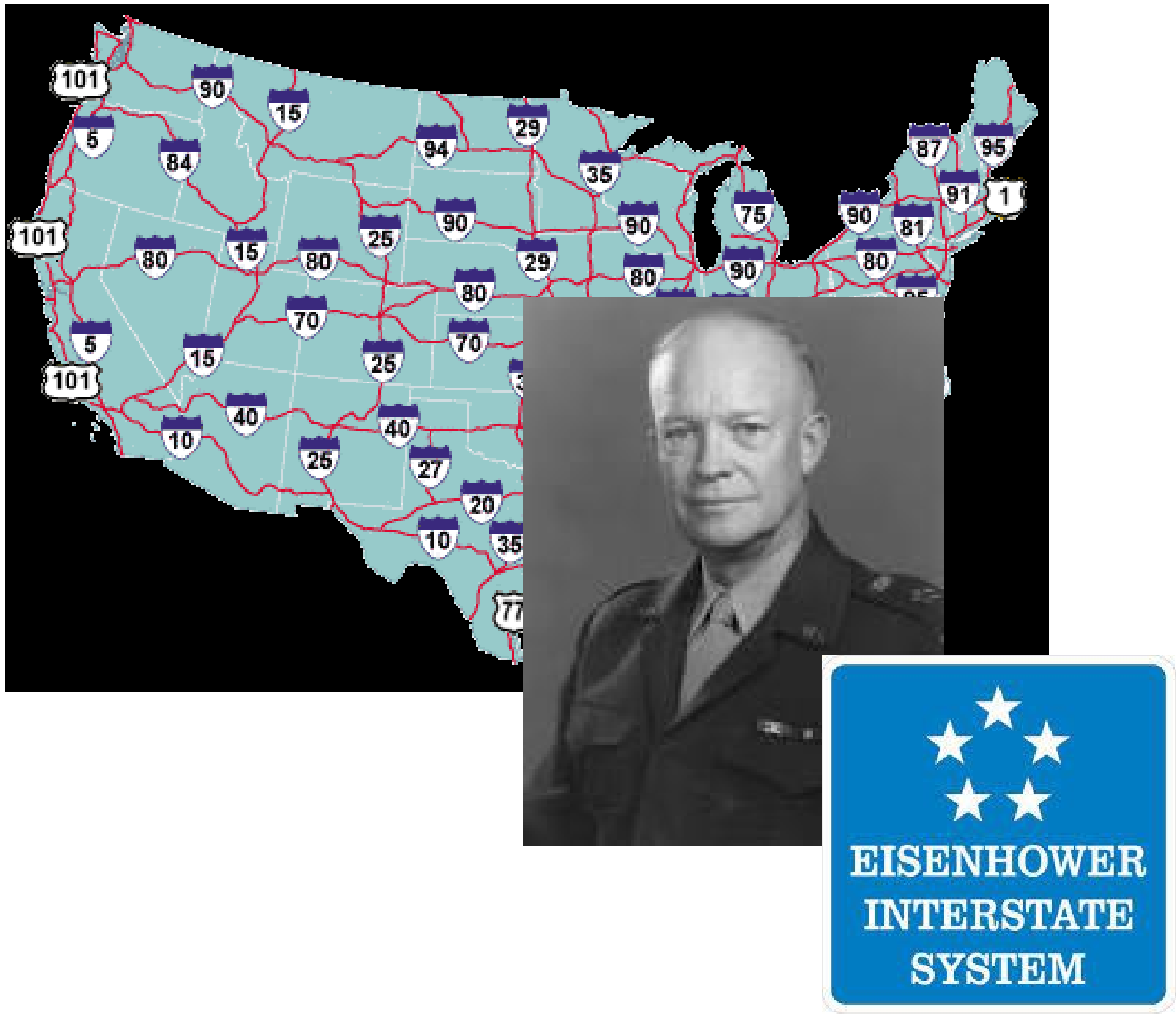




Consider…

- The CU Campus
  - 1919 Campus Development Plan
- The US Interstate System
  - Eisenhower - Federal Aid Highway Act of 1956
  - 2018 25% of all vehicle traffic on Interstate





System-Level Architecture

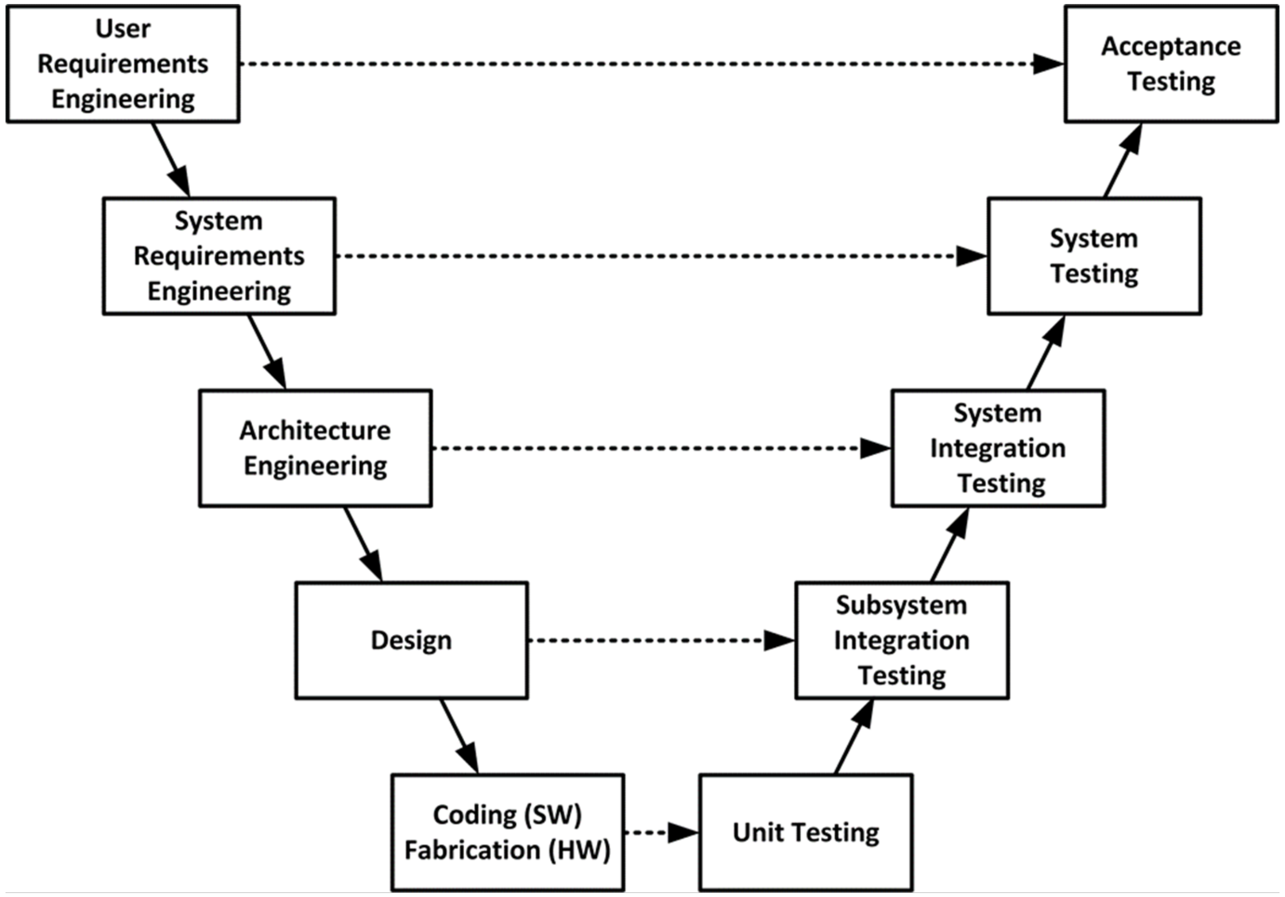
- Set of global project decisions/constraints
- Structures and patterns to be realized with modules
- Structure and flow of components in use
- Views of the system from multiple perspectives

- From Martin Fowler:
  - The shared understanding that the expert developers have of the system design
  - The decisions you wish you could get right early in a project
  - Architecture is about the important stuff; whatever that is
  - https://martinfowler.com/architecture/

- Your approach to architecture may vary by the complexity of the project

Architecture in Software

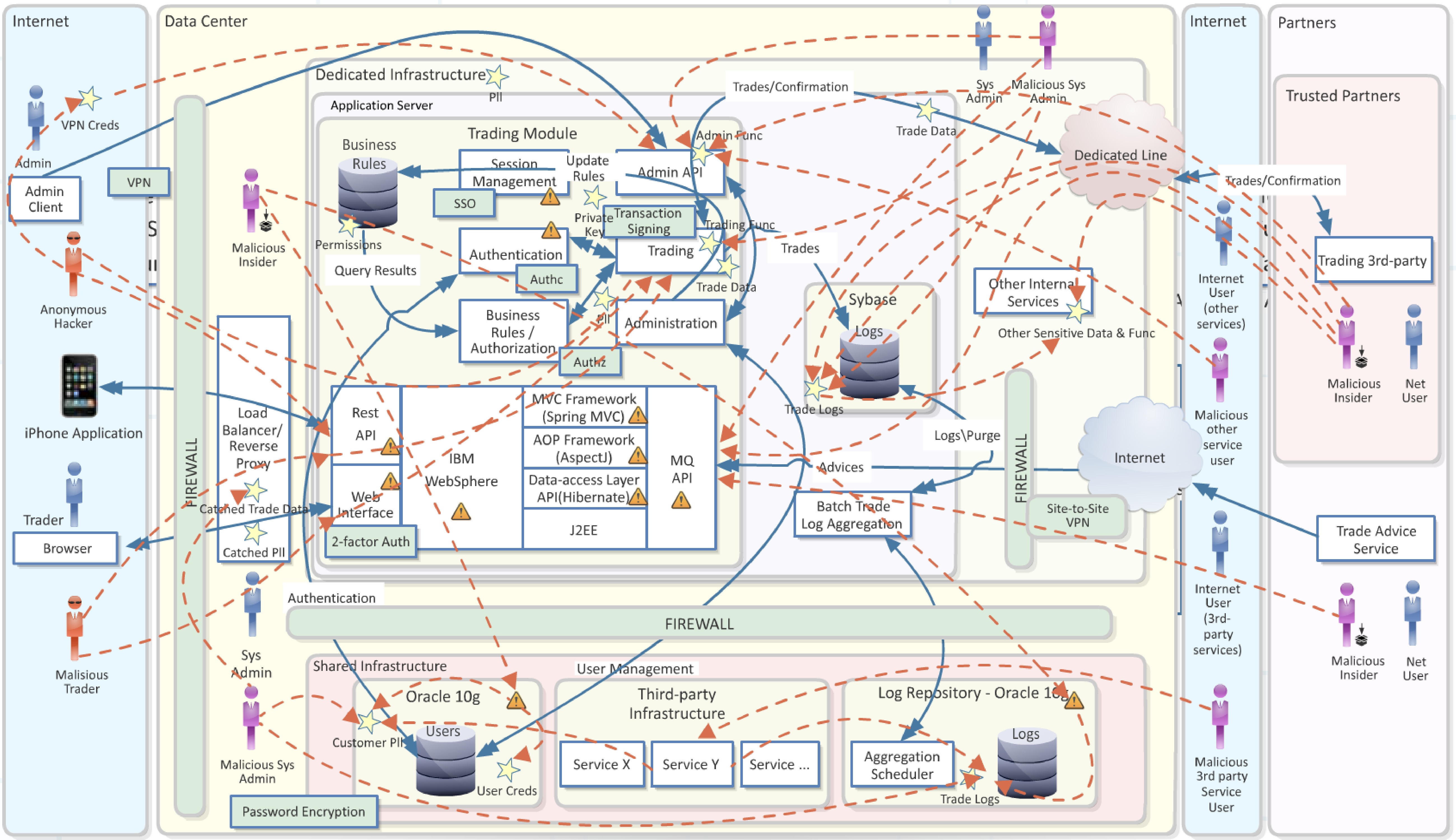
- Architecture: A phase in the software lifecycle, between user/system requirements and detailed design work
- System Architecture: The form and structure of the system
  - Verified with system integration and overall system tests
- Architectural Patterns: Repeatable solution to common software system problems


User Requirements Engineering
Acceptance Testing
System Requirements Engineering
System Testing
Architecture Engineering
System Integration Testing
Design
Subsystem Integration Testing
Coding (SW) Fabrication (HW)
Unit Testing

https://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html

Planning an Architecture

- In building architecture, blueprints are drafted showing the system to be built from different perspectives
  - Floor plan
  - Electrical
  - HVAC
  - Plumbing
- In software, these might be:
  - Code view
  - Run time behavior view
  - Data flow view
  - Etc.
- Why multiple views?

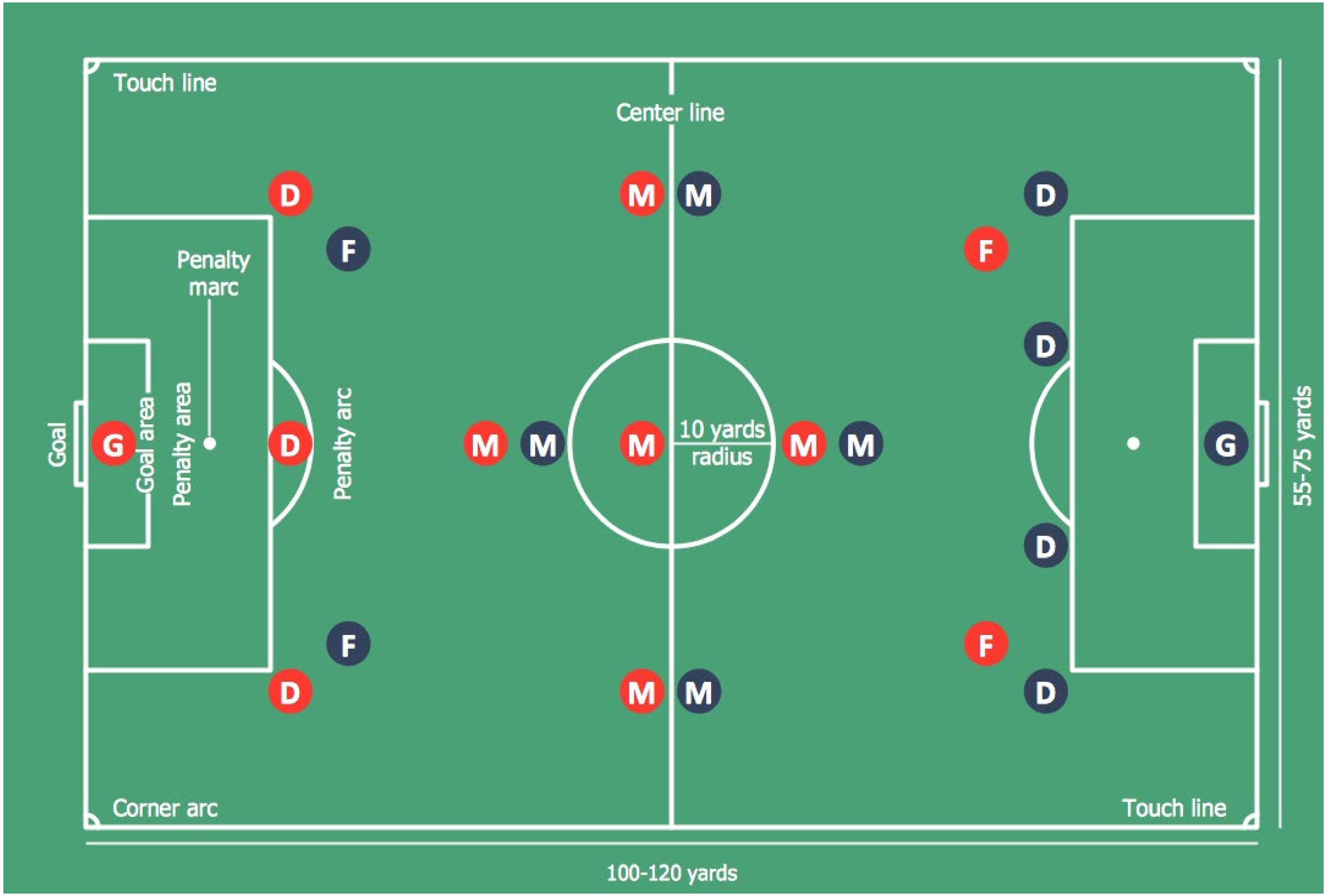


Too complicated for a single view…

Keys: Simplicity and Consistency

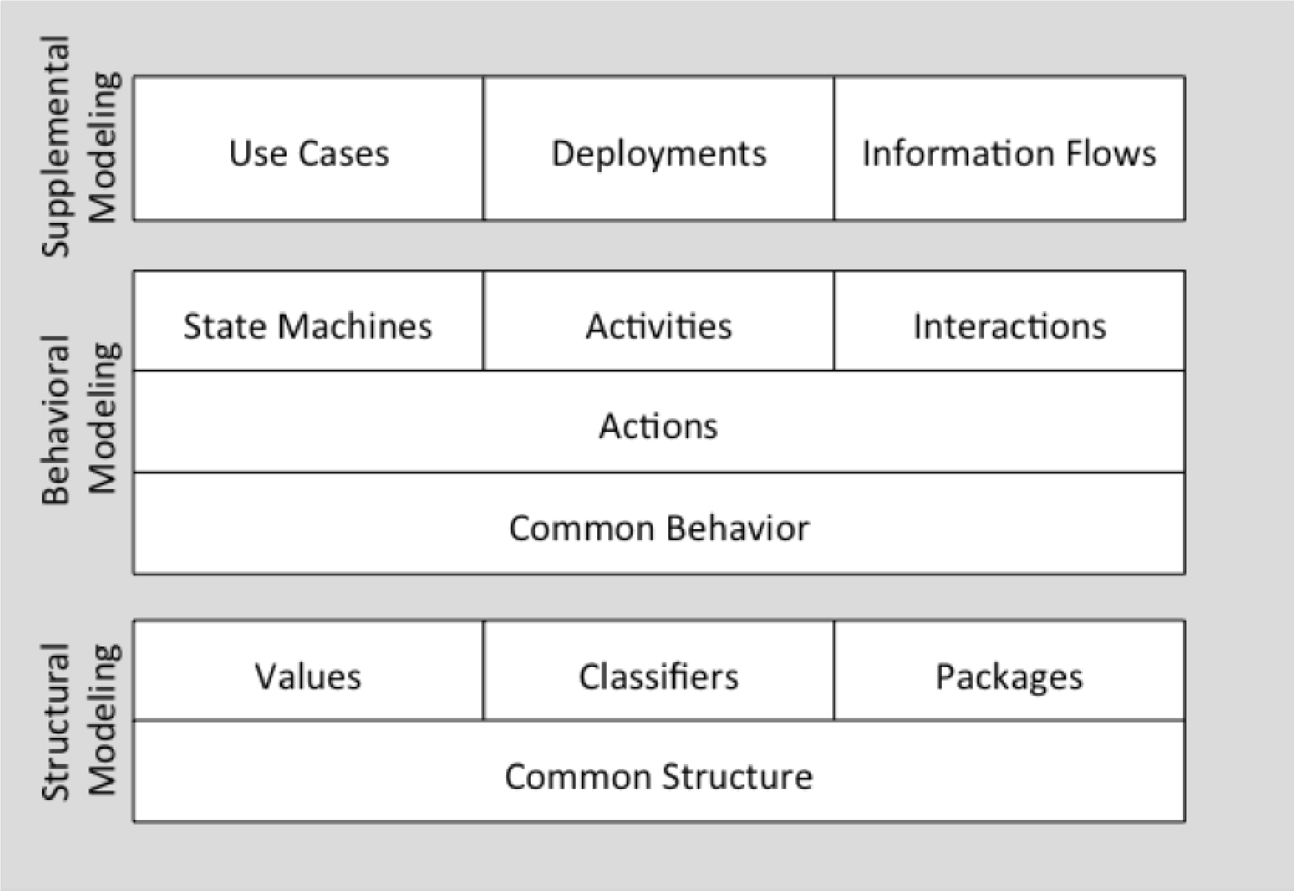
- Prior examples – clear system goals
  - CU = Campus infrastructure that is visually consistent and supports University operations
  - Interstate = Transportation system to support interstate transport for defense and commerce
- Simplicity
  - Use multiple viewpoints to specify architecture
  - Separation of concerns for local consideration and optimization
  - Complexity will mask underlying issues and mistakes
- Consistency
  - Enhances system understanding
  - Discoveries of commonality and behavior
  - Unnecessary diversity in system may lead to issues

Does this define a soccer game?



UML Diagrams Provide Multiple Views

- Diagrams from the current UML release (https://www.omg.org/spec/UML/2.5.1/PDF)
- Structural (Static)
  - Class
  - Object
  - Package
  - Model
  - Composite Structure
  - Internal Structure
  - Collaboration Use
  - Component
  - Manifestation
  - Network Architecture
  - Profile
- Supplemental (both structural and behavioral elements)
  - Use Case
  - Information Flow
  - Deployment



Supplemental Modeling

Use Cases | Deployments | Information Flows

Behavioral Modeling

State Machines | Activities | Interactions

Actions

Common Behavior

Structural Modeling

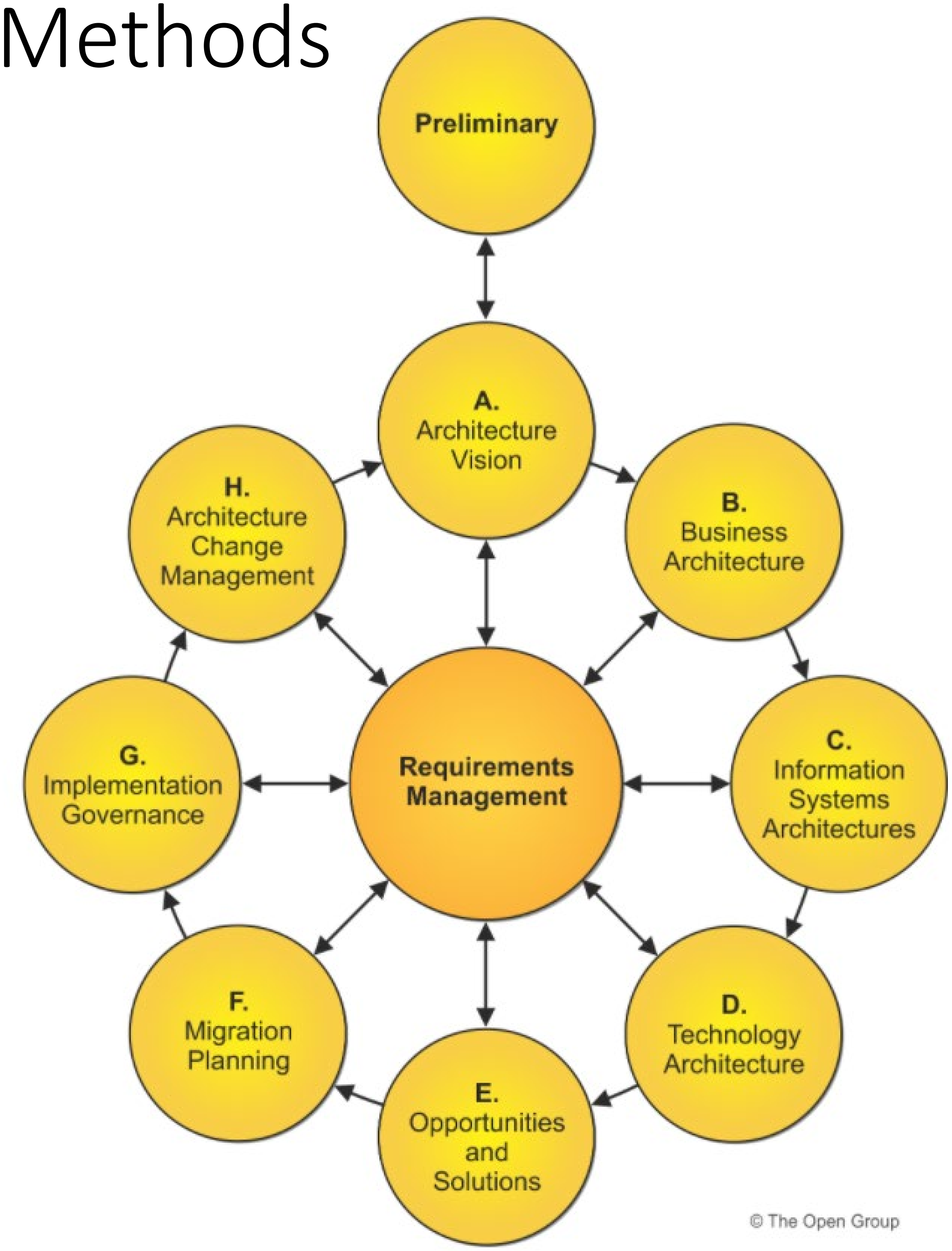
Values | Classifiers | Packages

Common Structure

- Behavior (Dynamic)
  - Activity
  - Sequence
  - State (Machine)
  - Behavioral State Machine
  - Protocol State Machine
  - Interaction
  - Communication (was Collaboration)
  - Timing
  - Interaction Overview

Architecture Modeling Methods

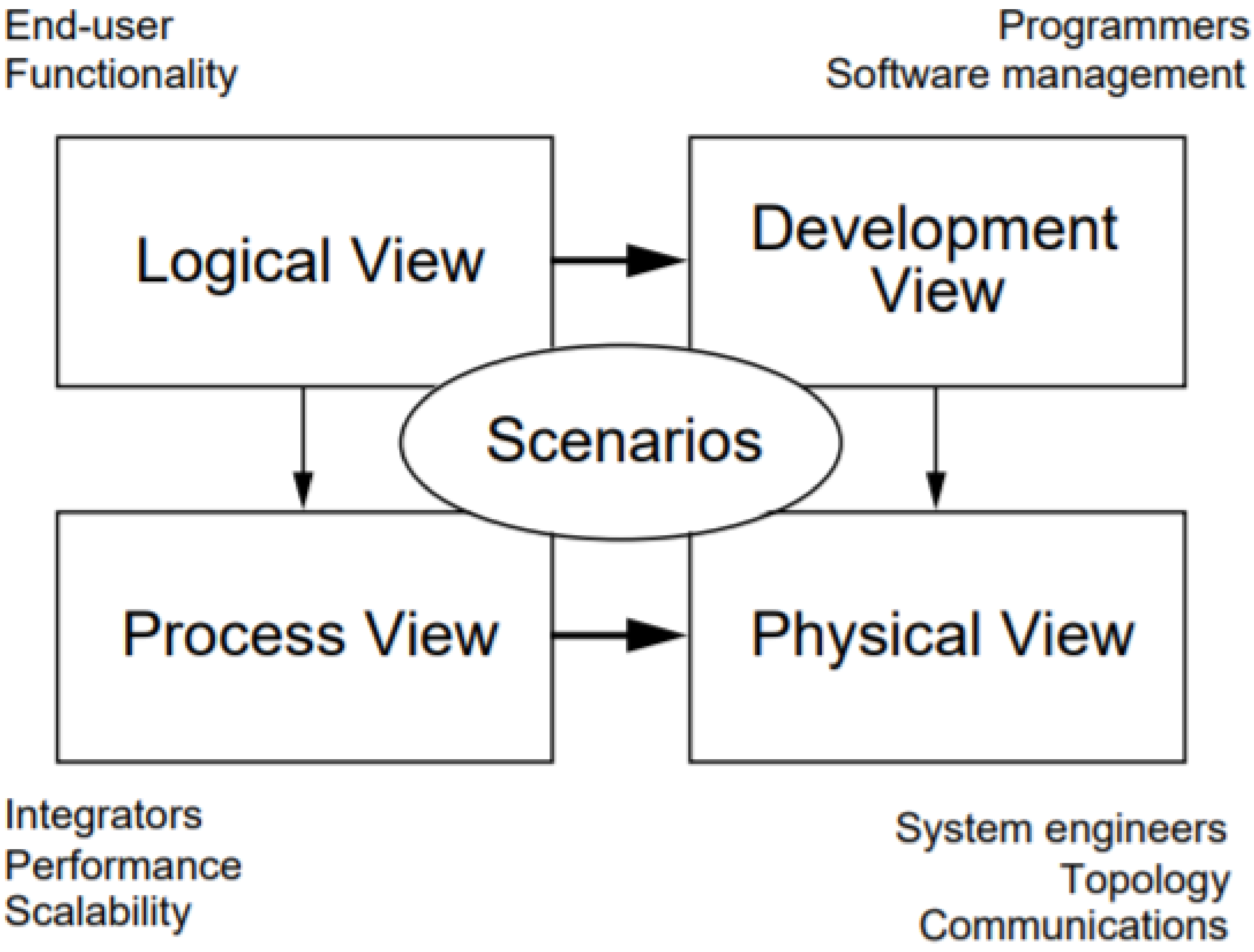
- Structuring our approach to defining our views of the system
- Common Methods
  - Iterative UML-based Design
  - 4+1
  - C4
  - Others
    - TOGAF →
      - The Open Group Architecture Framework – process for enterprise level architecture
      - https://www.opengroup.org/togaf
    - Arc42
      - Template-based with multiple views for architecture description
      - https://arc42.org/overview/


Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

© The Open Group

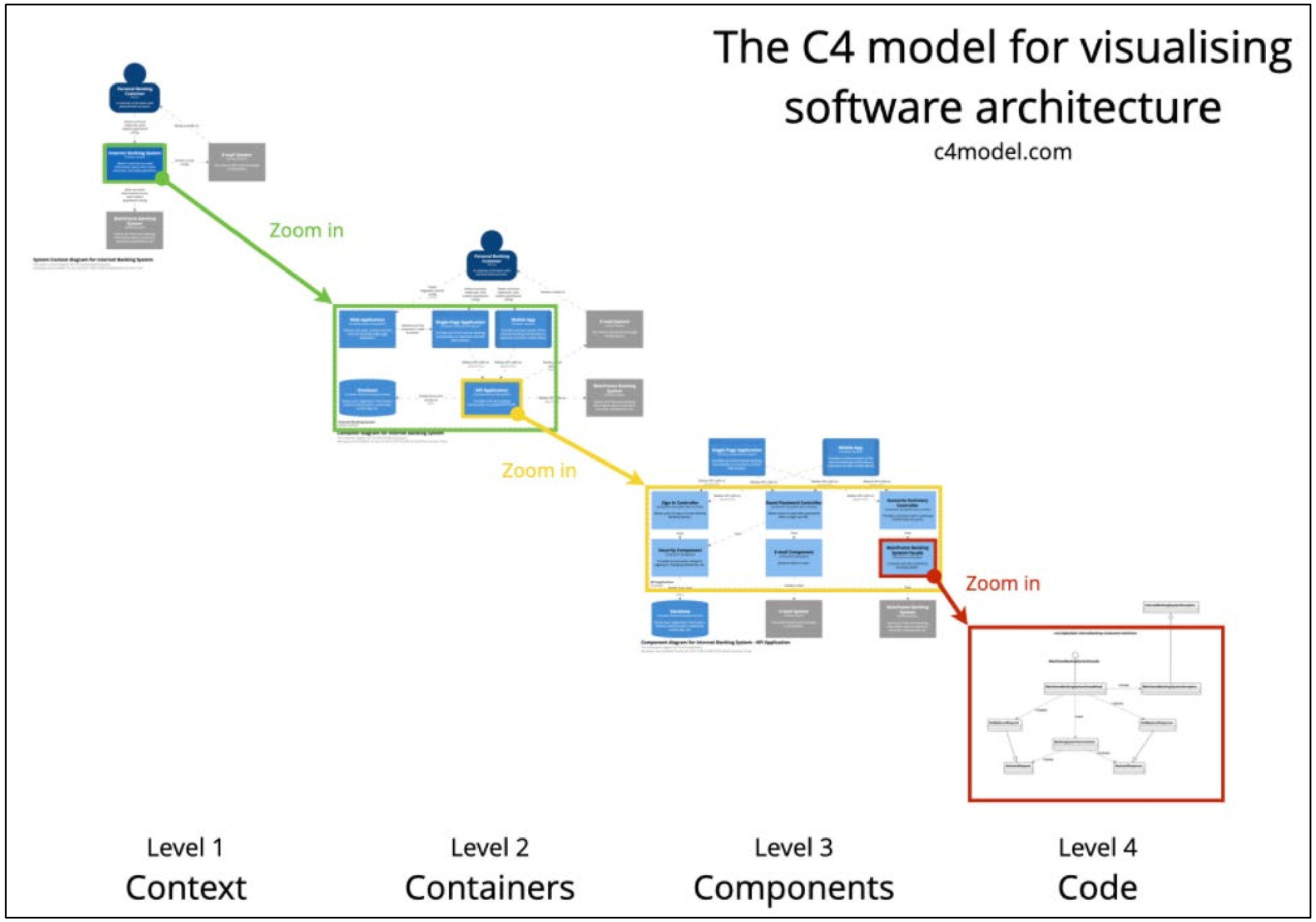
4+1 Model

- By Phillipe Kruchten, 1995, in development at Rational around the time of UML
- Use cases at the core, different perspectives on the system elements in the four views
- Still a useful way to breakdown a complex design into different views
- https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf



C4 Model

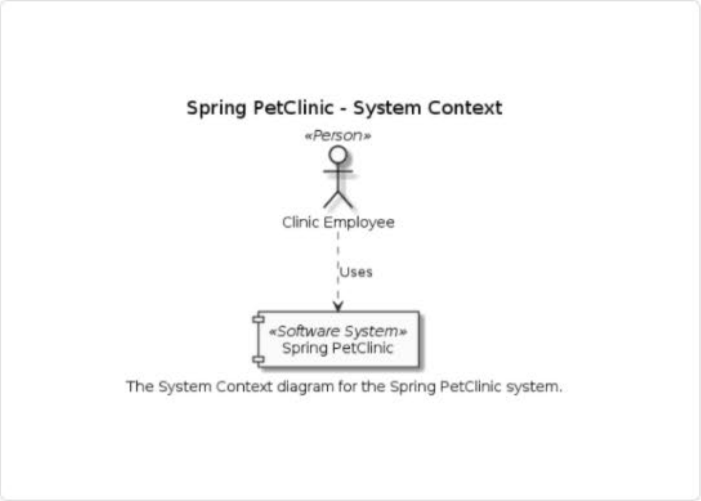
- A response to agile vs. architecture
- More modern view of architectural and progressive design –
  - “Maps of your code”
  - abstraction first
- 4 Levels
  1. System Context – how the system fits into its environment
  2. Containers – high level technical building blocks (not Docker, but different types of apps or data stores)
  3. Components – details of components in containers
  4. Code – UML Class Diagram
- https://c4model.com/

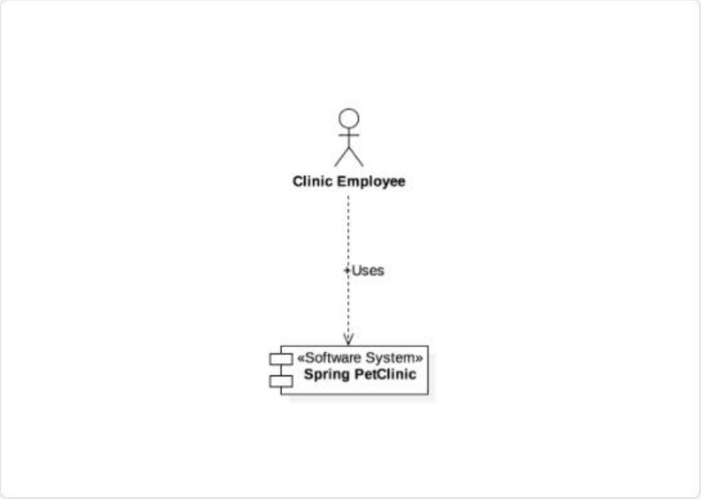


C4 & UML

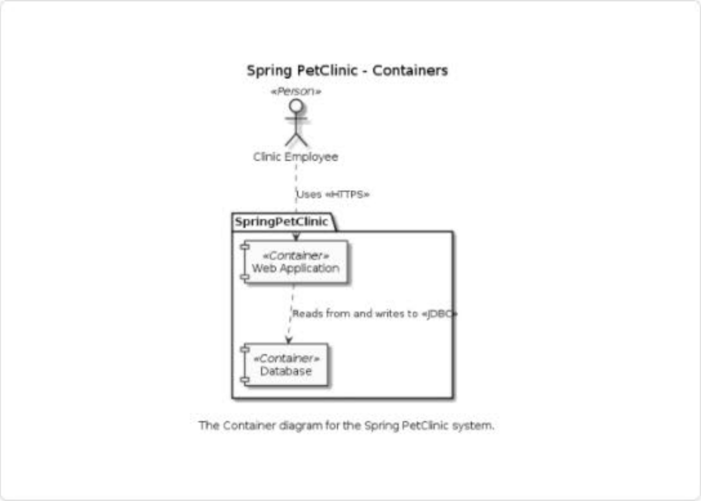
- Common to replace the boxes/arrow models in C4 with appropriate UML diagrams (https://c4model.com/):

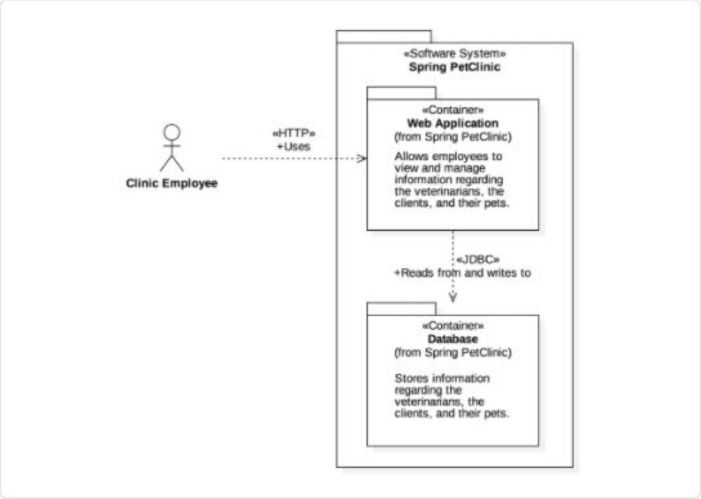
System Context diagram


Spring PetClinic - System Context
«Person»
Clinic Employee
Uses
«Software System»
Spring PetClinic
The System Context diagram for the Spring PetClinic system.

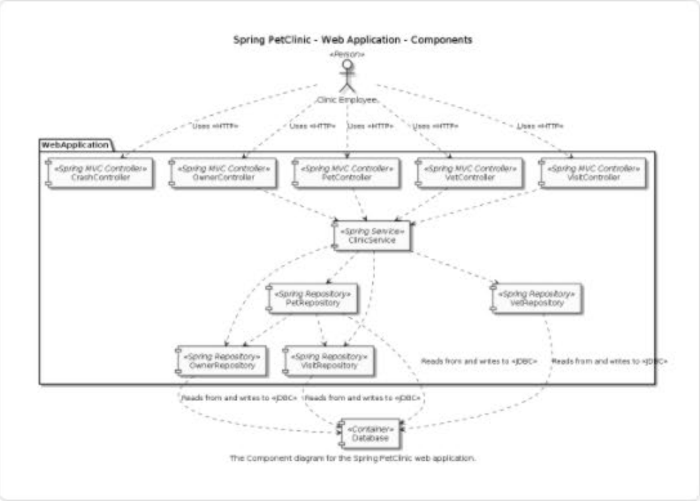


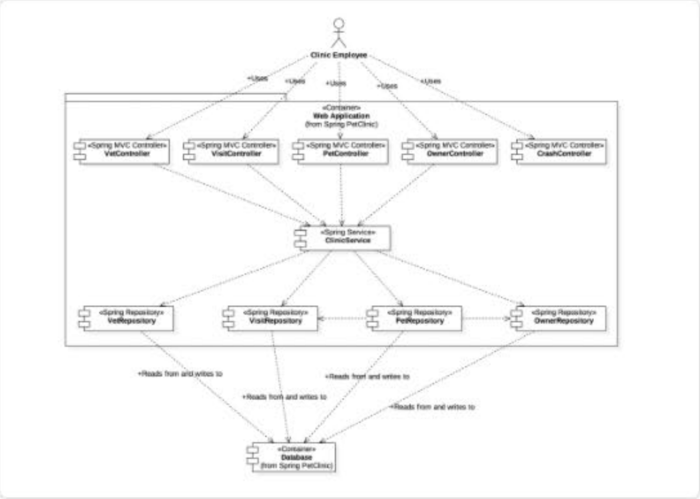
Container diagram


Spring PetClinic - Containers
«Person»
Clinic Employee
Uses «HTTPS»
SpringPetClinic
«Container»
Web Application
Reads from and writes to «JDBC»
«Container»
Database
The Container diagram for the Spring PetClinic system.


«Software System»
Spring PetClinic
«Container»
Web Application
(from Spring PetClinic)
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
Clinic Employee
«HTTP»
+Uses
«JDBC»
+Reads from and writes to
«Container»
Database
(from Spring PetClinic)
Stores information regarding the veterinarians, the clients, and their pets.

Component diagram


Spring PetClinic - Web Application - Components
«Person»
Clinic Employee
Uses «HTTP»
Uses «HTTP»
Uses «HTTP»
Uses «HTTP»
Uses «HTTP»
WebApplication
«Spring MVC Controller» CrashController
«Spring MVC Controller» OwnerController
«Spring MVC Controller» PetController
«Spring MVC Controller» VetController
«Spring MVC Controller» VisitController
«Spring Service» ClinicService
«Spring Repository» PetRepository
«Spring Repository» VetRepository
«Spring Repository» OwnerRepository
«Spring Repository» VisitRepository
Reads from and writes to «JDBC»
Reads from and writes to «JDBC»
Reads from and writes to «JDBC»
Reads from and writes to «JDBC»
«Container» Database
The Component diagram for the Spring PetClinic web application.


Clinic Employee
+Uses
+Uses
+Uses
+Uses
+Uses
«Container»
Web Application
(from Spring PetClinic)
«Spring MVC Controller» VetController
«Spring MVC Controller» VisitController
«Spring MVC Controller» PetController
«Spring MVC Controller» OwnerController
«Spring MVC Controller» CrashController
«Spring Service» ClinicService
«Spring Repository» VetRepository
«Spring Repository» VisitRepository
«Spring Repository» PetRepository
«Spring Repository» OwnerRepository
+Reads from and writes to
+Reads from and writes to
+Reads from and writes to
+Reads from and writes to
«Container»
Database
(from Spring PetClinic)

Other concerns

- Usability?
- Security?
- Reliability?
- Robustness?
- Scalability?
- Performance?
- Others?
- Often represented in requirements by non-functional entries
- Use different architectural views to consider these questions and your response in the design

Architecture in Agile

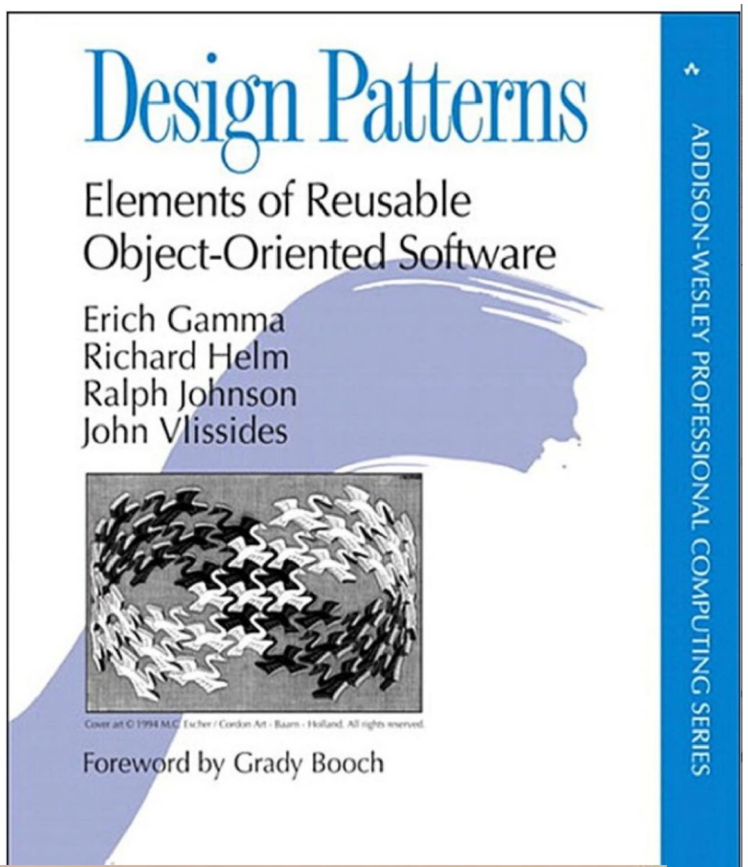
- Likely more iterative than all up-front
    - Doesn’t mean we don’t design
- Right-size the effort to the application
- Have an Architecture Owner
    - Focus on facilitating the evolution of the architecture over time
    - Facilitates creation of the architecture
    - Plans architectural “spikes” and refactors
    - Mentors team members in
        - Coding guidelines (automated standards)
        - Database and data model guidelines
        - Security guidelines
        - Documentation principles

Don’t over-engineer

- Many systems we develop are relatively small and don’t require a full architectural plan
- …but they can grow, which often requires re-architecting
- It’s good to think of the future
  - How will the use of your system differ in 10 years?
  - What if your system becomes quickly popular and user numbers multiply by 10?
  - What components of your system would need to change for a new client?
- …but it’s also important to start with something achievable that can be envisioned by the team

Look for Patterns

- In 1995, a book was published by the “Gang of Four” (Gamma, Helm, Johnson, Vlissides) called Design Patterns
    - It applied the concept of patterns – standard solutions to common problems - to object-oriented software design and described 23 of them
    - The authors did not invent these patterns; they included patterns they found in at least 3 “real” software systems
- Design patterns in software design traces its intellectual roots to work performed in the 1970s by an architect named Christopher Alexander
    - His 1979 book called “The Timeless Way of Building” that asks the question “Is quality objective?”
    - In particular, “What makes us know when an architectural design is good? Is there an objective basis for such a judgement?”
    - His answer was “yes” that it was possible to objectively define “high quality” or “beautiful” buildings



Design Patterns

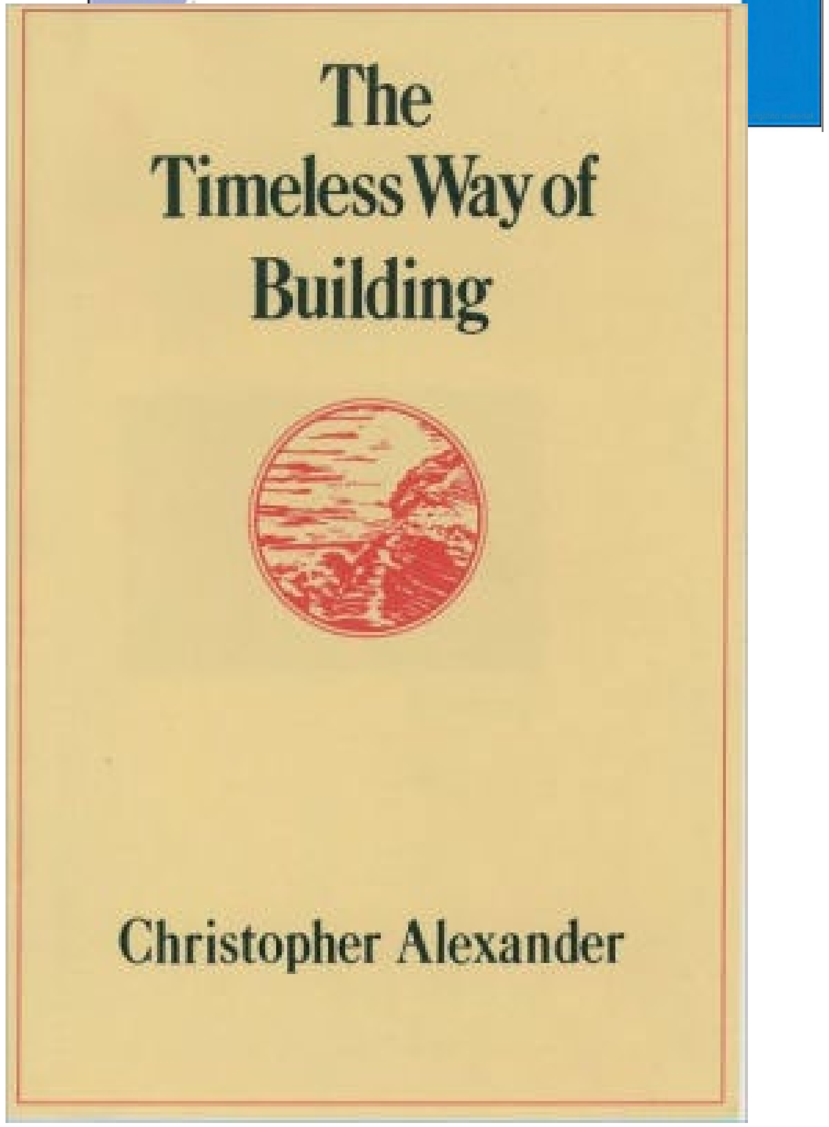
Elements of Reusable Object-Oriented Software

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

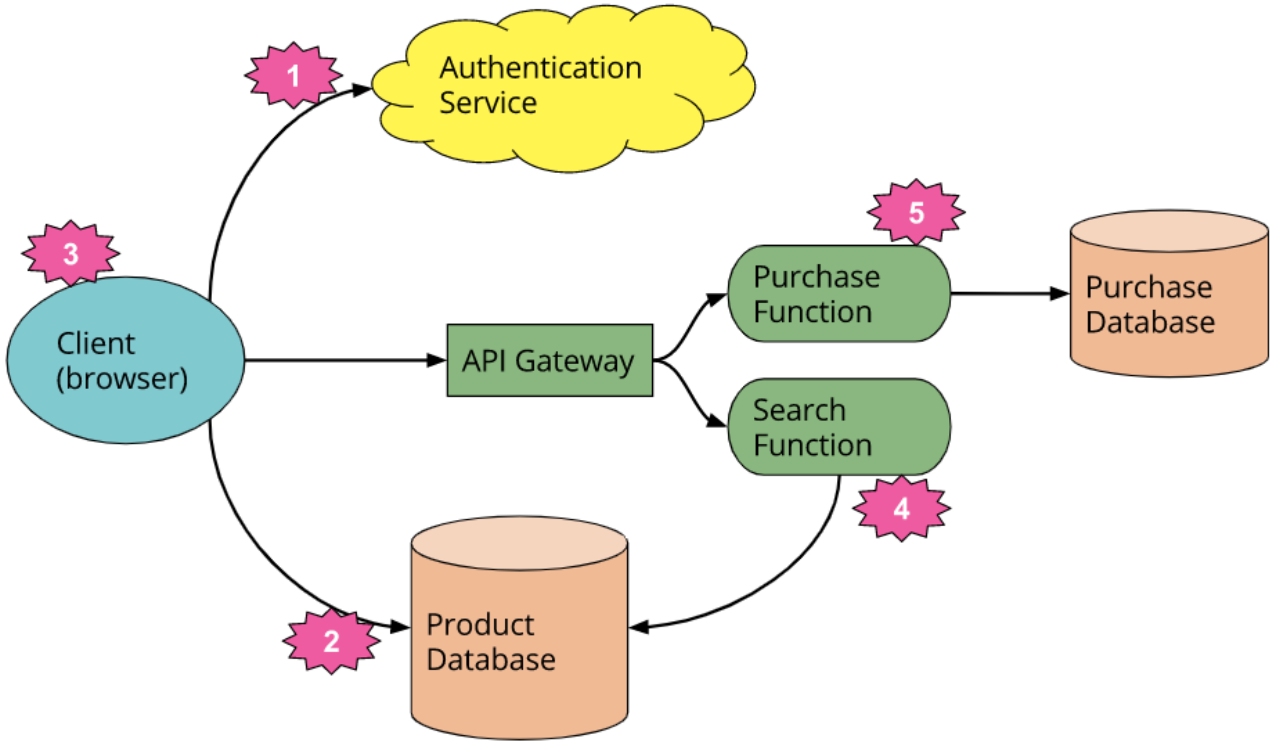


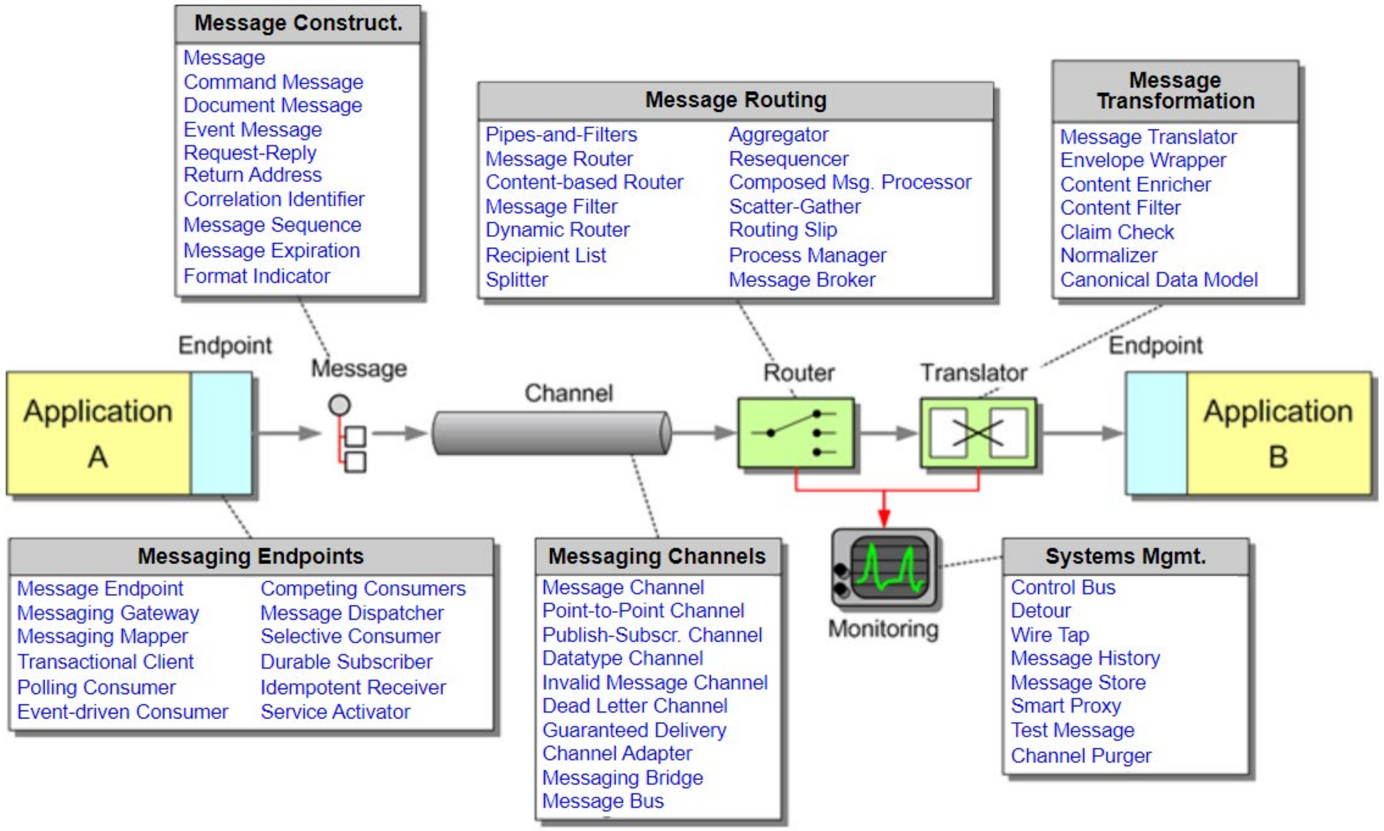
The Timeless Way of Building

Christopher Alexander

Architecture Patterns

- Patterns – structured solutions to identified common problems
  - Experience reuse (not code)
- High Level
  - Application Boundaries
  - Microservices (next)
  - Serverless Architectures →
  - Micro Frontends
  - GUIs and MVC
  - Presentation/Domain/Data Layering
  - https://martinfowler.com/architecture/
- Pattern Libraries
  - https://www.enterprise integrationpatterns.com/ →
  - From the book by Hohpe & Woolf
- Cloud-based Architectures
  - AWS, Azure, GCP, etc.
  - Vendor support for solution patterns is generally strong





Message Construct.
Message
Command Message
Document Message
Event Message
Request-Reply
Return Address
Correlation Identifier
Message Sequence
Message Expiration
Format Indicator

Message Routing
Pipes-and-Filters
Message Router
Content-based Router
Message Filter
Dynamic Router
Recipient List
Splitter
Aggregator
Resequencer
Composed Msg. Processor
Scatter-Gather
Routing Slip
Process Manager
Message Broker

Message Transformation
Message Translator
Envelope Wrapper
Content Enricher
Content Filter
Claim Check
Normalizer
Canonical Data Model

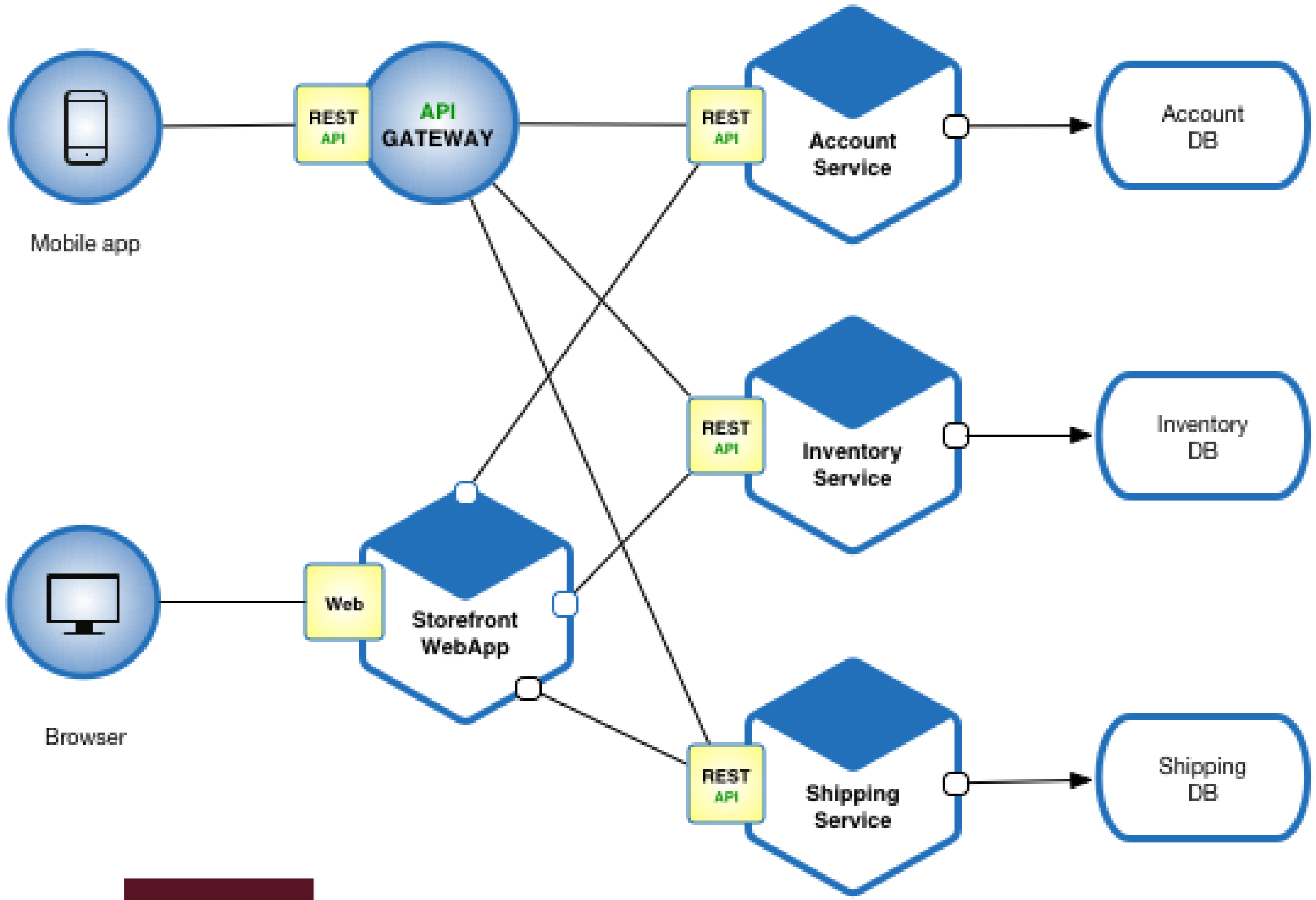
Messaging Endpoints
Message Endpoint
Messaging Gateway
Messaging Mapper
Transactional Client
Polling Consumer
Event-driven Consumer
Competing Consumers
Message Dispatcher
Selective Consumer
Durable Subscriber
Idempotent Receiver
Service Activator

Messaging Channels
Message Channel
Point-to-Point Channel
Publish-Subscr. Channel
Datatype Channel
Invalid Message Channel
Dead Letter Channel
Guaranteed Delivery
Channel Adapter
Messaging Bridge
Message Bus

Systems Mgmt.
Control Bus
Detour
Wire Tap
Message History
Message Store
Smart Proxy
Test Message
Channel Purger

Architectural Pattern Example: Microservices

- Microservice – an independently deployable component of bounded scope that supports interoperability through message-based communication
- Characteristics
  - Small in size
  - Messaging enabled
  - Bounded by contexts
  - Autonomously developed
  - Independently deployable
  - Decentralized
  - Built and released with automated processes
- Goal – balance speed, safety, and scale
- Focus on API Design
  - Standard API Gateway
  - Containers (e.g. Docker)
  - Service Discovery
  - Standard Security
  - Routing
  - Monitoring and Alerting



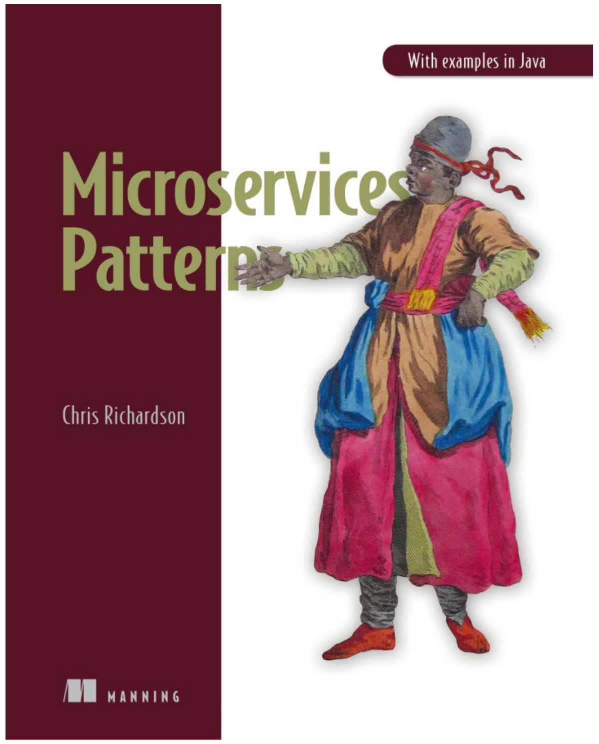


Image from https://microservices.io/patterns/microservices.html

Architecture (and general software) trends

- All about the cloud – leveraging services, tools, and infrastructure (with the IoT) – AWS, Google, Azure
- Containers – Docker and Kubernetes are key technologies
- Microservice Architectures – vs. large monolithic systems (almost OO at larger scale)
- OO rules: Python in general, Java (with Spring) for enterprise scale
    - Still C (and increasingly C++) for embedded systems
    - Watch the newer languages – Rust, Swift, Kotlin, TypeScript…
- For the Web and Cross Platform Apps – JavaScript and React (Angular, Vue also)
- Native App Development – Kotlin/Android, Swift/Apple/iOS
- APIs = REST
- Databases – top choices are still SQL based
    - Always roles for NoSQL/caching DBs: Mongo, Redis, Cassandra
- Agile development processes and DevOps continuous test/integration
- Be user aware – UX and UI design are key skills
- https://towardsdatascience.com/20-predictions-about-software-development-trends-in-2020-afb8b110d9a0

Architecture References

- Martin Fowler's Books
  - Refactoring
  - Patterns of Enterprise Application Architecture
- Hoope & Woolf
  - Enterprise Integration Patterns
- Michael Keeling
  - Design It!
    - Process of doing architecture, basic architecture patterns, good introductory book
- Simon Brown
  - Software Architecture for Developers
    - 2 volume e-books on architecture and the C4 method
    - https://leanpub.com/b/software-architecture
- Brown & Wilson
  - The Architecture of Open Source Applications
    - See how popular open source programs were architected (or not)

Overall PMP Schedule

- Week 3: 9/7
  - Team assignments, sponsor meetings, Charter/project brief assigned
- Week 4: 9/14
  - Sponsor meetings, Charter/project brief completed
- Week 5: 9/21
  - Start development of WBS & Requirements
- Week 6: 9/28
  - WBS & Requirements – pass 1 reviewed by sponsor
- Week 7: 10/5
  - WBS & Requirements
  - Start to build out your Product Backlog (review with Sponsor when able)
  - Start at practice/short Scrum sprint
- Week 8: 10/12
  - WBS & Requirements – (if needed) pass 2 reviewed by sponsor
  - Complete a practice/short Scrum sprint
  - Submit first Sprint Summary Report Form
  - Midterm exam (take home)

Overall PMP Schedule

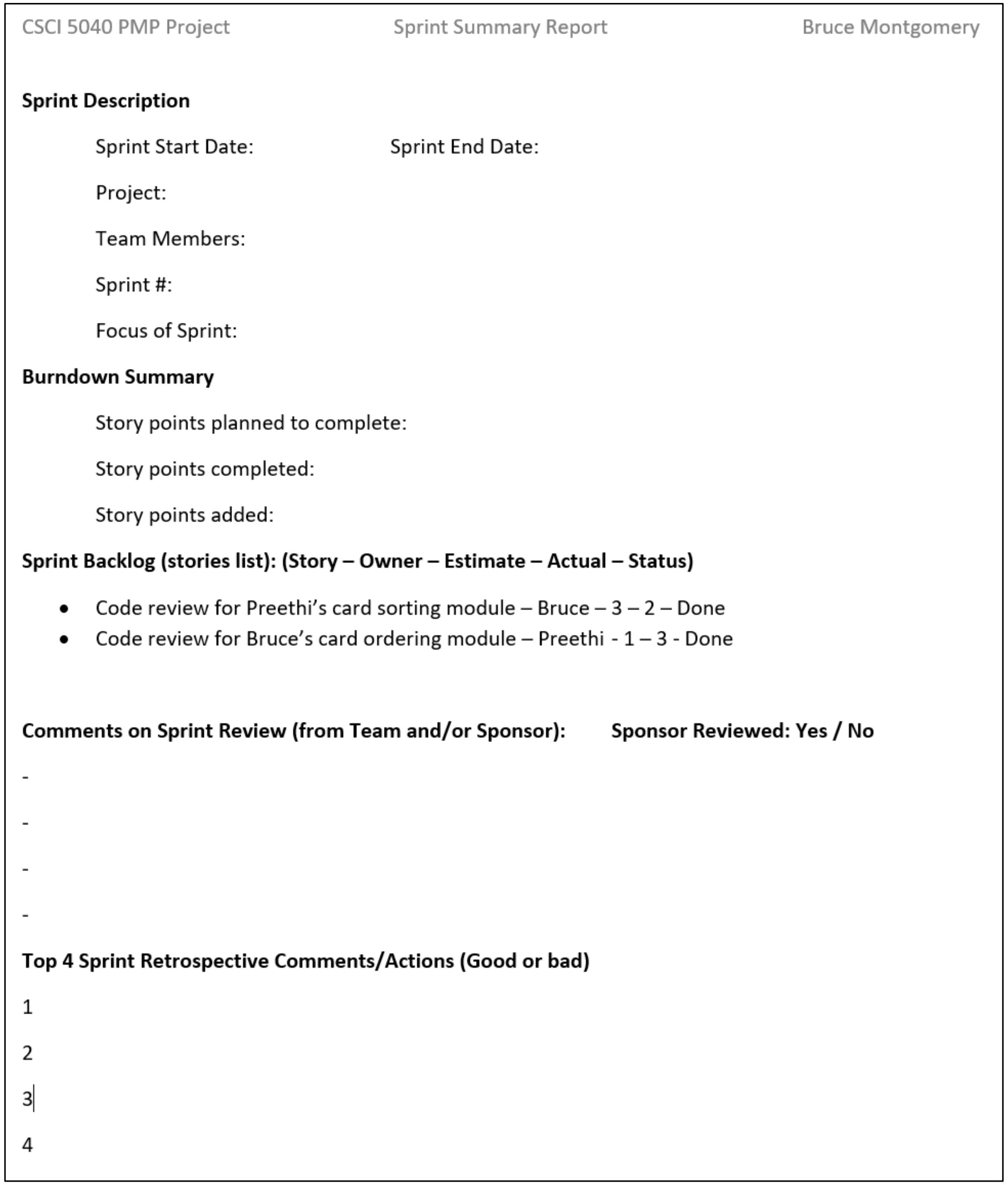
- Week 9: 10/19
  - Begin full two-week Scrum sprint – Architectural/System Design?
- Week 10: 10/26
  - Scrum ends – Submit Sprint Summary Report Form
- Week 11: 11/2
  - Begin sprint – Design/Prototyping?
- Week 12: 11/9
  - Scrum ends – Submit Sprint Summary Report Form
- Week 13: 11/16
  - Begin sprint – Design/Prototyping?
- Week 14: 11/24 (off 11/26-11/27)
  - Sprint ends – Submit Sprint Summary Report Form
- Week 15: 11/30
  - Final sponsor and in-class presentations
  - Assessments: Instructor, GSS, sponsors, peer
- Week 16: 12/7
  - Final exam (take home)

Expectations for your Scrums

- Use your WBS and Requirements to create and maintain your initial Product Backlog (identify epics, broken down into stories)
- Someone on the team gets the role of ScrumMaster
- Sprint planning at start of sprint to establish sprint backlog and story assignments and estimates (planning poker)
  - At least review with or send to Product Owner (Sponsor) if they don't directly participate
- Clearly defined, assigned, estimated stories for each Sprint, tracked in a tool
  - 10 hours per team member for initial sprint
  - 20 hours per team member for 2 week sprints
  - 30 hours per team member for 2 week sprints next semester
- Status of each story in the Sprint: To Do, In Progress, Done, Reviewed
  - You can modify the status categories your team uses
- (At least every other day) 15-minute Scrum stand-up for team
- Sprint Review each sprint with Sponsor
  - Show elements that are done, share the good and not good from the sprint
- Sprint Retrospective each sprint with Team
- Turn in Sprint Summary Report Form (every sprint, no firm deadline)

Sprint Summary Report Form

- Artifact for grading each sprint
- Provide ASAP after each sprint ends
- Graded based on thoroughness of report and clear effort on work items, not on any particular misses or deliveries
- Like anything in your team's Sprint processes, if you need to change the format of the report, please do, as long as the basics are shared



CSCI 5040 PMP Project   Sprint Summary Report   Bruce Montgomery

Sprint Description

Sprint Start Date: Sprint End Date:

Project:

Team Members:

Sprint #:

Focus of Sprint:

Burndown Summary

Story points planned to complete:

Story points completed:

Story points added:

Sprint Backlog (stories list): (Story – Owner – Estimate – Actual – Status)

- Code review for Preethi's card sorting module – Bruce – 3 – 2 – Done
- Code review for Bruce's card ordering module – Preethi - 1 – 3 - Done

Comments on Sprint Review (from Team and/or Sponsor):   **Sponsor Reviewed: Yes / No**

-

-

-

-

Top 4 Sprint Retrospective Comments/Actions (Good or bad)

1

2

3

4

Next Steps

- Sprint summary status form due every two weeks
- Regular weekly status updates still required
- Midterm exam due Wednesday 10/21 at 7 PM
- Speaker 10/22 – Grady Booch
  - New Discussion Topics up on Piazza to ask Grady questions, you can submit questions during the interview as well
- Please try to visit Discussion Topics weekly for comments (and participation grades)
- Teams should be starting their first full two-week Scrum sprint
- Standard stuff
  - Regular meetings with sponsor and me/Preethi should be set
    - Project Status Forms, review and turn in weekly!
  - Aligned with sponsors on tools, project processes, deliverables
  - Always cc Bruce & Preethi on sponsor e-mails
  - Preethi and I are available for questions or ANY other support