



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



INTRODUCTION TO SOFTWARE ARCHITECTURE

Portions taken and adapted from Richard N. Taylor (UCI), Nenad Medv

CSS 553, Spring 2023, 1b Intro SW Arch



A Brief History



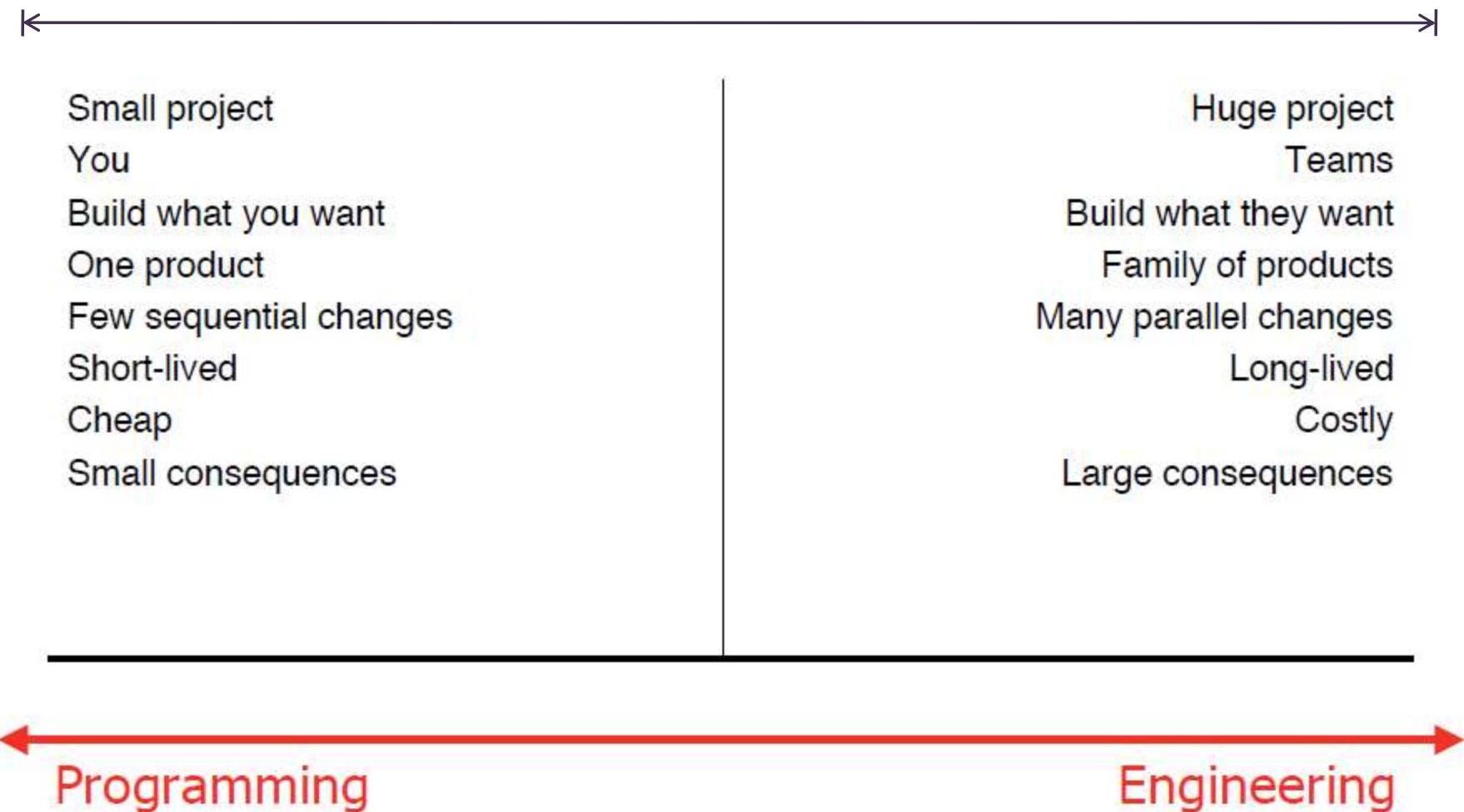
- Software crisis - software projects that were over budget and behind schedule
 - But this is not a temporary problem, but an inherent difficulty in building new and complex applications
- Goal is to address the problems of building large software systems
- Software engineering – late 1960s

Software Engineering



- “A discipline that deals with the building of software systems that are so large or so complex that they are built by a team or teams of engineers” [Ghezzi, Jazayeri, Mandrioli]
- “Multi-person construction of multiversion software” [Parnas]

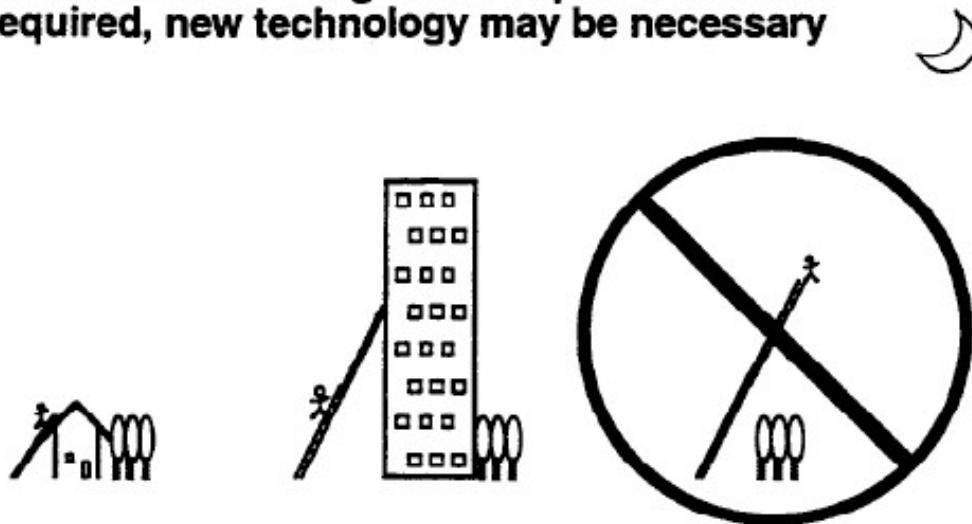
Context



Matters of Scale

← →

When orders-of-magnitude improvement are required, new technology may be necessary



- High powered techniques not appropriate for all problems (Using a forklift to carry a pen)

Why is software engineering difficult?



- Brainstorm for 5 minutes

Why is it difficult to get an error free software?

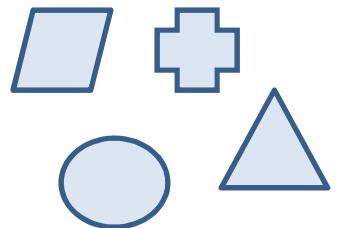


- Four essential problems
 - Complexity
 - Conformity
 - Changeability
 - Invisibility
- And a bunch of accidental ones
 - Human error
 - Poor interfaces
 - Inadequate abstractions
 - Lack of solid mathematical/engineering foundation

Complexity

← →

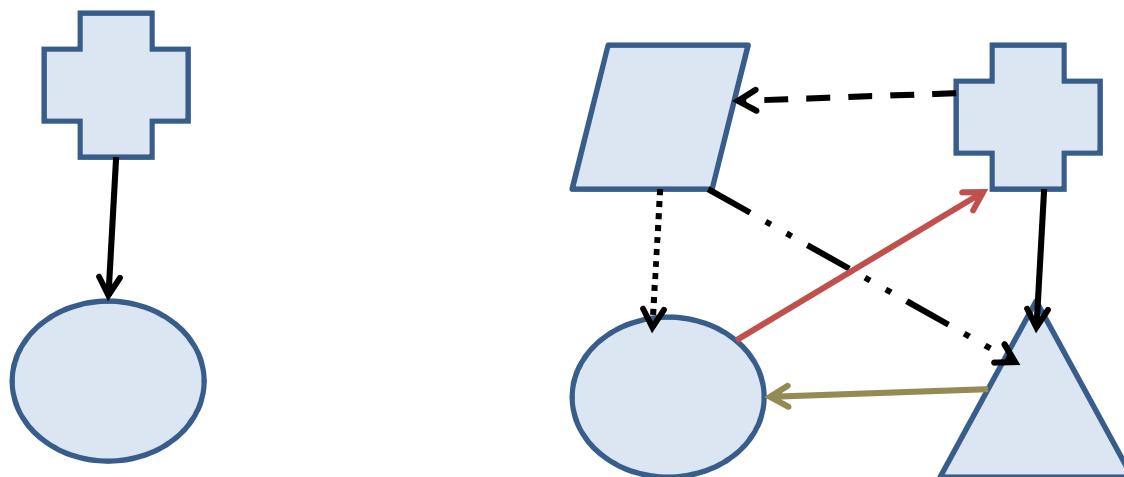
- No two software parts are alike
 - If they are, they would be unified into one
 - Compare to buildings and automobiles



Complexity



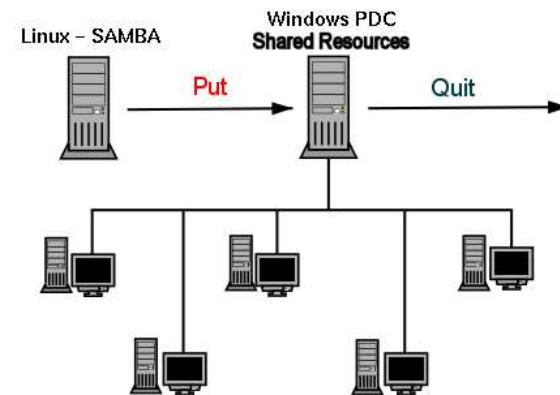
- Complexity grows non-linearly with size
 - Scaling-up is necessarily an increase in the number of different elements
 - Interaction between elements varies



Conformity



- Software must conform to the existing environment
 - Human institutions
 - Existing systems
 - Different for each context



Changeability



- Software is under pressure to change
- Other manufactured items are infrequently changed
 - Cars, computers are superseded by later models
 - Buildings, bridges have a high cost of change

Changeability

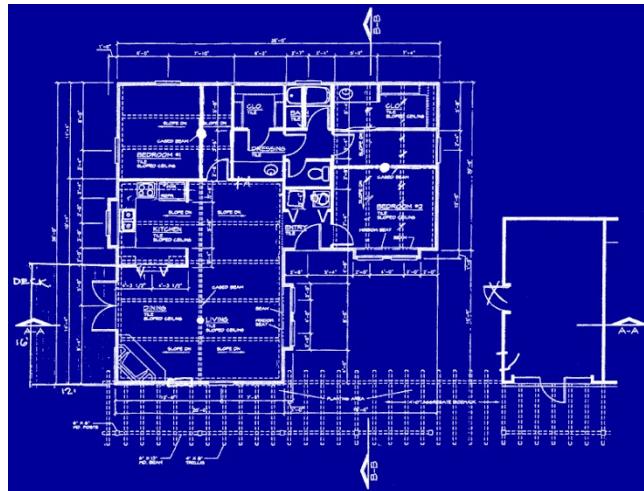


- All successful software get changed
 - Users want additional functionality
 - Software survives beyond the normal life of the machine for which it was written

Invisibility

← →

- Software is invisible and unvisualizable
- Compare with floor plan of a building



Invisibility



- Software is not inherently embedded in space
- Designing software is more difficult
- Communicating concepts about the software is more difficult

Engineering part of SE



- ...creating cost-effective solutions...
- ...to practical problems...
- ...by applying scientific knowledge...
- ...building things...
- ...in the service of people...

Software engineering...



- ...is a fairly young field
- Software development – often do not take advantage of knowledge from successful / unsuccessful software projects

Solution?



- Software architectures
 - A way of encapsulating the experiences and lessons from past projects
 - Through the use of styles or patterns

Writeup: Analogy to Architecture of Buildings



- Review section 1.1 in the book
 - What are the insights presented?
 - Are there other insights?
 - Do you agree with this analogy? Why or why not?
 - Where does the analogy break down?
- Submit in Canvas: Assignments/BuildingAnalogy



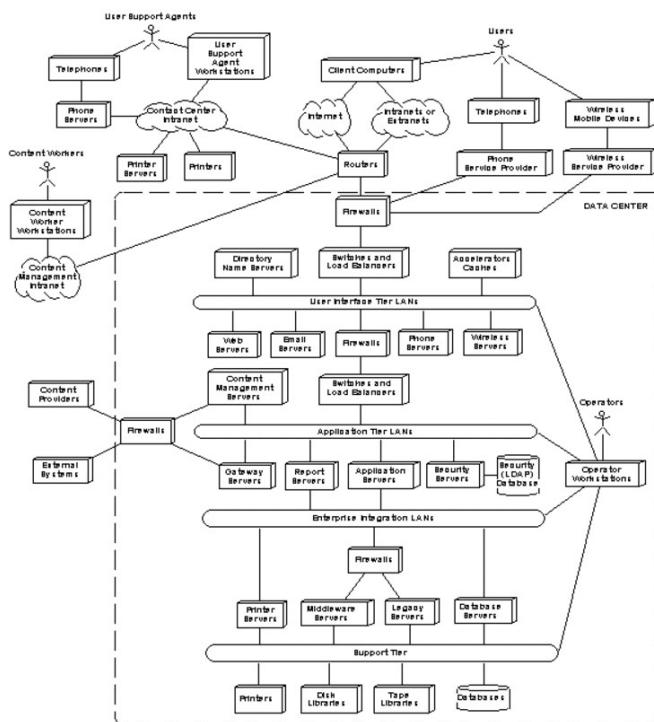
2023, 1b Intro



Analogy to Architecture of Buildings



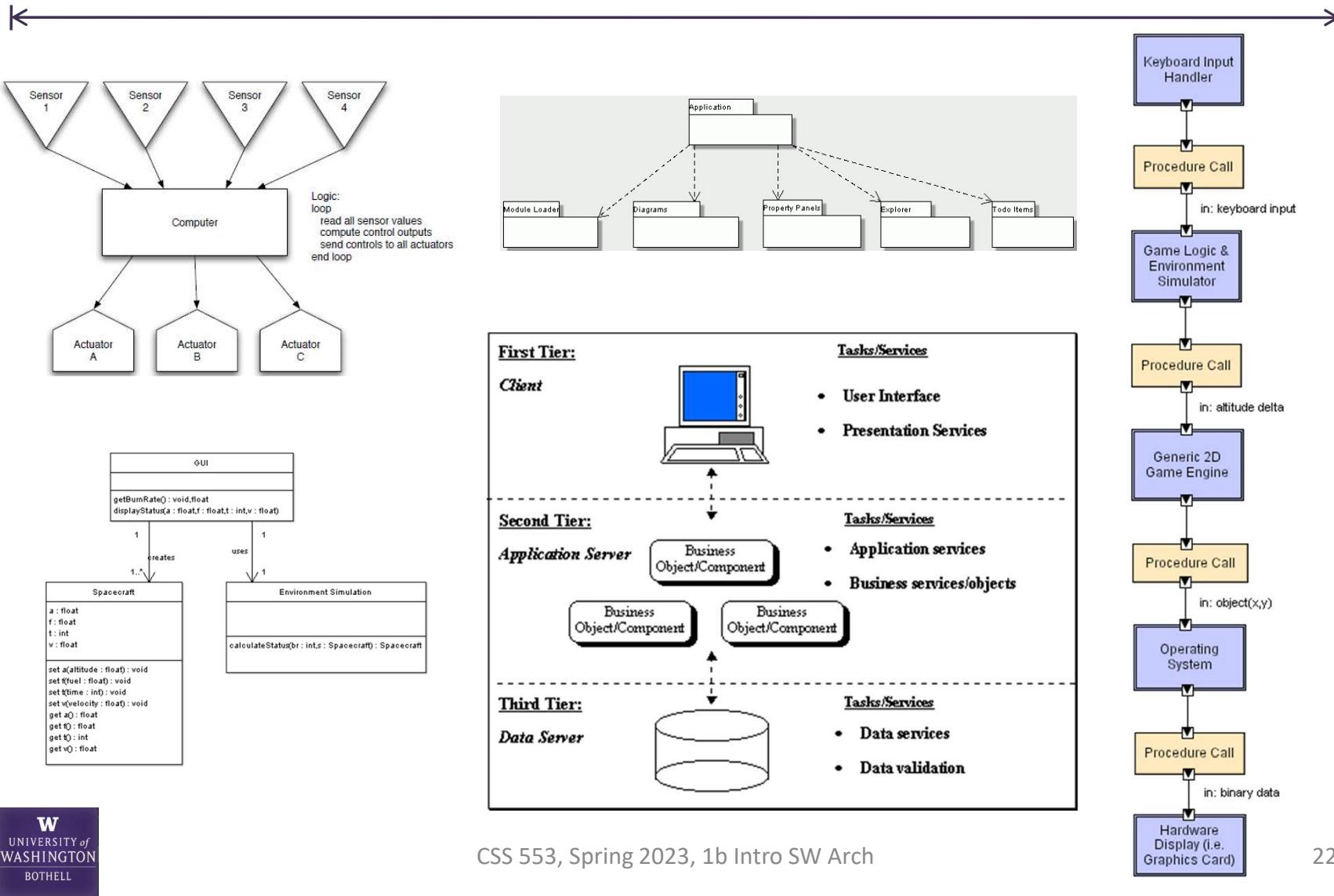
- Do you agree with this analogy? Why or why not?



Architectural Styles or patterns



Architectural styles or patterns



What is Software Architecture?



- Definition: “the set of principal design decisions about the system”
- The blueprint for a software system’s construction and evolution
- Design decisions encompass every facet of the system under development
 - Structure
 - Behavior
 - Interaction
 - Non-functional properties

Software Architecture



- “Principal” implies a degree of importance that grants a design decision “architectural status”



Design decision? Which data structure to use?

Making design decisions



- How do we make design decisions?
 - Look at the requirements document / user story
 - Has this problem been solved before?
 - Yes: How was it done?
 - No: Requires creativity and imagination. Can you use a combination of existing solutions?

Making design decisions



- A requirements specification or a story contains lots of useful information to be leveraged during design
 - Nouns: modules/classes (Sometimes!)
 - Verbs: methods (Sometimes!)
 - Adjectives: properties/attributes/member variables (Sometimes!)
- Why?
 - To identify likely design elements
- Example:
 - “The system shall provide appropriate viewers for the user to read documents in the document store”



Making design decisions



- A requirements specification or a story contains lots of useful information to be leveraged during design
 - Nouns: modules/classes (Sometimes!)
 - Verbs: methods (Sometimes!)
 - Adjectives: properties/attributes/member variables (Sometimes!)
- Why?
 - To identify likely design elements
- Example:
 - “The system shall provide appropriate viewers for the user to read documents in the document store”

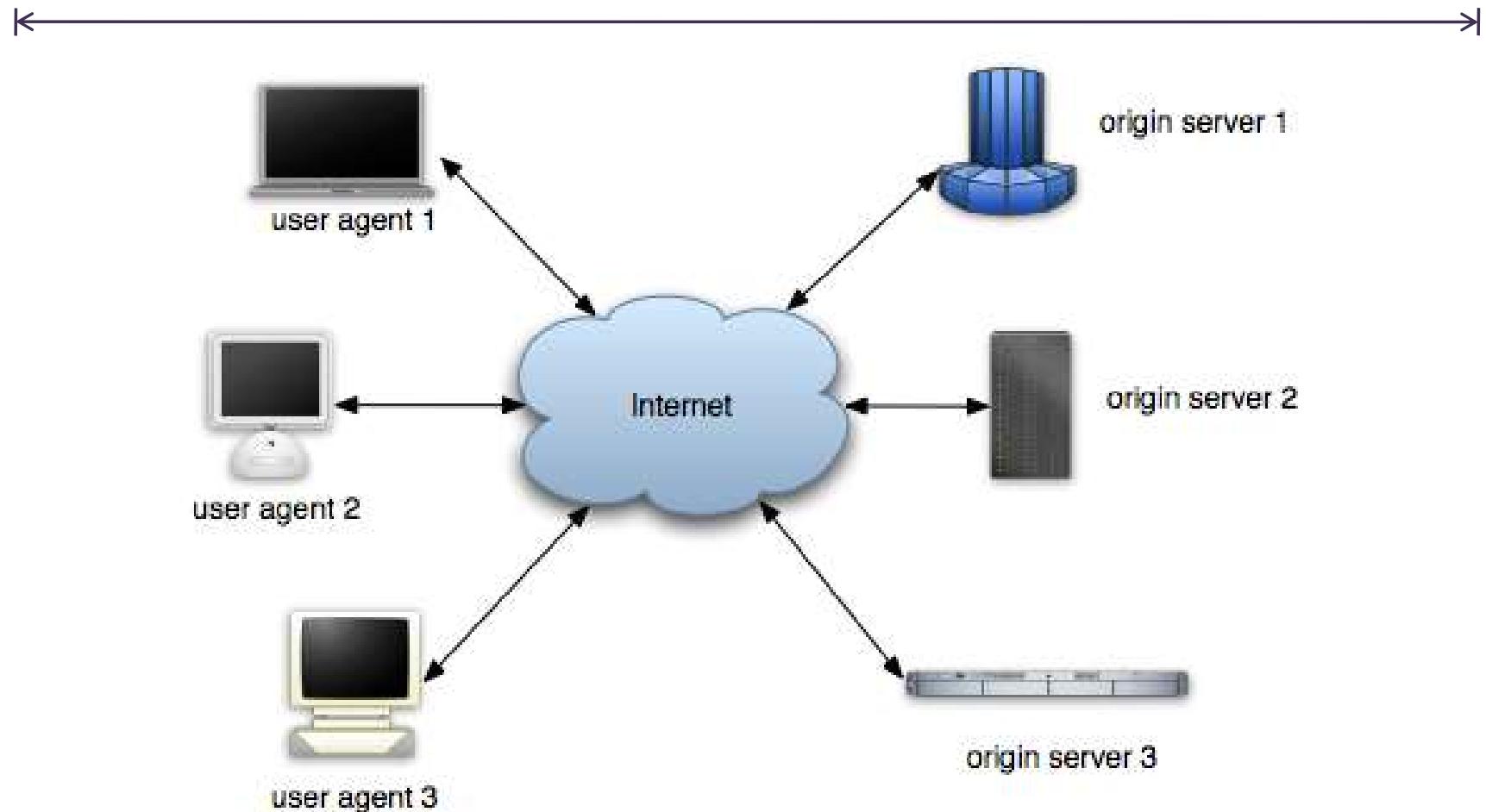


Identifying design decisions



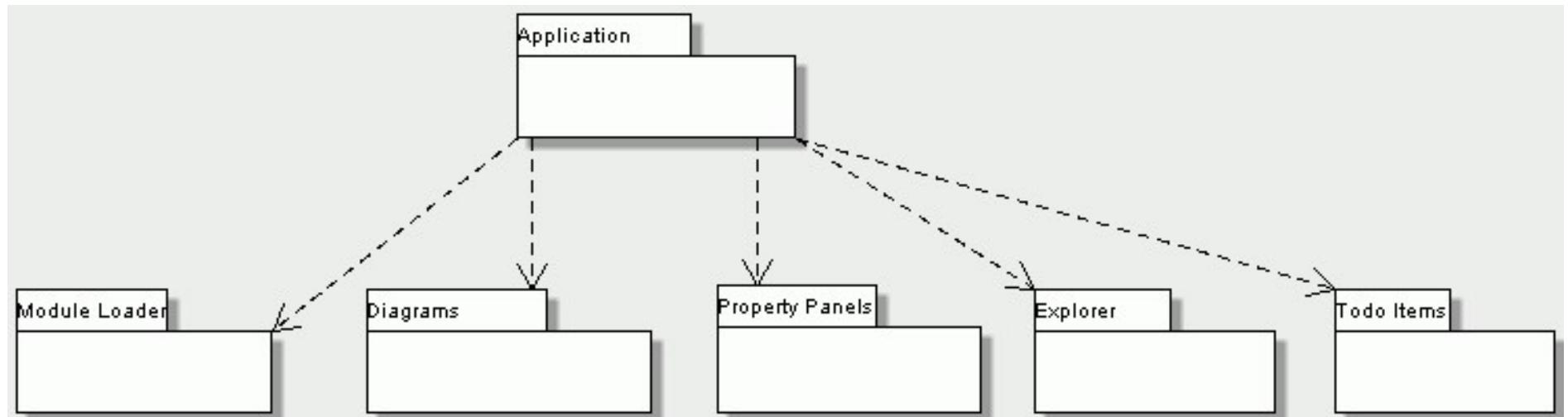
- What design decisions are made in the next systems?

Identifying principal design decisions



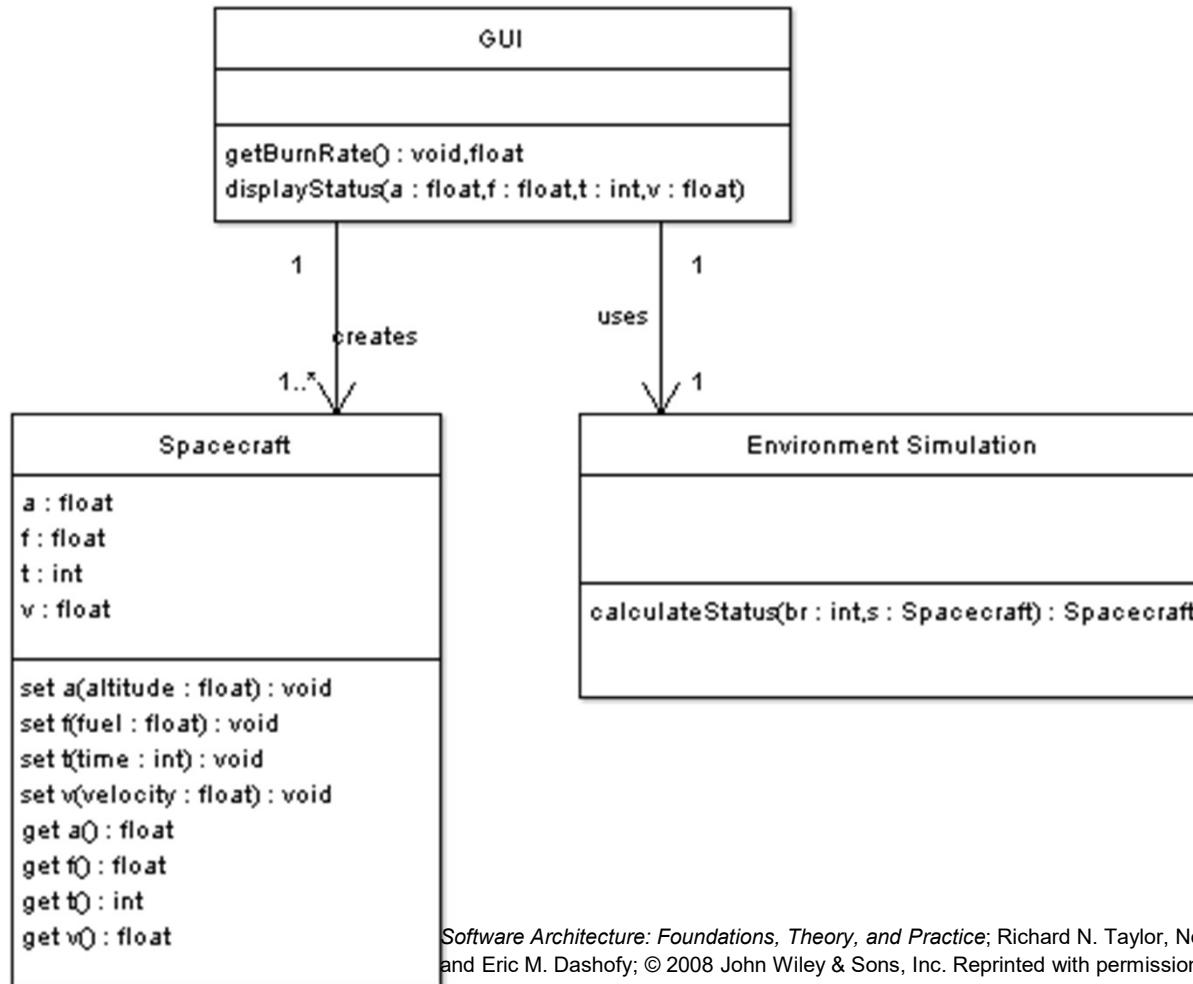
Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

Identifying principal design decisions



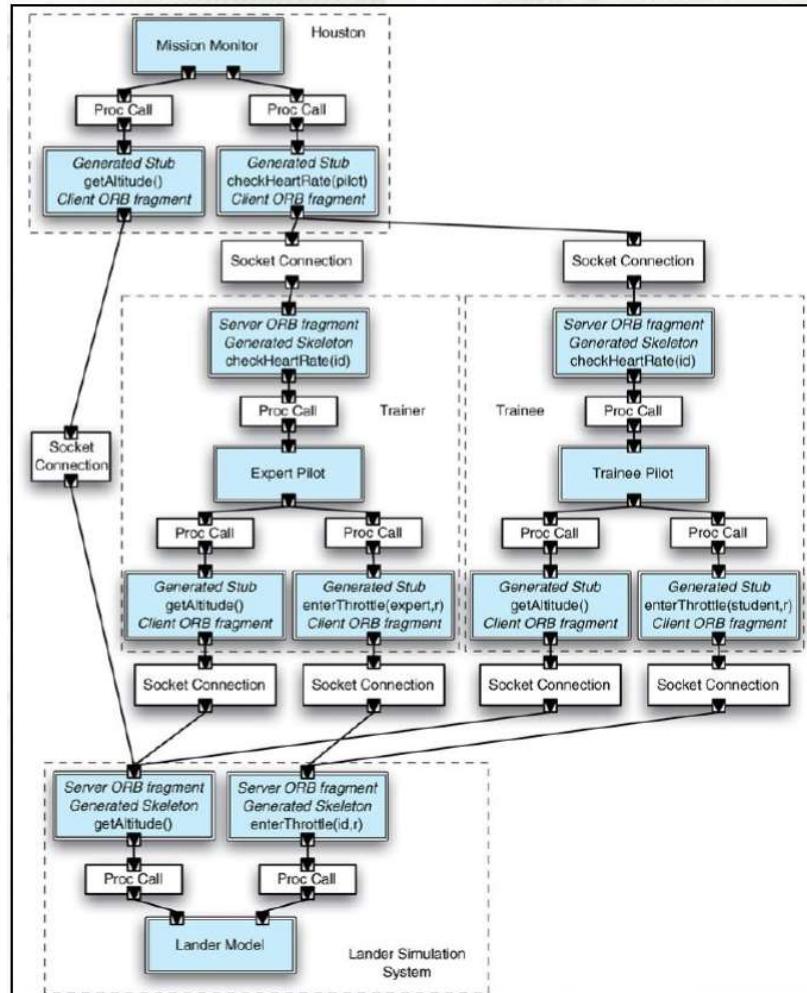
ArgoUML Top Level Subsystem: <http://argouml.tigris.org>

Identifying principal design decisions



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

Identifying principal design decisions



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

Summary



- Software is complex
- So are buildings
 - And other engineering artifacts
 - Building architectures are an attractive source of analogy
- Software engineers can learn from other domains
- They also need to develop—and have developed—a rich body of their own architectural knowledge and experience



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



SOFTWARE ARCHITECTURE IN CONTEXT – PART I

Fundamental Understanding



- Architecture is a set of principal design decisions about a software system
- Three fundamental understandings of software architecture
 - Every application has an architecture
 - Every application has at least one architect
 - Architecture is not a phase of development

Context of Software Architecture



- Requirements
- Design
- Implementation
- Analysis and Testing
- Evolution

Requirements Analysis



- Based on your readings/experience, what happens during requirements analysis?

Requirements Analysis



- Traditional SE – requirements analysis separate from design
 - Is this easy to do?
 - In engineering new products come from the observation of existing solution and their limitations
- Requirements analysis and consideration of design must be pursued at the same time

Question:



- What are two types of requirements?
- Give one example for each type.

Two types of requirements



- Functional requirements
- Non-functional requirements

Non-functional Requirements (NFRs)



NFRs are software qualities or non-functional properties that software must possess

- Correctness
- Reliability
- Robustness
- Performance
- Usability
- Verifiability
- Maintainability
- Reusability
- Portability
- Understandability
- Interoperability

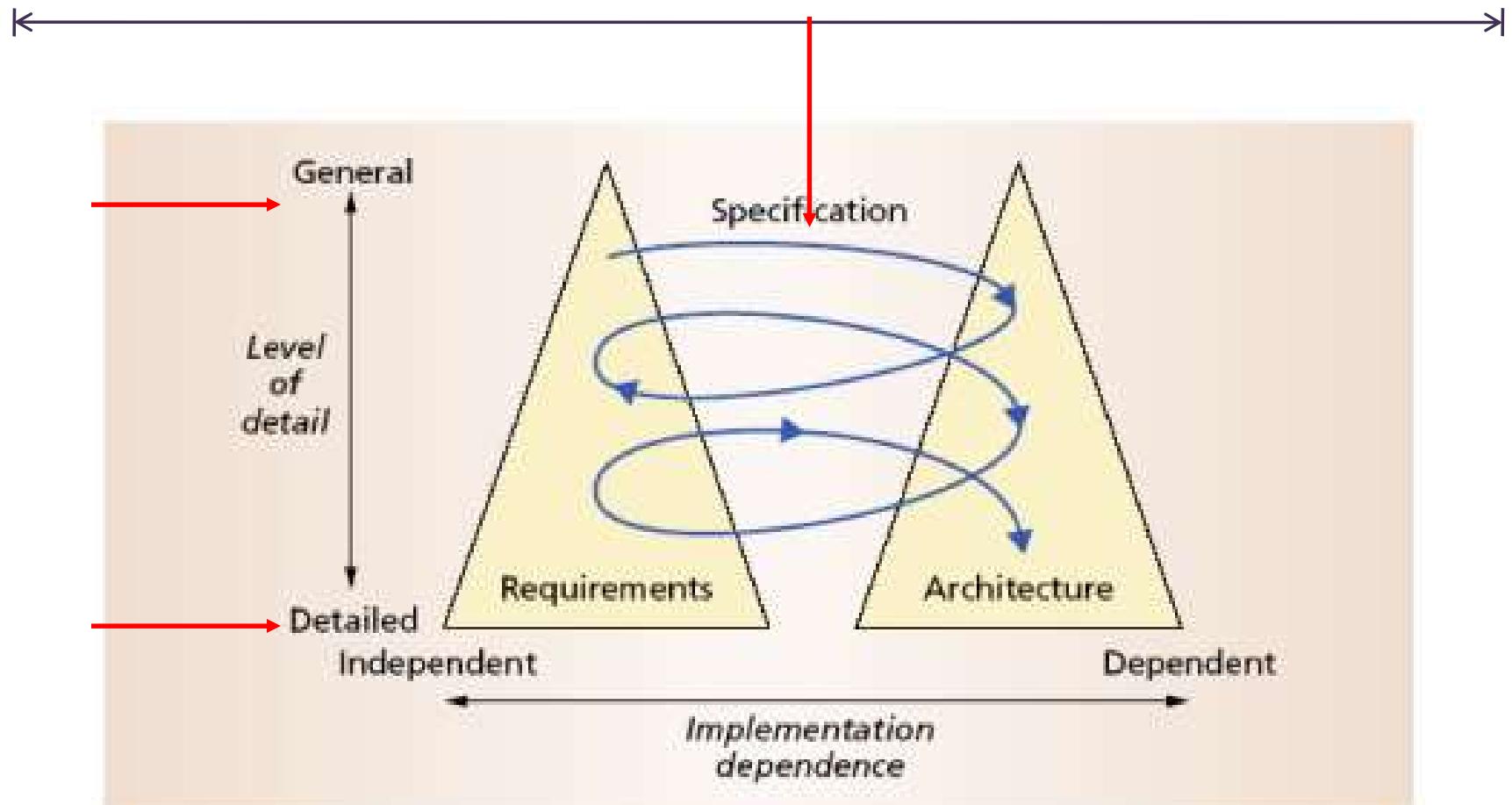
-ilities

How to satisfy...



- ...functional requirements?
 - Modules / Components
- ...non-functional requirements?
 - Architectural choices

The Twin Peaks Model



B. Nuseibeh, "Weaving Together Requirements and Architectures," Computer, vol. 34, no. 3, 2001, pp. 115–119.

Design and Architecture



- Typically in the traditional Design Phase decisions concern
 - A system's structure
 - Identification of its primary components
 - Their interconnections
- Architecture denotes the set of principal design decisions about a system
 - That is more than just structure

Architecture-Centric Design



- Traditional design phase:
 - translate requirements into algorithms to be implemented
- Architecture-centric design
 - stakeholder issues
 - decision about use of COTS component
 - overarching style and structure
 - package and primary class structure
 - deployment issues
 - post implementation/deployment issues

Design Techniques (Review)

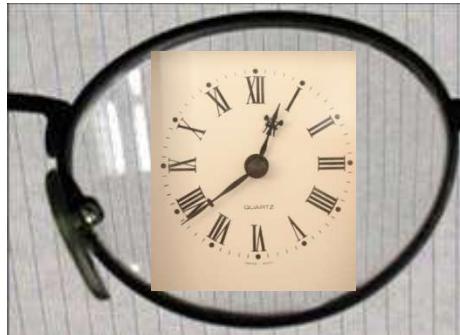


- Basic conceptual tools
 - Separation of concerns
 - Abstraction
 - Modularity
- Two widely adapted strategies
 - Object-oriented design
 - Domain-specific software architectures (DSSA)

Separation of Concerns



- Allows us to deal with different aspects of a problem, so that we can concentrate on each separately
- Different ways concerns may be separated
 - Time
 - Qualities (correctness, performance)
 - Views (data flow, control flow)



Abstraction



- A special case of separation of concerns
 - A technique for mastering complexity
- Process where we identify the important aspects of a phenomenon and ignore its details
- There may be different abstractions of the same reality

Abstraction



2 Different Abstractions

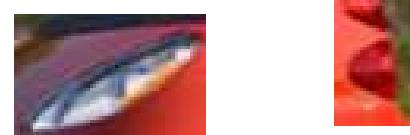
```
/***
 * Method opens the dialog for specifying a URL of the repository
 * @return URI to add as a trace endpoint
 */
public int open() {

    Shell parent = getParent();
    final Shell shell =
        new Shell(parent, SWT.TITLE | SWT.BORDER | SWT.APPLICATION_MODAL);
    shell.setText("Recover Links");
}
```

Modularity



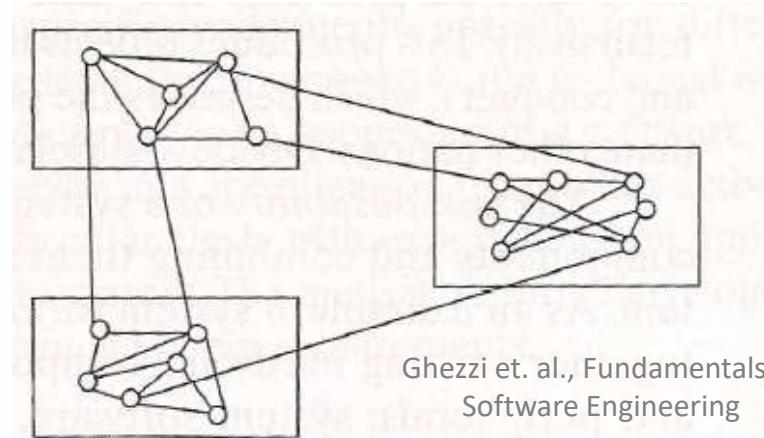
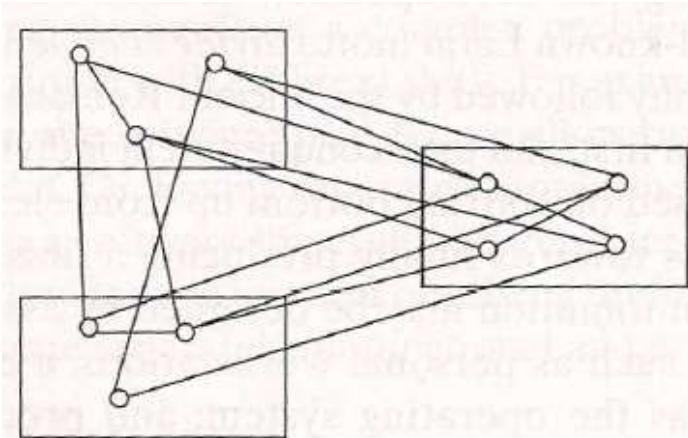
- Complex system may be divided into simple pieces called *modules*
 - Doable building blocks
- Can enhance reusability
 - the next version
 - Another product



Modularity



- High cohesion and low coupling
 - Cohesion: strength of relationship between elements within a module
 - Coupling: strength of relationship between different modules
- Which one?

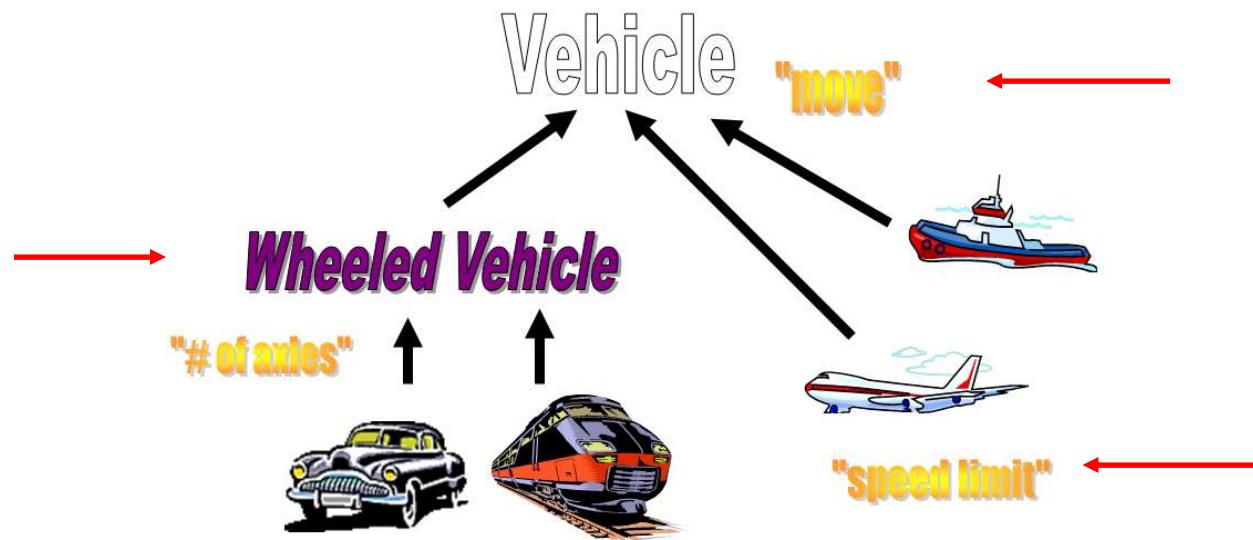


Ghezzi et. al., Fundamentals of Software Engineering

Object-Oriented Design (OOD)



- Objects
 - Main abstraction entity in OOD
 - Encapsulations of state with functions for accessing and manipulating that state



OOD

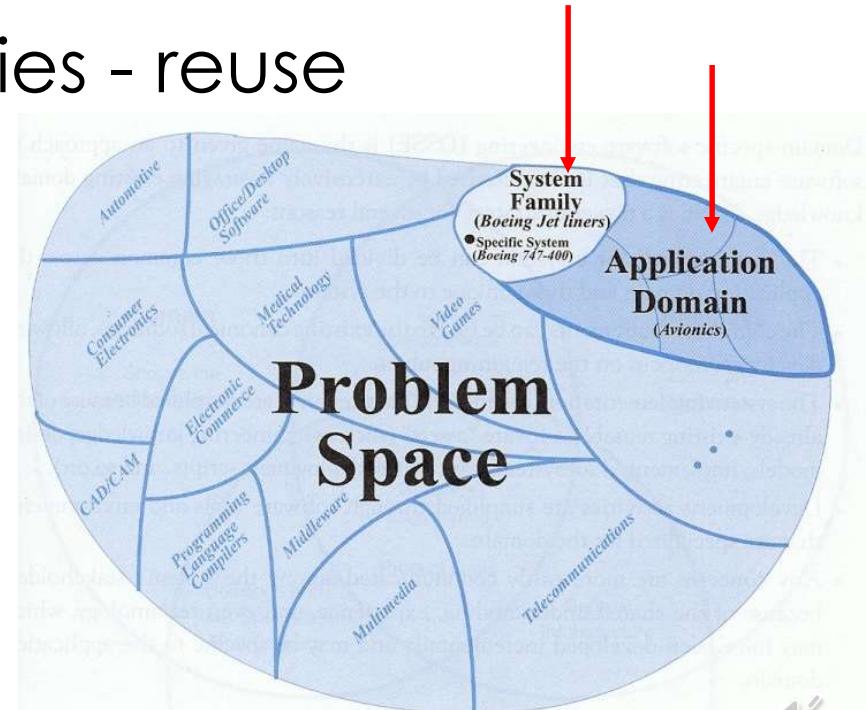


- Should we always follow OOD for all products we design?

Domain Specific Software Architecture (DSSA)



- Best solutions and best practices from past projects within a domain
- New applications - focus on novel variations
- Same functions/properties - reuse
- Applies to product lines





Pros and Cons of OOD



- Pros
 - UML modeling notation
 - Design patterns
- Cons
 - Provides only
 - One level of encapsulation (the object)
 - One notion of interface
 - One type of explicit connector (procedure call)
 - Even message passing is realized via procedure calls
 - OO programming language might dictate important design decisions
 - OOD assumes a shared address space



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



SOFTWARE ARCHITECTURE IN CONTEXT – PART II

Implementation



- The objective is to create machine-executable source code
 - Code conforms to the architecture
 - Or adapt the architecture
 - How much adaptation is allowed?
 - Architecturally-relevant or -irrelevant adaptations
 - Code fully develop all outstanding details of the application

Implementation



- Faithful or
- Unfaithful?

Faithful Implementation



- All structural elements in architecture - implemented in source code
- Source code:
 - No major computational elements that is not in the architecture
 - No new connections between architectural elements not found in the architecture
- Is this realistic?
Overly constraining?
What if we deviate from this?



Unfaithful Implementation



- No architecture?
 - Latent, not documented.
- Are planned and implemented architecture the same?
 - Unable to reason about the architecture in the future
 - Misleads stakeholders regarding what they believe they have vs what they really have
 - Makes any development or evolution strategy based on documented (but inaccurate) architecture doomed to failure

Implementation Strategies



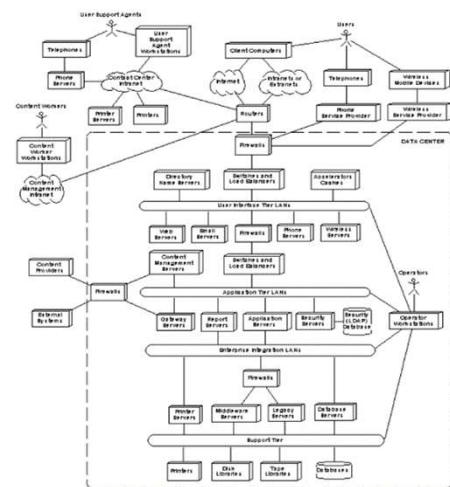
- Generative Techniques
- Frameworks
- Middleware
- Reuse-based techniques
 - COTS, open-source, in-house
- Writing all code manually



Generative Techniques



- Generate code from models
 - parser generator
 - Model-Driven Development (aka Model-Driven Engineering)



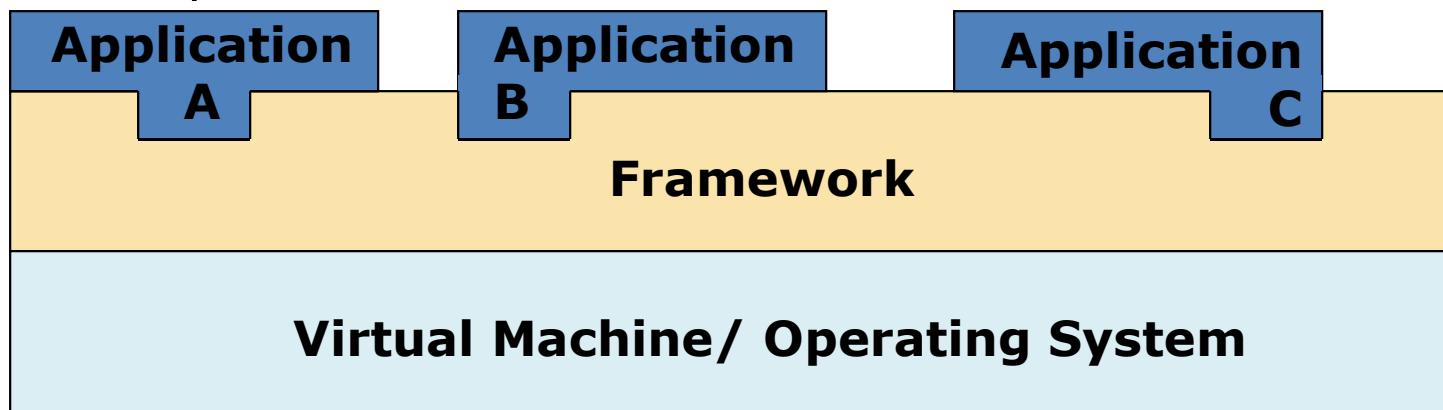
```
public int calculateA() {  
    width = 90;  
    length = 30;  
    :  
    return width * length;  
}
```

Sample tools: <https://tomassetti.me/code-generation/>

Frameworks



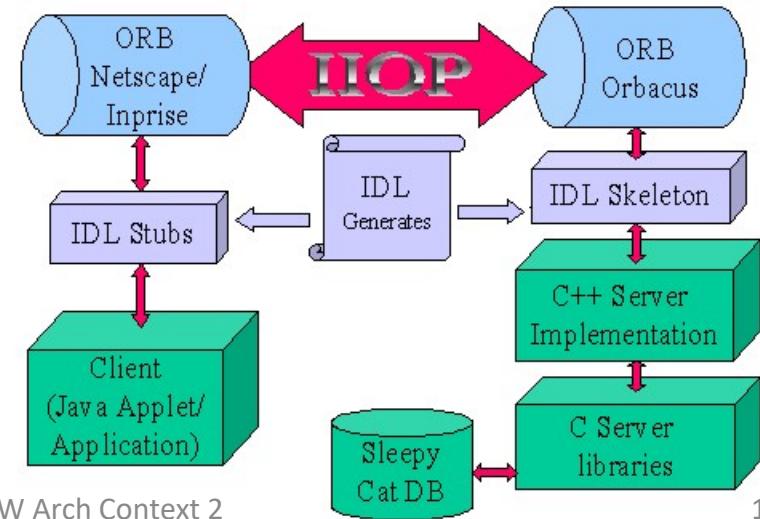
- Collections of source code with identified places where the engineer must “fill in the blanks”
 - Microsoft Foundation Classes (MFC), .NET Framework
 - • Java Swing library
 - • Ruby on Rails



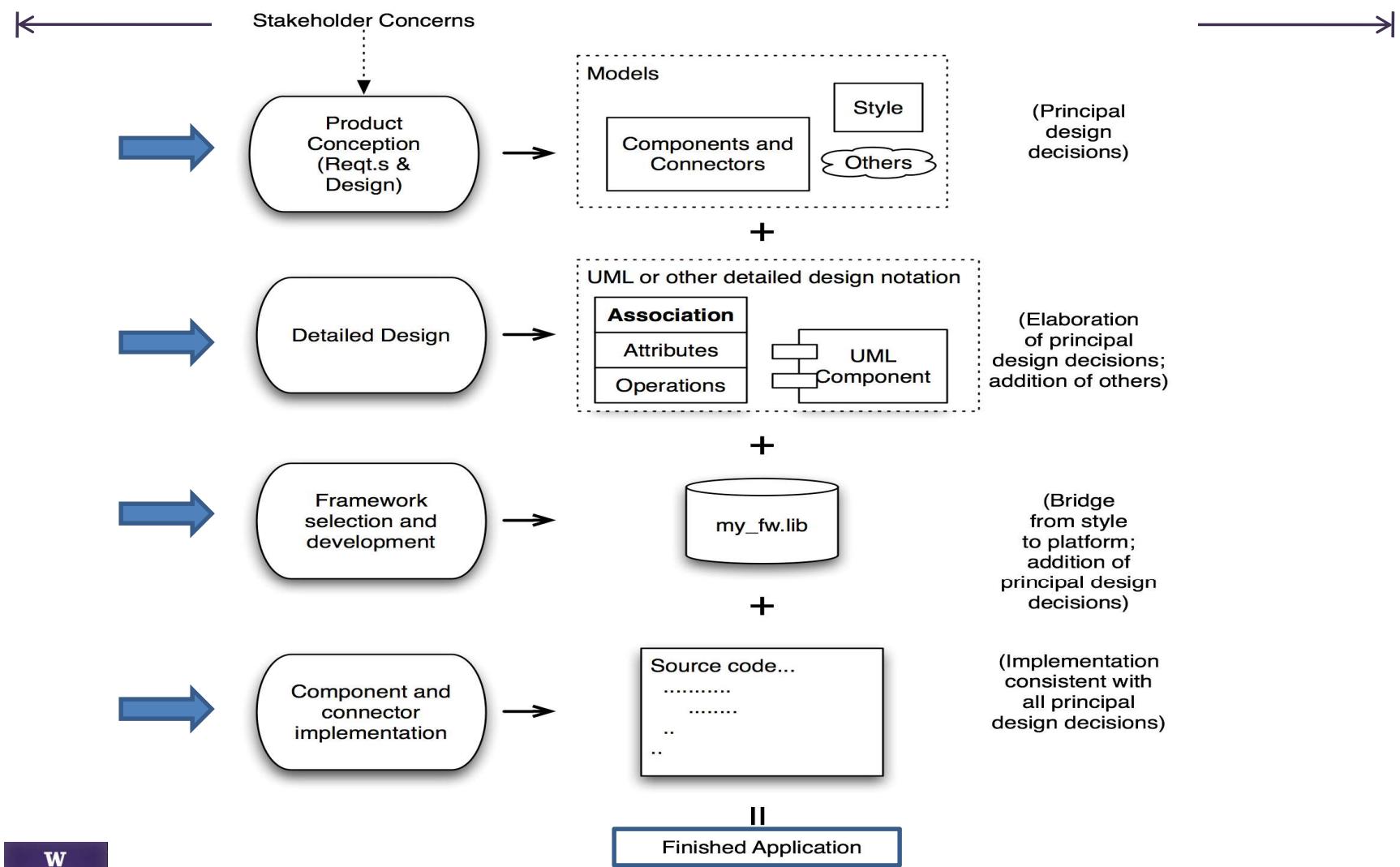
Middleware



- Support communication between components
 - CORBA – Common Object Request Broker Architecture (Object Management Group or OMG)
 - DCOM – Distributed Component Object Model (Microsoft)
 - gRPC - Remote Procedure Call + many features (Google)



How It All Fits Together



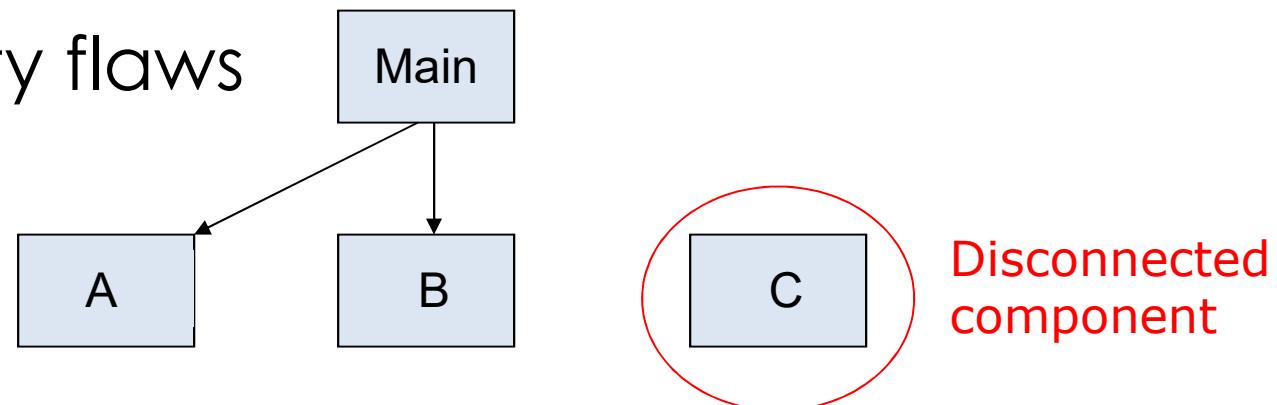
Architecture & Testing

- Architecture analysis
 - system properties
 - functional requirements
- Architecture as a guide to source code testing

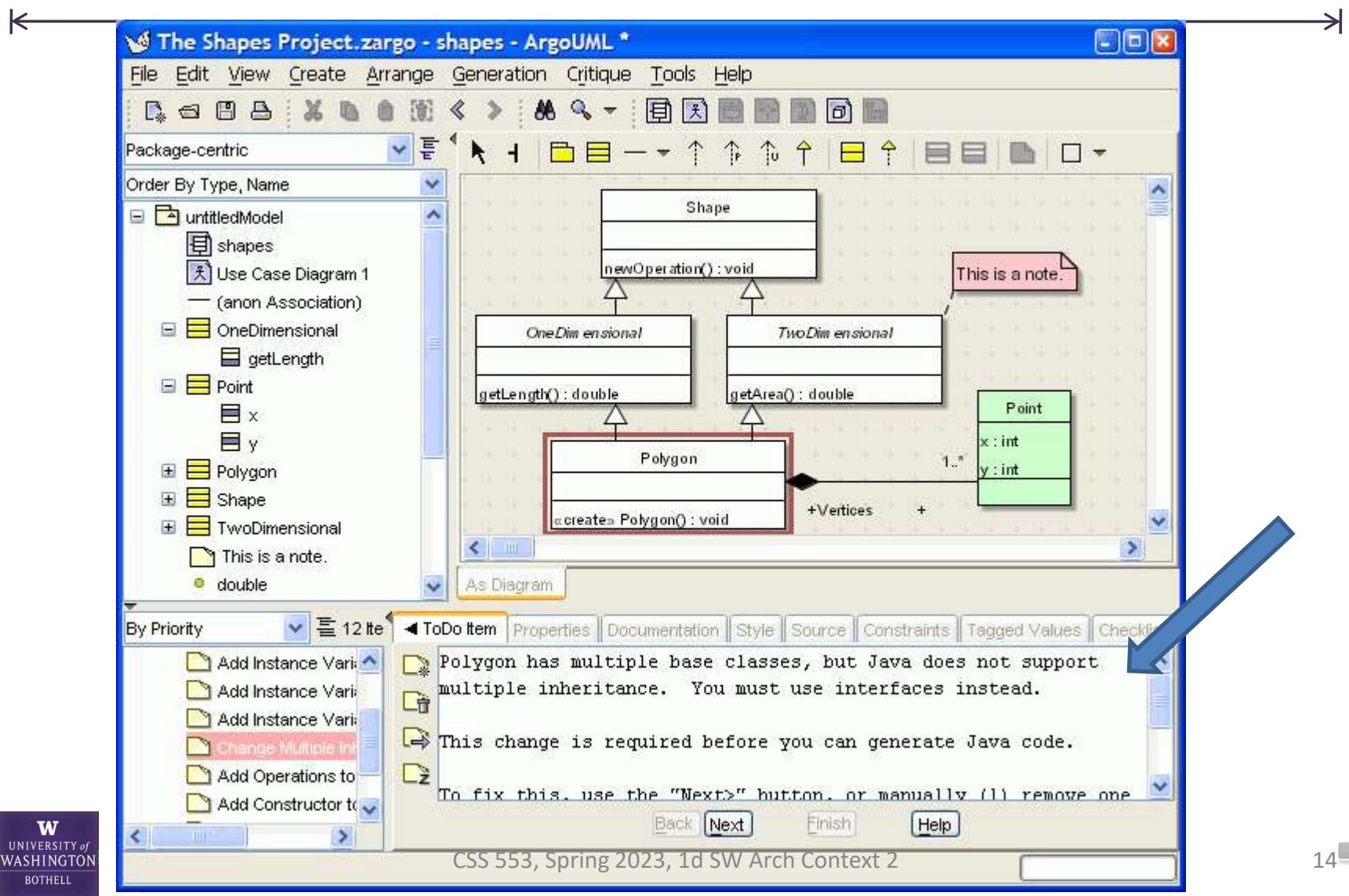
Architecture & Testing



- System properties
 - Component mismatch
 - Incomplete specifications
 - Undesired communication patterns
 - Deadlocks
 - Security flaws

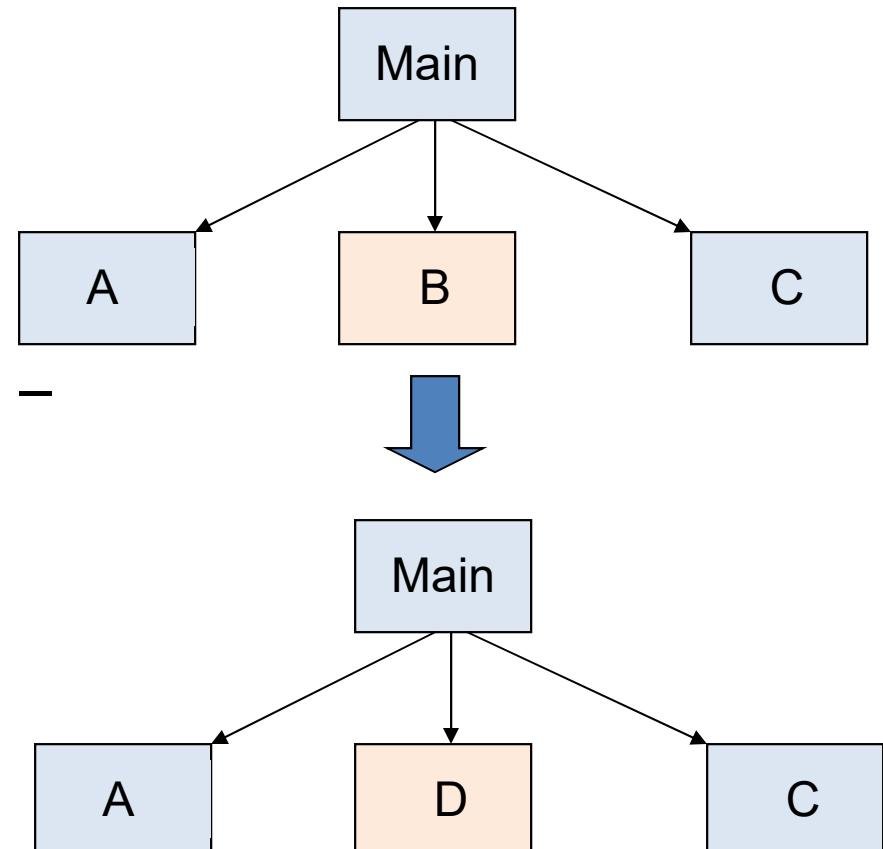


Early Detection of Faults



Architecture & Maintenance

- Guide to system modification
 - Software evolution will occur
 - Traditional approaches – largely ad hoc
 - Center changes to the architecture – future changes will be more systematic



Bottom Line



- Software process discussions make the process activities the focal point
- In architecture-centric software engineering the product becomes the focal point
- No single “right” software process for architecture-centric software engineering exists
 - With discipline, this may be adapted to existing processes





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



BASIC CONCEPTS

Portions taken and adapted from Richard N. Taylor (UCI), Nenad Medvidovic (USC)

What is Software Architecture?



- Definition: “the set of principal design decisions about the system”
- The blueprint for a software system’s construction and evolution
- Design decisions encompass every facet of the system under development
 - Structure
 - Behavior
 - Interaction
 - Non-functional properties

Other definitions...



Perry and Wolf

- Software Architecture = {Elements, Form, Rationale}
what how why

Shaw and Garlan

- Software architecture [is a level of design that] involves
 - The description of elements from which systems are built
 - Interactions among those elements
 - Patterns that guide their composition
 - Constraints on these patterns

Other definitions...



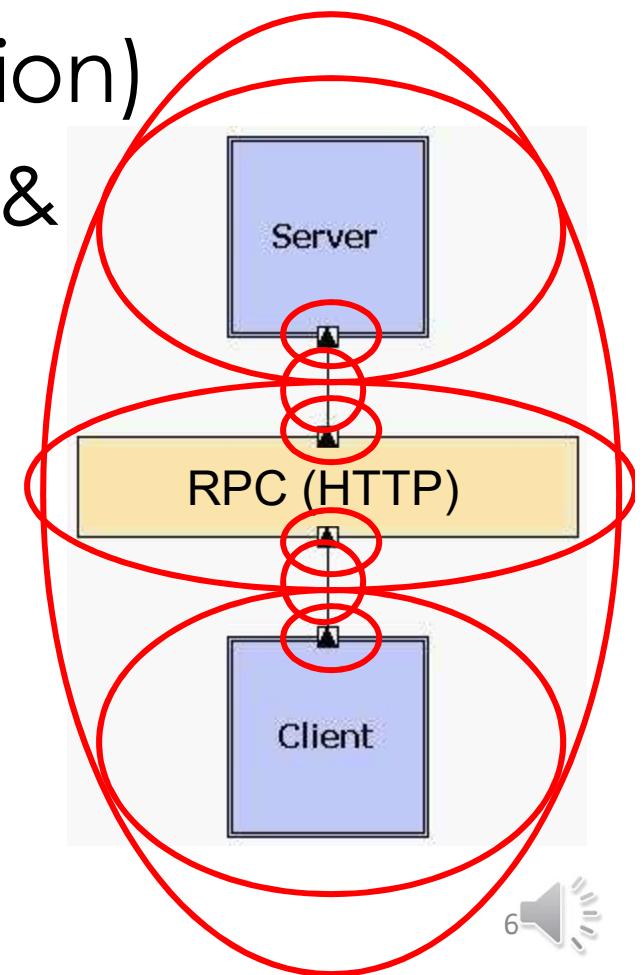
Kruchten

- Software architecture deals with the design and implementation of the high-level structure of software
- Architecture deals with abstraction, decomposition, composition, style, and aesthetics

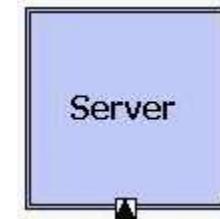
What are similarities?

Basic Software Architecture Elements

- Components (computation)
- Connectors (communication)
- Interfaces (exposed entry & exit points for components and connectors)
- Links
- Configuration (topology)



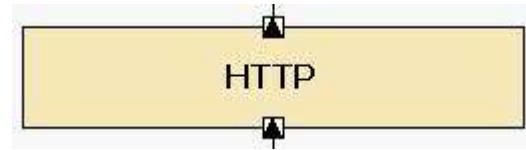
Components



← →

- Encapsulate processing and/or data in a system's architecture are referred to as *software components*
- An architectural entity that
 - Encapsulates a subset of the system's functionality and/or data
 - Restricts access to that subset via an explicitly defined interface
 - Has explicitly defined dependencies on its required execution context
- Typically provide application-specific services

Connectors



- *Interaction* may be more important and challenging than functionality of individual components
- Connector – regulate interactions between components
- Connectors - usually simple procedure calls or shared data accesses
 - Much more sophisticated and complex connectors are possible!
- Connectors - provide application-independent interaction facilities

Examples of Connectors



- Procedure call connectors
- Shared memory connectors
- Message passing connectors
- Streaming connectors
- Distribution connectors
- Wrapper/adaptor connectors

Configurations



- Composition of components and connectors in a specific way to achieve a system's objective
- Aka topology

Temporal Aspect



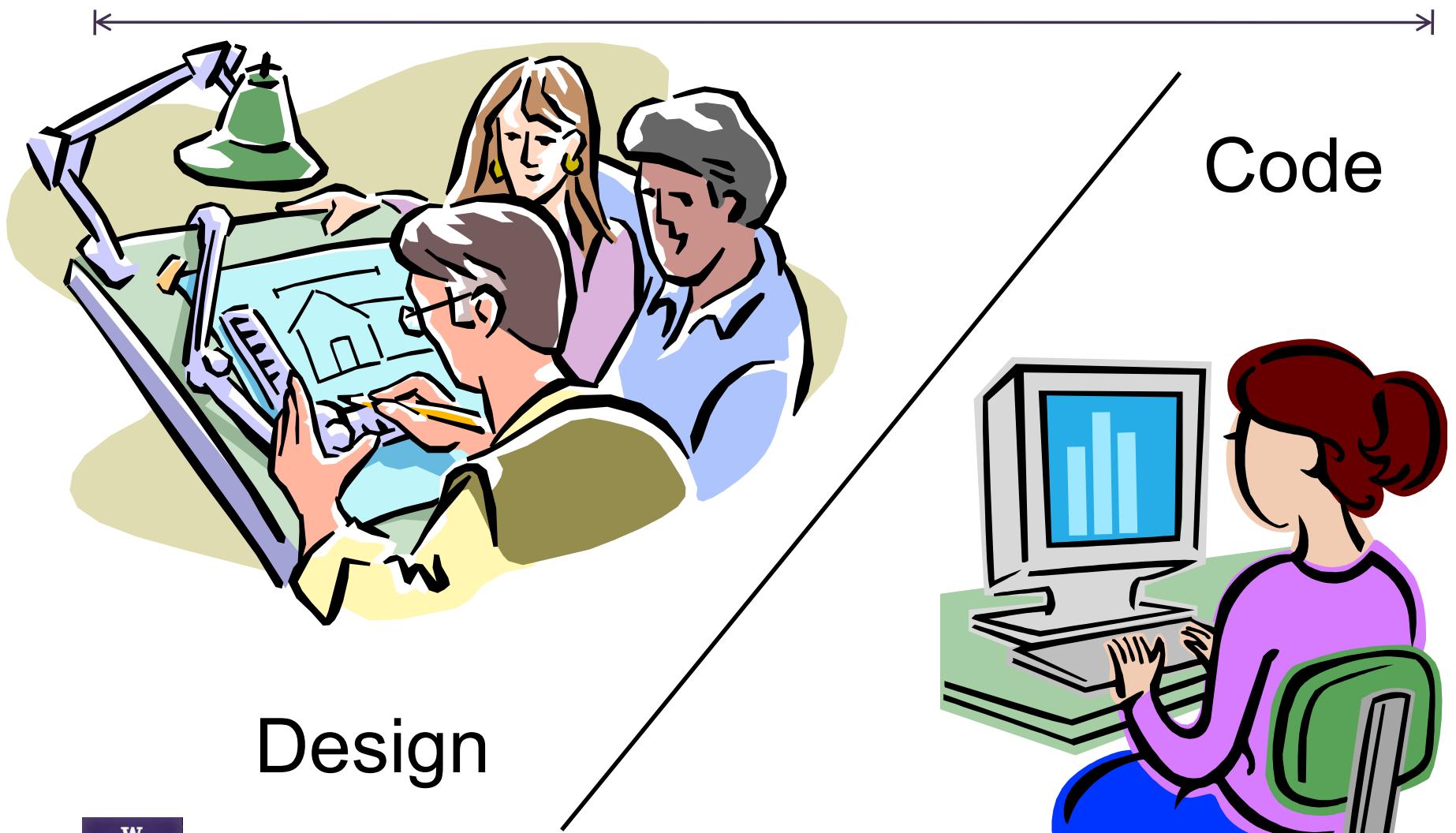
- Design decisions are made and unmade over a system's lifetime
→ architecture has a temporal aspect
- At any given point in time the system has only one architecture
- A system's architecture will change over time

Prescriptive vs. Descriptive Architecture

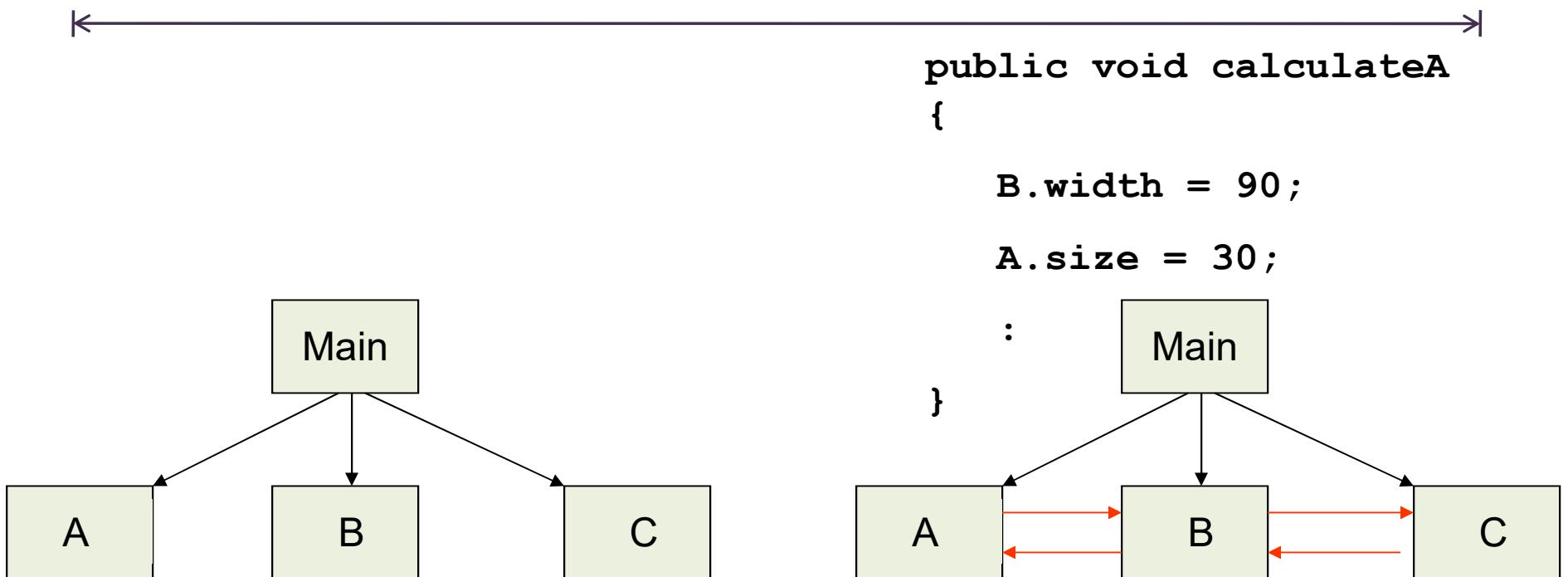


- Prescriptive architecture
 - design decisions made prior to the system's construction
 - *as-conceived* or *as-intended* architecture
- Descriptive architecture
 - how the system has been built
 - *as-implemented* or *as-realized* architecture

Current practice...



Typical scenario



“as conceived” or “as-intended”

“as-implemented” or
“as-realized”

Design

CSS 553, Spring 2023, 2a Basic Concepts

Code

Architectural Evolution



- Ideally prescriptive architecture is modified first
- In practice, the system (descriptive architecture) is often directly modified
- Reasons
 - Developer sloppiness
 - Perception of short deadlines which prevent thinking through and documenting
 - Lack of documented prescriptive architecture
 - Need or desire for code optimizations
 - Inadequate techniques or tool support

Typical scenario



```
public void calculateA  
{  
    B width = 90.  
}
```

- Are the two architectures consistent with one another?
- Which architecture is “correct”?
- What criteria are used to establish the consistency between the two architectures?
- On what information is the answer to the preceding questions based?

“as conceived” or “as-intended”

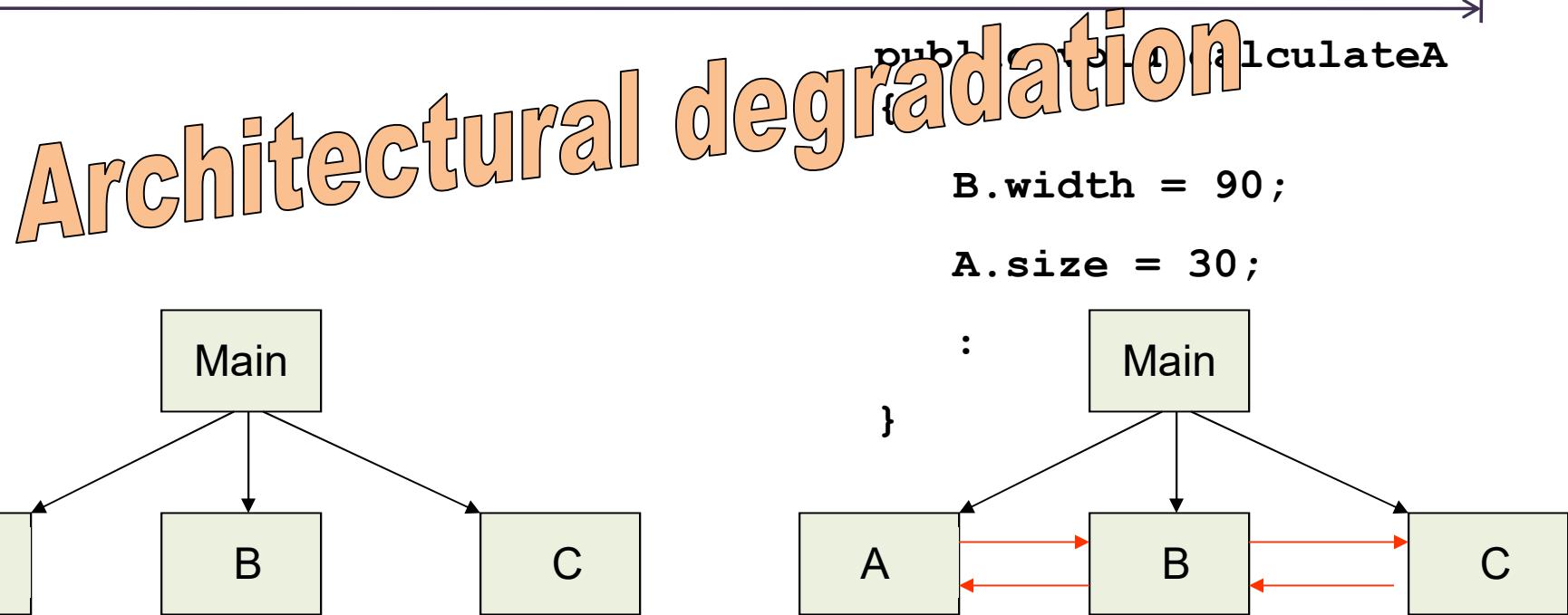
as-implemented or
“as-realized”

A

P

C

Typical scenario



Prescriptive Architecture

“as conceived” or “as-intended”

Descriptive Architecture

“as-implemented” or
“as-realized”

Design

CSS 553, Spring 2023, 2a Basic Concepts

Code

Typical Scenario



- Architectural degradation
 - Expensive and dangerous
 - ...what happens when the original developers leave?
 - ...what if you have a large system to modify? How do you know where to start?
 - ...what if you are working on a critical system?

Architectural Degradation



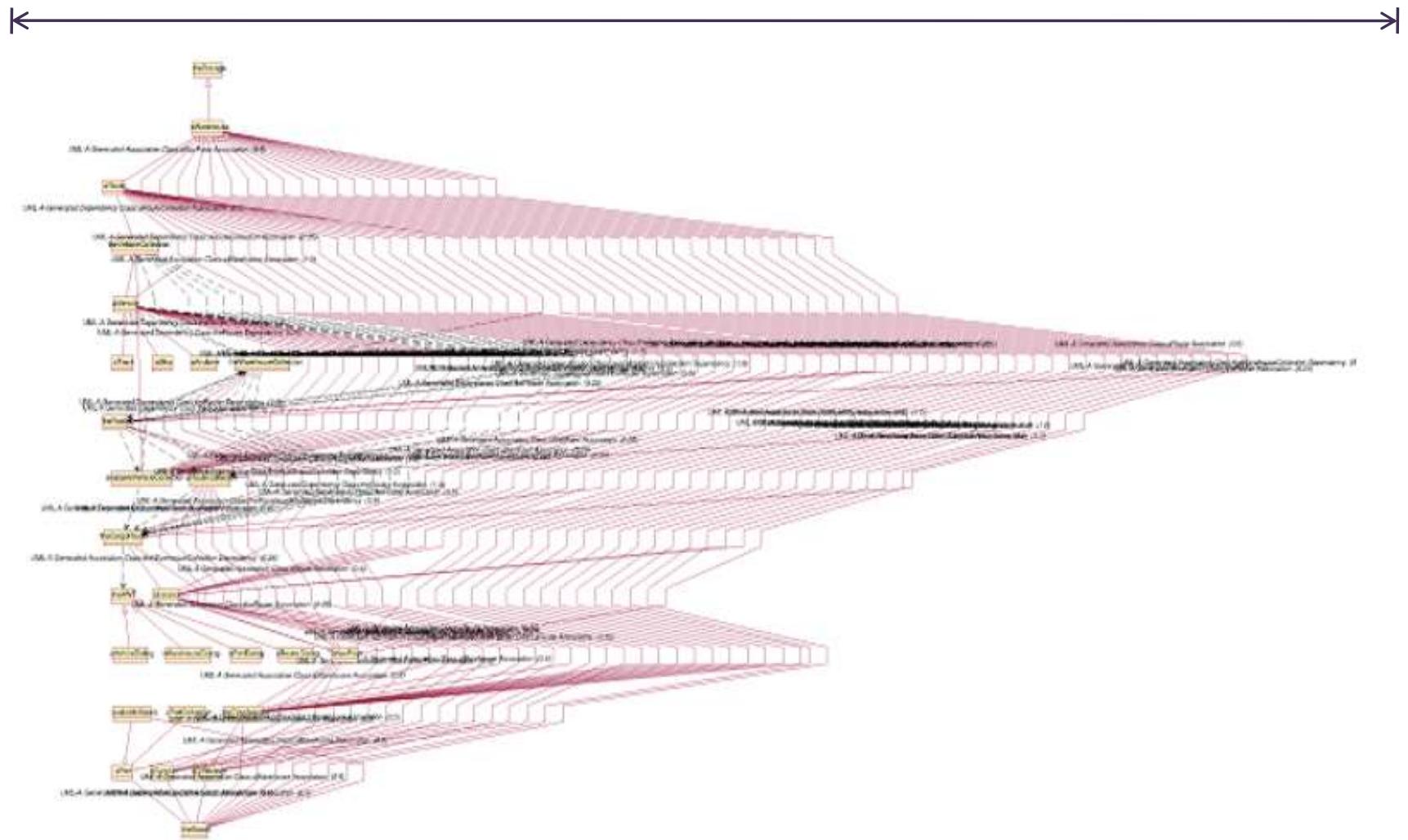
- Two related concepts
 - Architectural drift
 - Architectural erosion
- **Architectural drift** - introduction of principal design decisions into a system's descriptive architecture that
 - Are not included in the prescriptive architecture
 - But do not violate any of the prescriptive architecture design decisions
- **Architectural erosion** - introduction of architectural design decisions into a system's descriptive architecture that violate its prescriptive architecture

Architectural Recovery



- ...necessary when architectural degradation occurs
- **Architectural recovery** - process of determining a software system's architecture from its implementation-level artifacts & documentation
- Implementation-level artifacts can be
 - Source code
 - Executable files
 - Java .class files

Implementation-Level View of an Application

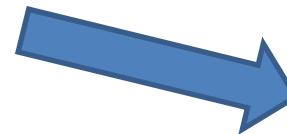


Deployment



- A software system cannot fulfill its purpose until it is deployed
 - Executable modules physically placed on the hardware devices on which they are supposed to run
- Deployment view - can be critical in assessing whether the system will be able to satisfy its requirements
- Possible assessment dimensions
 - Available memory
 - Power consumption
 - Required network bandwidth

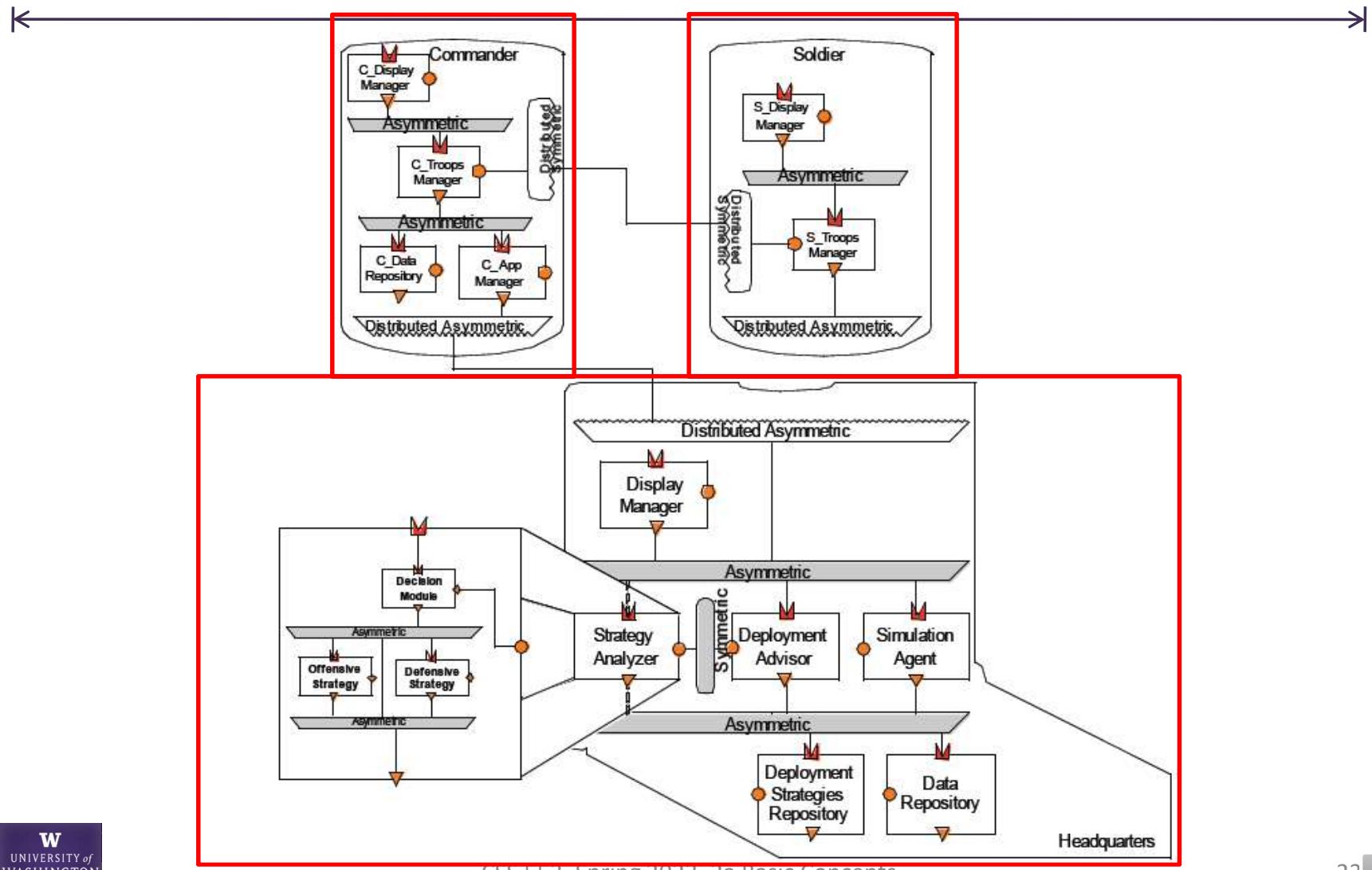
software



***Customer's
machine***



A System's Deployment Architectural Perspective



Stakeholders in a System's Architecture



- Architects
- Developers
- QA Engineers
- Managers
- Customers
- Users
- Vendors

Question:



- Software architects are often asked the difference between architecture and design. Given the definition of software architecture provided in this lecture, can you distinguish between the two? Explain.



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



Outline



- Process of Design
- Tool: Design Recovery

Part I

PROCESS OF DESIGN

Portions taken and adapted from Richard N. Taylor (UCI), Andre van der Hoek (UCI)

How Do You Design?



Where do architectures come from?

Creativity

- 1) Fun!
- 2) Fraught with peril
- 3) May be unnecessary
- 4) May yield the best

- 1) Efficient in familiar terrain
- 2) Not always successful
- 3) Predictable outcome (+ & -)
- 4) Quality of methods varies

Method

Typical Design Process



- **Feasibility stage:** identify *set of feasible concepts for the design as a whole*
- **Preliminary design stage:** choose the best concept.
- **Detailed design stage:** develop engineering descriptions of the concept.
- **Planning stage:** evaluate & alter concept to suit requirements
 - production, distribution, consumption & product retirement.

Potential Problems



- Designer is unable to produce a set of feasible concepts
- Increase in complexity, difficult for one individual to design
- → As complexity increases or the experience of the designer is not sufficient, alternative approaches to the design process must be adopted.

Identifying a Viable Strategy



- Use fundamental design tools: abstraction and separation of concerns.
 - *But how?*
- Inspiration, where inspiration is needed. Predictable techniques elsewhere.
 - *But where is creativity required?*
- Apply own experience or experience of others.

Tools: Abstraction



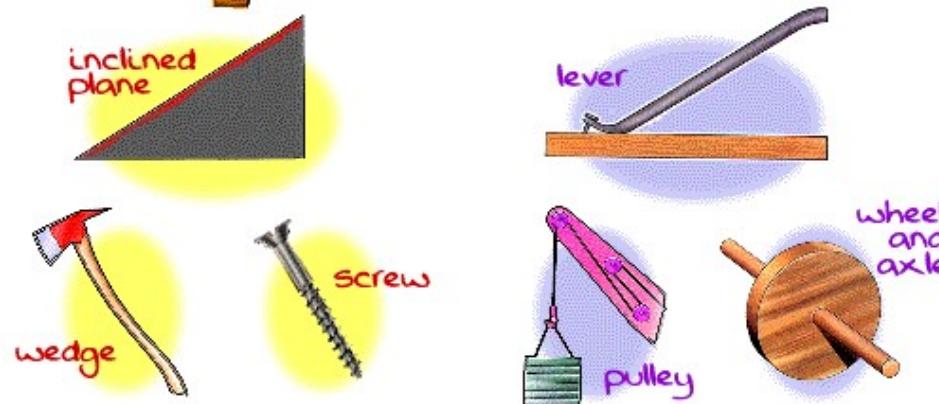
- Abstraction
 - Abstraction(1): look at details, and abstract “up” to concepts
 - Abstraction(2): choose concepts, then add detailed substructure, and move “down”

Tools: Abstraction



- What concepts should be chosen at the outset of a design task?
 - One technique: Search for a “simple machine”

Simple Machines



<http://www.fi.edu/qa97/spotlight3/>

Simple Machines



Domain	Simple Machines
Graphics	Pixel arrays Transformation matrices Widgets Abstract depiction graphs
Word processing	Structured documents Layouts
Process control	Finite state machines
Income Tax Software	Hypertext Spreadsheets Form templates
Web pages	Hypertext Composite documents
Scientific computing	Matrices Mathematical functions
Banking	Spreadsheets Databases Transactions

Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Tools: Separation of Concerns



- Separation of concerns: subdivision of a problem into (hopefully) independent parts.
- Challenge: issues are either actually or apparently intertwined.
- Tradeoffs: total independence of concepts may not be possible.
 - Architect – assess which requirements are highest priority

Tools: Design Recovery

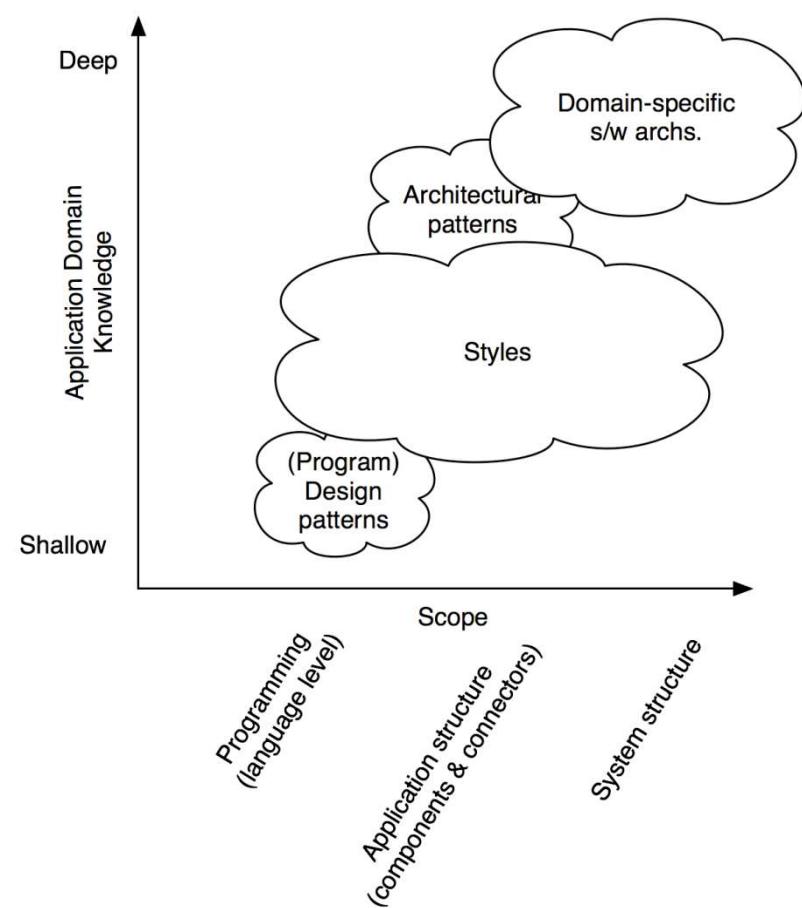


- Discuss this later

The Grand Tool: Refined Experience



- The lessons from prior work include not only the lessons of successes, but also the lessons arising from failure.



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Architectural Patterns/Styles



- “Walk like this!”
- “Why should I walk like that?”
- Canonical structures and rules
 - provide context
 - focus attention
 - promote shared understanding
 - carry hard-won domain knowledge and proven solution techniques

Benefits of Using Patterns/Styles



- Design reuse
 - Well-understood solutions applied to new problems
- Code reuse
 - Shared implementations of invariant aspects of a style
- Understandability of system organization
 - A phrase such as “client-server” conveys a lot of information
- Interoperability
 - Supported by style standardization
- Style-specific analyses
 - Enabled by the constrained design space
- Visualizations
 - Style-specific depictions matching engineers’ mental models

When There's No Experience to Go On...



Other Tools

- Analogy searching
- Brainstorming
- Literature Searching
- Morphological Charts
- Removing Mental Blocks

Control the design strategy

Analogy Searching



- Find analogies to the problem from other fields
- See how the problem is addressed in the other domain
- What domain is often used to find design solutions?

Brainstorming



- Generating a wide set of ideas and thoughts to a design problem
 - without (initially) devoting effort to assessing the feasibility.
- Individual or groups
- Problem?
- Categories of possible designs, instead of specific design solutions
- After brainstorm - may proceed to the Transformation and Convergence steps.

“Literature” Searching



- Find ideas from published materials
 - Digital library collections make searching extraordinarily faster and more effective
 - IEEE Xplore
 - ACM Digital Library
 - Google Scholar
 - Free and open-source software

Morphological Charts



- The essential idea:
 - identify all the primary functions to be performed by the desired system
 - for each function identify a means of performing that function
 - attempt to choose one means for each function such that the collection of means performs all the required functions in a compatible manner.
- Sub-solutions don't need to be compatible with each other

Removing Mental Blocks

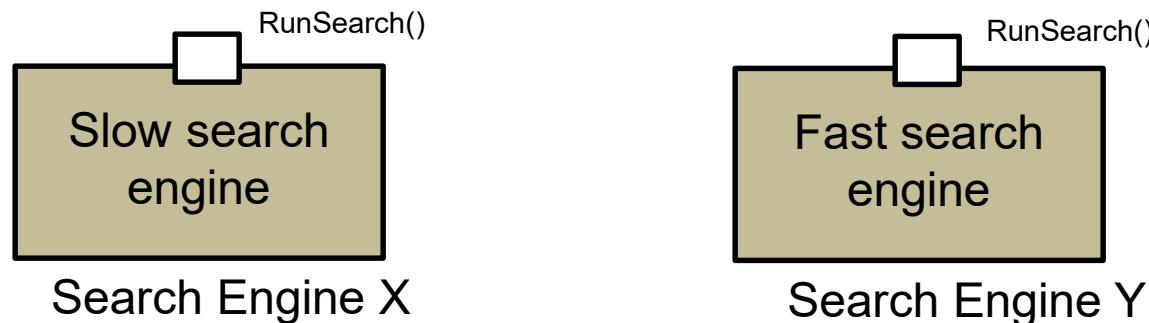


- If you can't solve the problem, change the problem to one you can solve.
 - If the new problem is “close enough” to what is needed, then closure is reached.
 - If it is not close enough, the solution to the revised problem may suggest new venues for attacking the original.

Controlling the Design Strategy



- Identify and review critical decisions. *How?*
 - Determine costs of research and design vs. penalty for taking wrong decisions
- Insulate uncertain decisions. *How?*
 - Continually re-evaluate system “requirements” in light of what the design exploration yields



Part II

TOOL: DESIGN RECOVERY

Tool: Design Recovery



- Design recovery
 - Examine the existing code base
 - Determine the system's components, connectors, and overall topology
- Challenges
 - Time (overhead)
 - Not fully automated
 - Automated software architecture recovery
 - Low accuracy rates
 - Difficult to scale to large code bases (750K or more)

Design Recovery Phases



- Extraction
 - Static analysis of source code
 - Reverse engineering techniques
- Analysis
 - Source code clustered together
 - Clustering techniques
 - Syntactic
 - Semantic

Syntactic Clustering



- Focuses exclusively on the static relationships among code-level entities
 - inter-component (a.k.a. coupling) and intra-component (a.k.a. cohesion) connectivity
- Can be performed without executing the system
- May ignore or misinterpret many subtle relationships, because dynamic information is missing

Sample tool:

<https://www.hello2morrow.com/products/sonargraph/architect9>

Semantic Clustering



- Includes all aspects of a system's domain knowledge and information about the behavioral similarity of its entities.
- Requires interpreting the system entities' meaning, and possibly executing the system on a representative set of inputs.
- Difficult to automate

Design Recovery: Case Study



- Apache HTTP Server
 - v1.3.17 ~ 100K lines of C code
 - 1 semester long
- Steps:
 - Determine purpose of analysis
 - What are the key concepts of the system?
 - Gather domain knowledge and understand the system
 - Find sources of information about Apache
 - Project website: Usage and administration of Apache (project website)
 - Info about implementation (“Writing Apache Modules with Perl and C”)

Design Recovery: Case Study



- Steps (cont'd)
 - Learn the functions of the software
 - Install, configure, administer the product
 - Study the source code
 - Used a tool to create hyperlinked HTML files
 - Classify source code and decide which ones contain important information to understand the running system
- Details
 - <http://www.fmc-modeling.org/projects/apache>

Application to OSS Project



- Use the steps from the Apache HTTP Server recovery project for recovering the design of your OSS project



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



ARCHITECTURE PATTERNS

Portions taken and adapted from Richard N. Taylor (UCI)



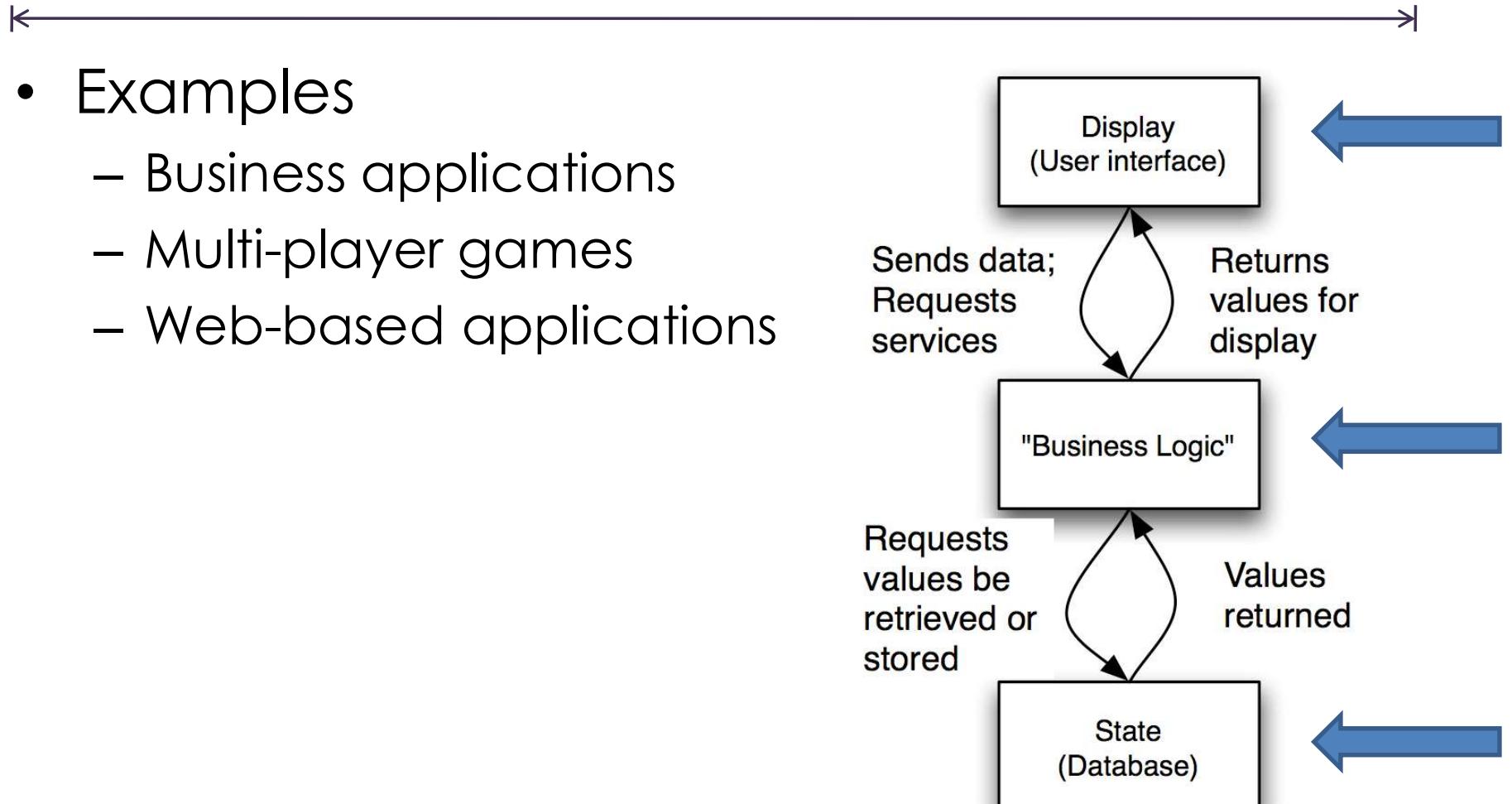
Architectural Patterns



- State-Logic-Display
- Model-View-Controller (MVC)
- Sense-Compute-Control



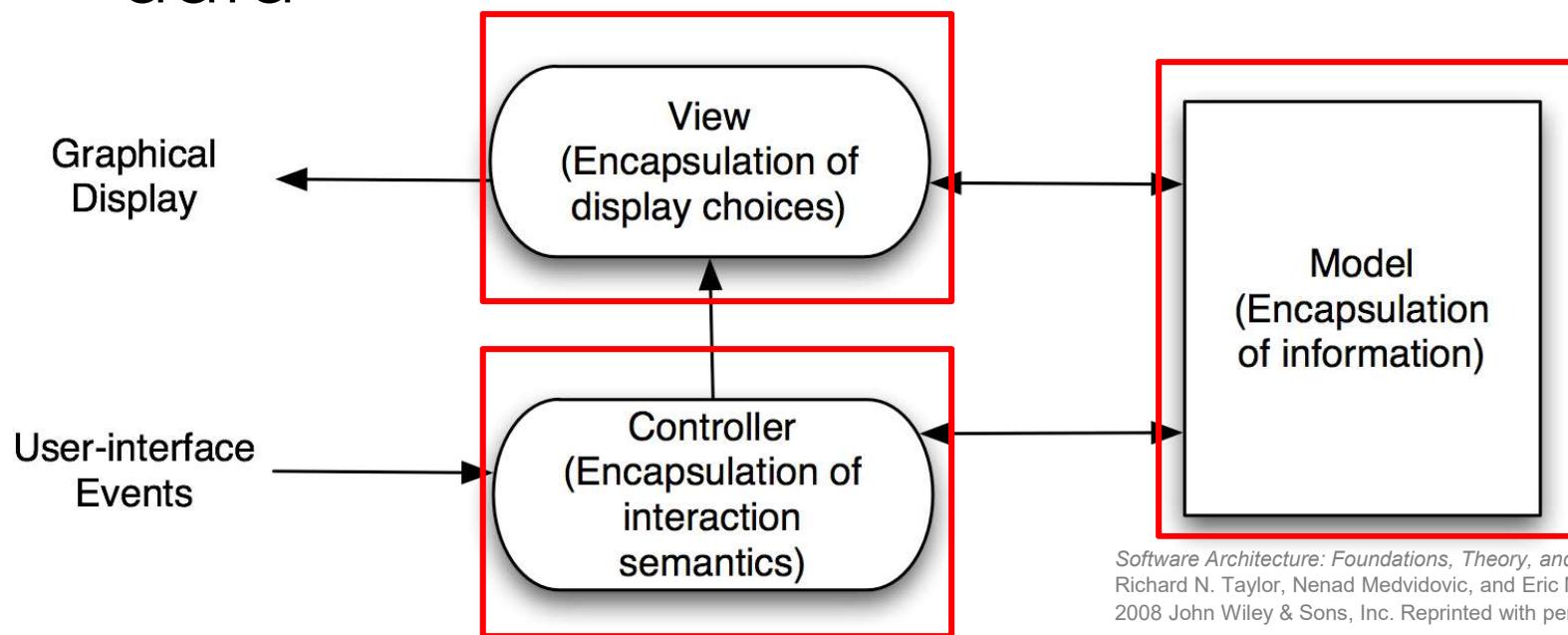
State-Logic-Display: Three-Tiered Pattern



Model-View-Controller (MVC)



- Objective is to promote separation
 - Can have multiple views on the same data



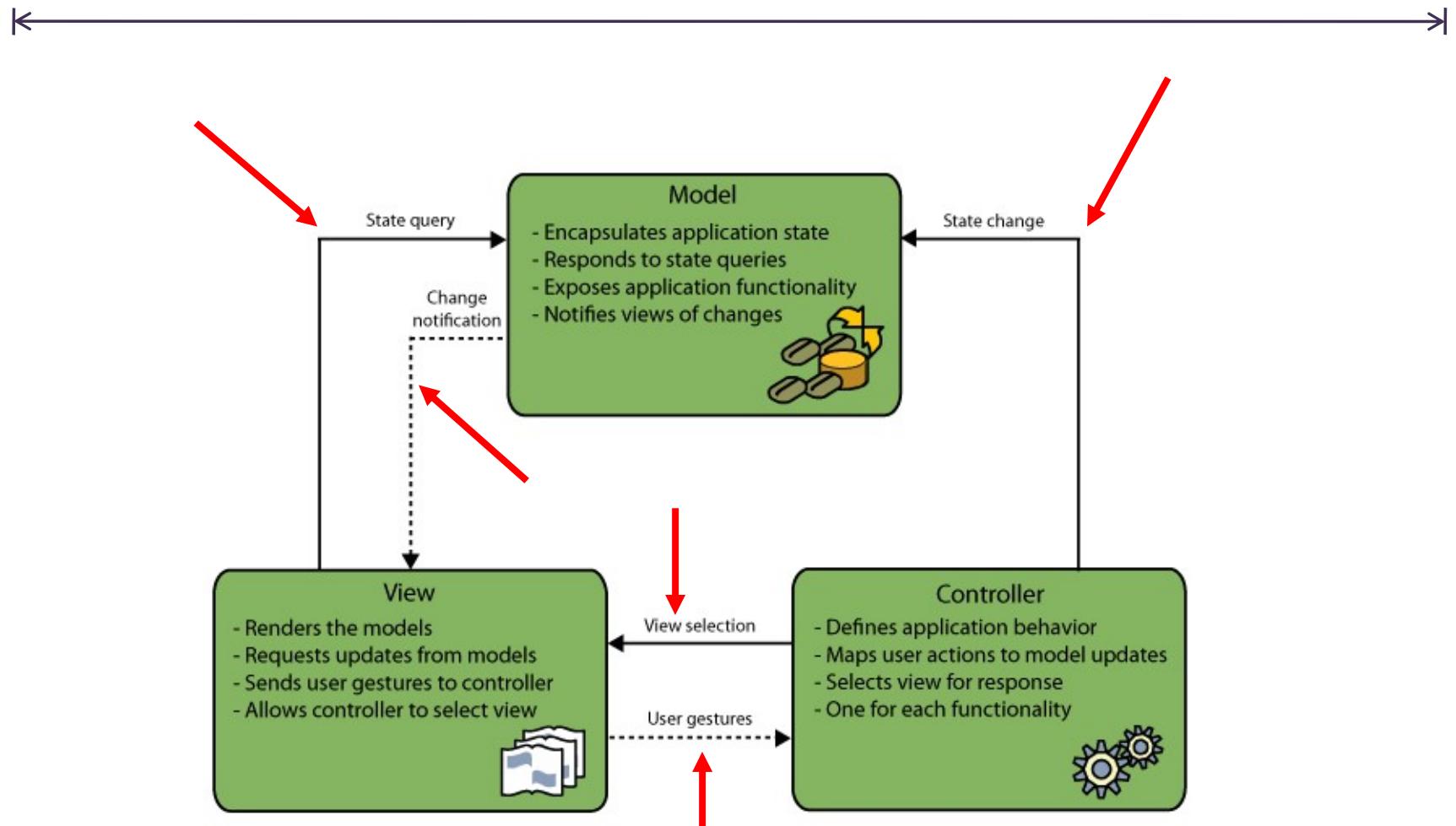
MVC Variants



CSS 553, Spring 2023, 3a Arch Pattern



Variant MVC Design

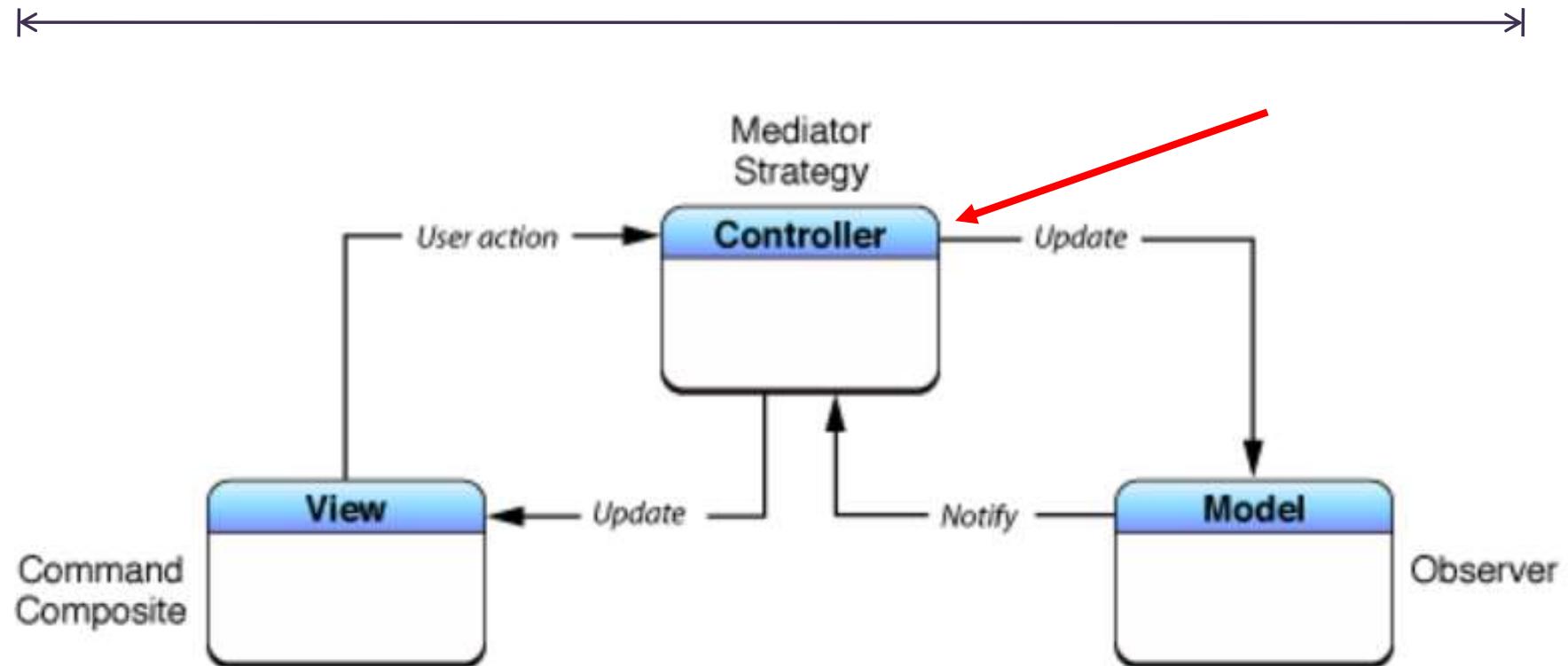


<http://netbeans.org/kb/docs/javaee/ecommerce/design.html#architecture>

CSS 553, Spring 2023, 3a Arch Pattern



Variant MVC Design



<https://developer.apple.com/library/archive/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html>

CSS 553, Spring 2023, 3a Arch Pattern



Model-View-Controller



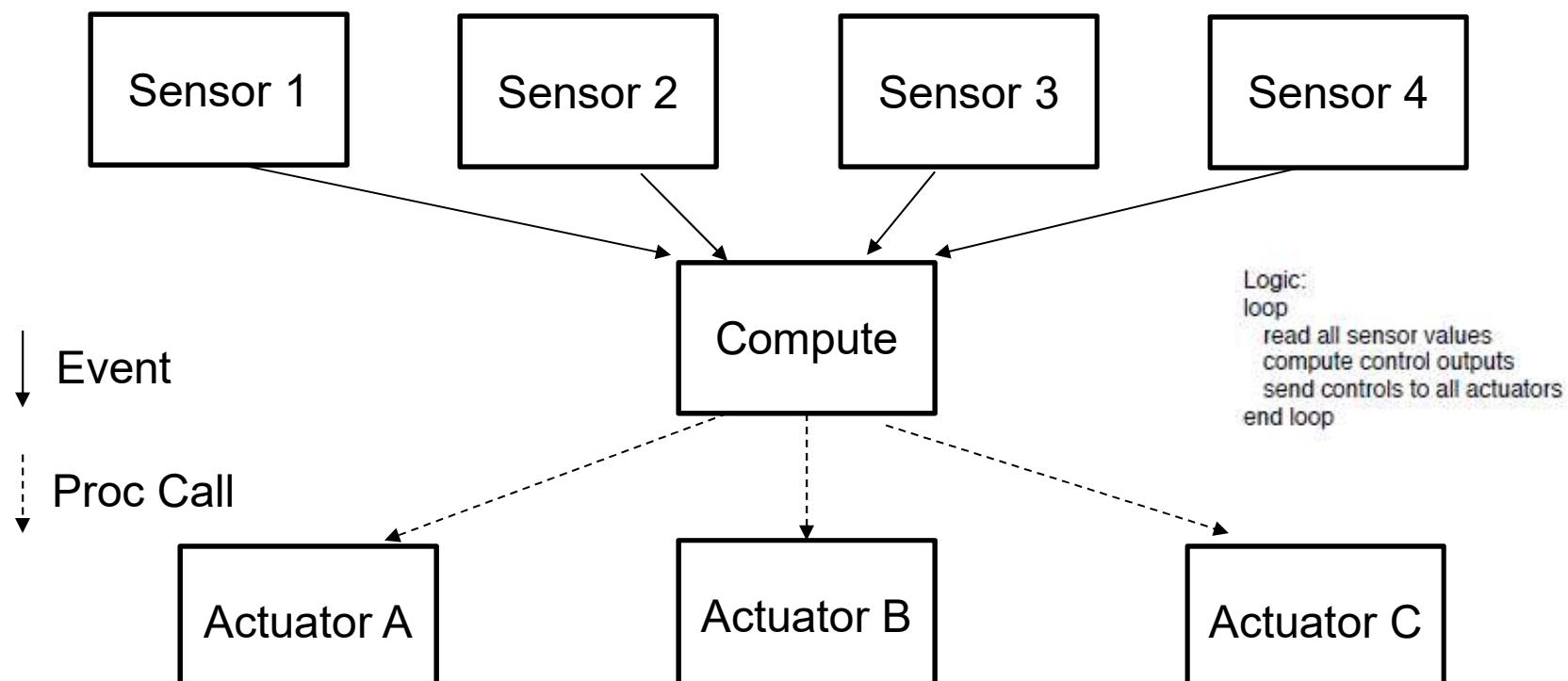
- Typical uses
 - Graphical user interfaces
 - Multiple ways to view or manipulate the data
- Caveat
 - If interactions between model and controller are simple, may have more overhead



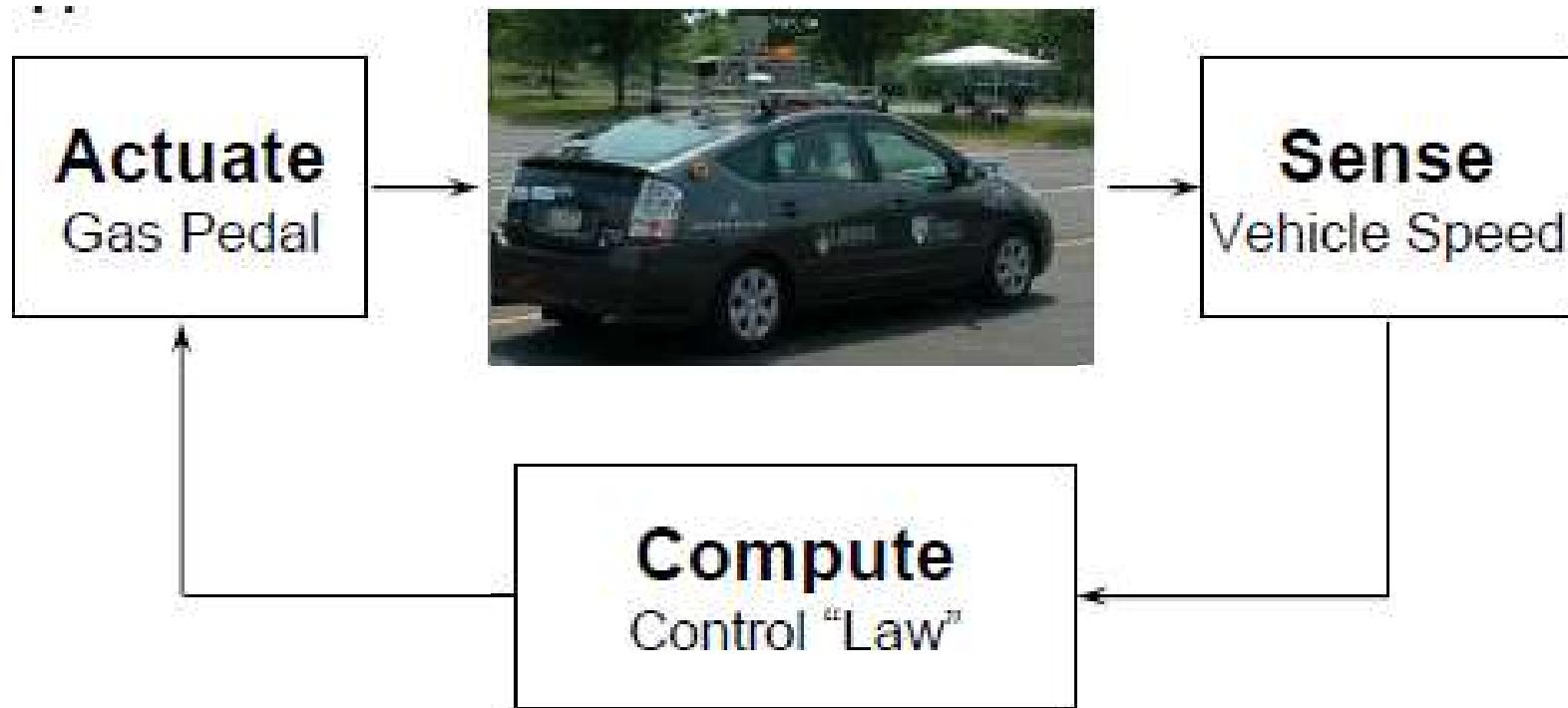
Sense-Compute-Control



- Also known as Stimulus/Response System
- Reacts to stimuli



Sense-Compute-Control



http://robotics.mem.drexel.edu/mhsieh/Courses/E58_S08/Notes/controlTheory_22.pdf



Sense-Compute-Control



- Typical uses
 - Real-time systems, embedded control applications
 - Systems that directly interact with the environment through sensors and actuators
- Caveat
 - Highly subject to environmental conditions
 - Hardware malfunction (e.g. in sensors) may not be detected and system can enter into an error state





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



ARCHITECTURE STYLES

Portions taken and adapted from Richard N. Taylor (UCI)

Common Architectural Styles



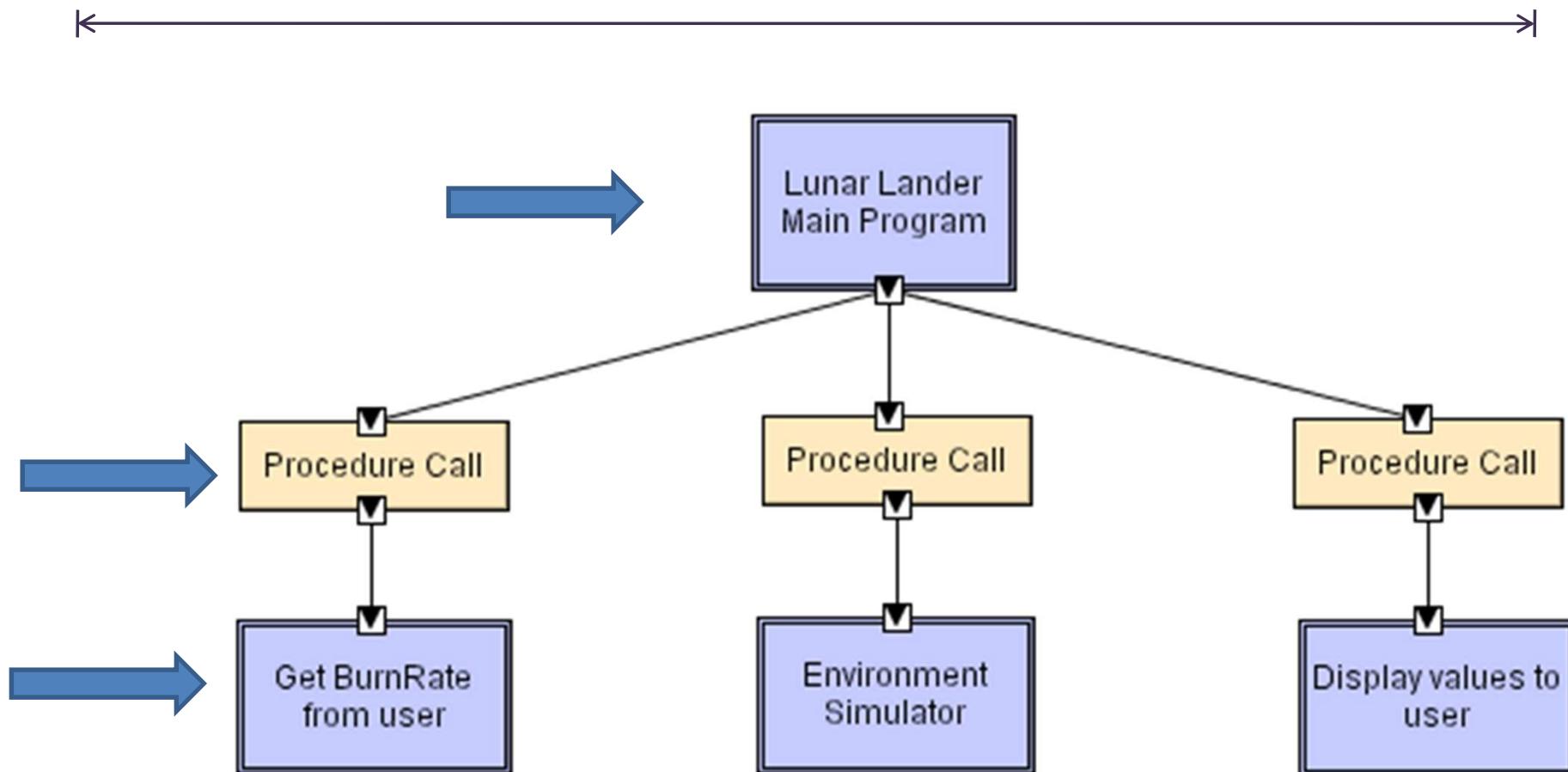
- Main program and subroutines
- Object-oriented
- Virtual Machine
- Client-Server
- Batch Sequential
- Pipe and Filter
- Repository / Blackboard
- Rule-based
- Interpreter
- Mobile Code

The Lunar Lander: A Long-Running Example



- A simple computer game that first appeared in the 1960's:
<http://www.youtube.com/watch?v=CnKzeHPgWy8>
- Simple concept:
 - You (the pilot) control the descent rate of the Apollo-era Lunar Lander
 - Throttle setting controls descent engine
 - Limited fuel
 - Initial altitude and speed preset
 - If you land with a descent rate of < 5 fps: you win (whether there's fuel left or not)
 - “Advanced” version: joystick controls attitude & horizontal motion

Main Program and Subroutines



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Main Program and Subroutines



```
public void main {  
    int burnRate = getBurnRate();  
    int altitude = simulateDescent(burnRate);  
    display(altitude);  
}
```

```
public int getBurnRate {  
    :  
}
```

```
public int simulateDescent(int br) {  
    :  
}
```

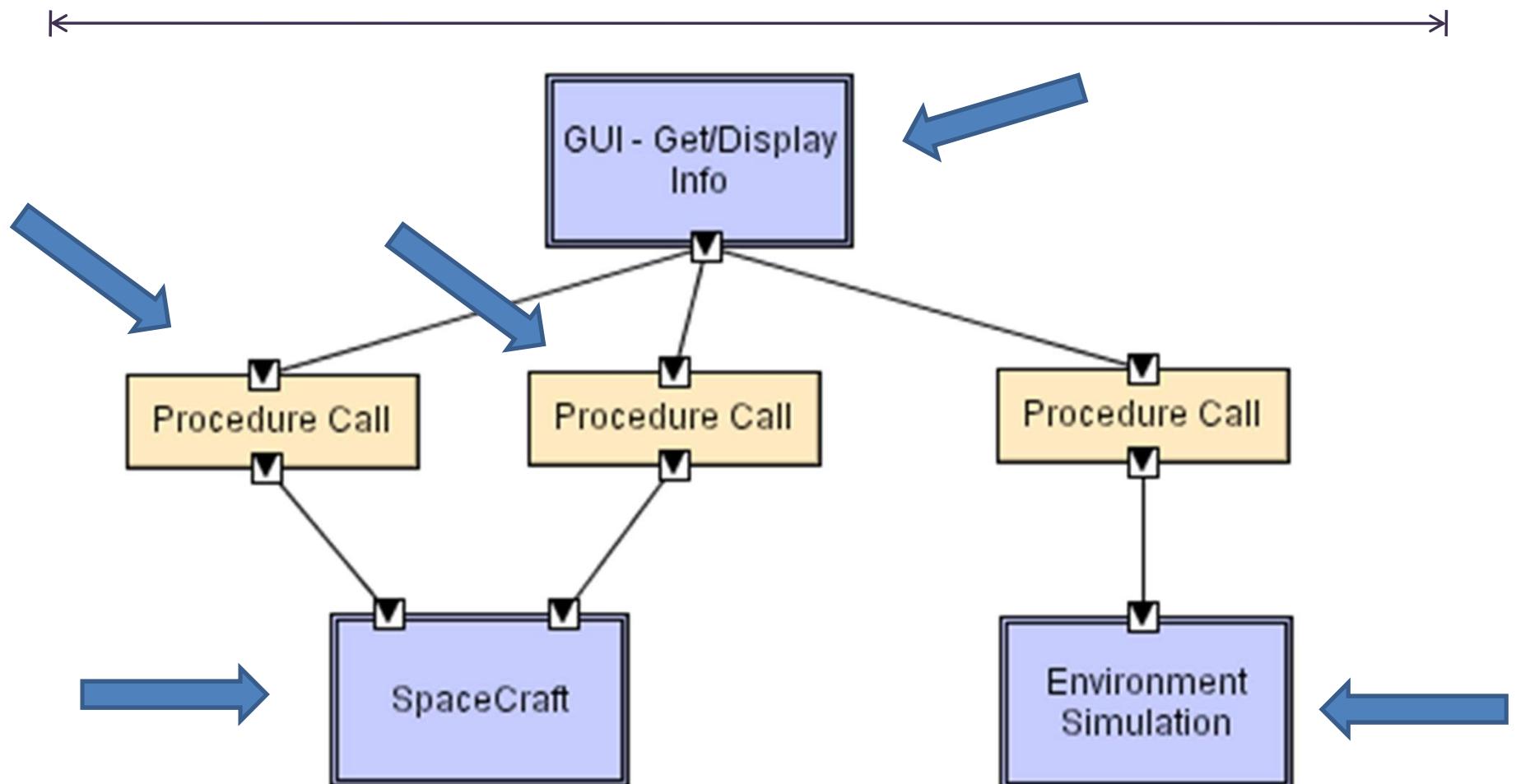


Main Program and Subroutines



- Typical uses
 - Small programs
- Caveat
 - Large applications – will not scale

Object-oriented



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Object-oriented

```
public void GUI {  
    Spacecraft s = new Spacecraft(altitude, fuel, time,  
                                velocity, mass);  
  
    int altitude = s.getAltitude();  
    :  
}  
  
public class Spacecraft {  
    public Spacecraft(int altitude, int fuel, int time, int  
                      velocity, int mass) {  
        :  
    }  
}
```



Object-Oriented (OO) Style



- Components are objects
 - Data and associated operations
- Connectors are messages and method invocations
- Style invariants
 - Objects are responsible for their internal representation integrity
 - Internal representation is hidden from other objects
- Advantages
 - “Infinite malleability” of object internals
 - System decomposition into sets of interacting agents
- Disadvantages
 - Objects must know identities of servers
 - Side effects in object method invocations

Object-oriented



- Typical uses
 - Applications where developers want a close correspondence to the real world
 - Dynamic data structures
- Caveat
 - If use in a distributed setting, will require extensive middleware
 - Relatively inefficient for high performance applications

Middleware



- Supported by extensions to operating systems
 - OMG CORBA/DCE
 - Microsoft COM/ActiveX/.Net
 - JavaBeans
 - Internet-related SOAP / WSDL / UDDI



Layered Style



- Hierarchical system organization
 - “Multi-level client-server”
 - Each layer exposes an interface (API) to be used by above layers
- Each layer acts as a
 - Server: service provider to layers “above”
 - Client: service consumer of layer(s) “below”
- Connectors are protocols of layer interaction
- Example: operating systems
- *Virtual machine* style results from fully opaque layers

Layered Style (cont'd)

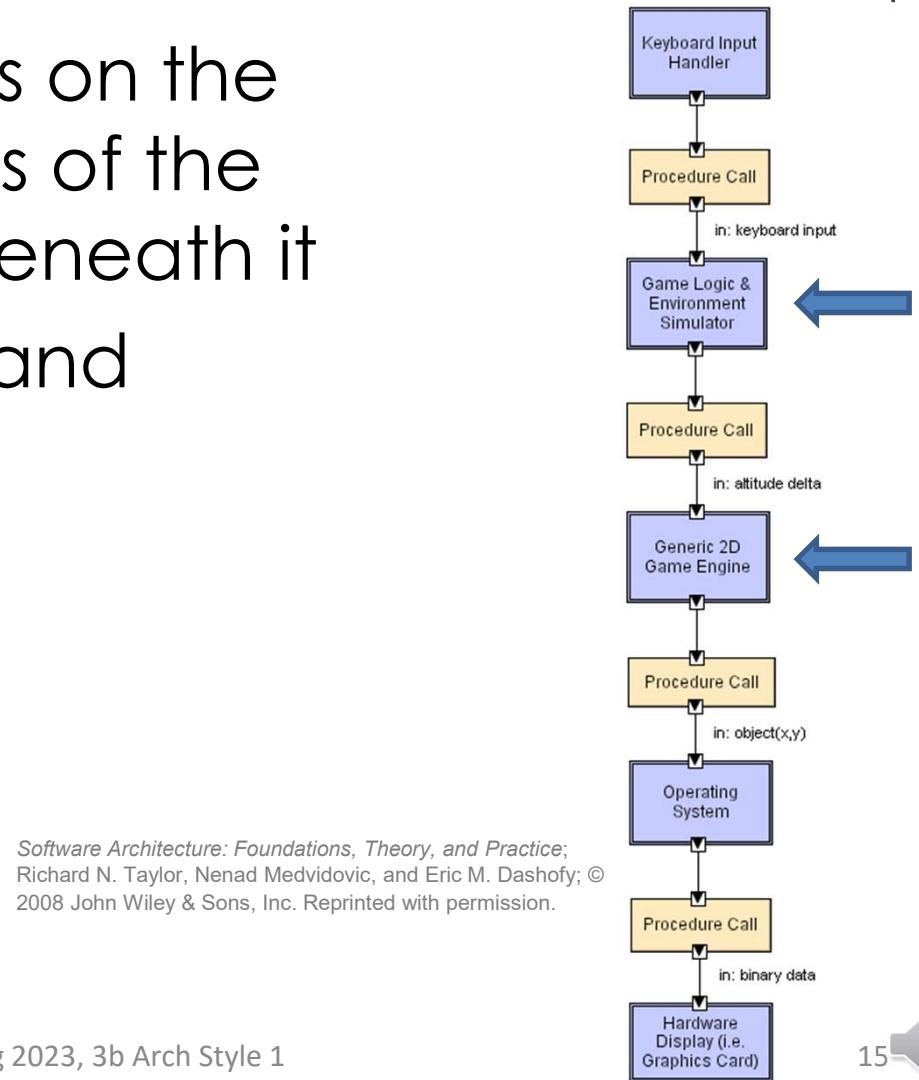


- Advantages
 - Increasing abstraction levels
 - Evolvability & reusability
 - Changes in a layer affect at most the adjacent two layers
 - Different implementations of layer are allowed as long as interface is preserved
 - Standardized layer interfaces for libraries and frameworks

Layered: Virtual Machine



- Each layer only relies on the facilities and services of the layer immediately beneath it
- Support separation and independence



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Layered: Virtual Machine



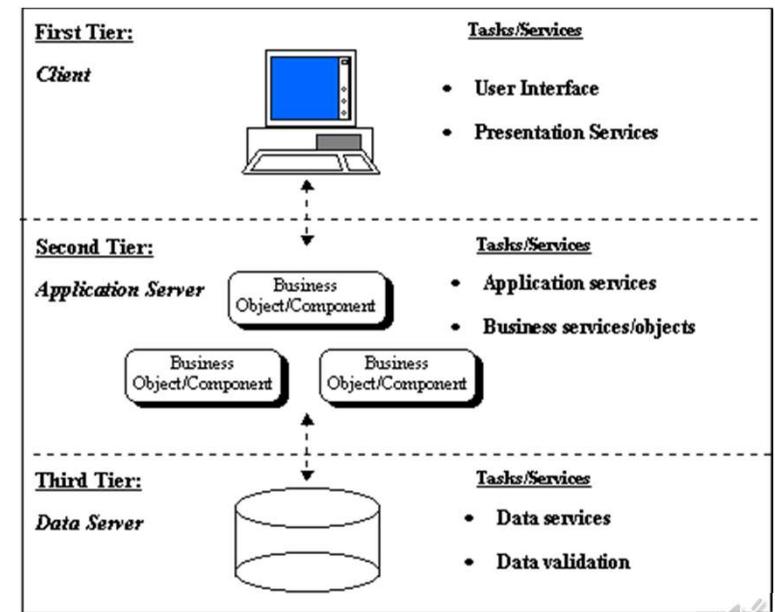
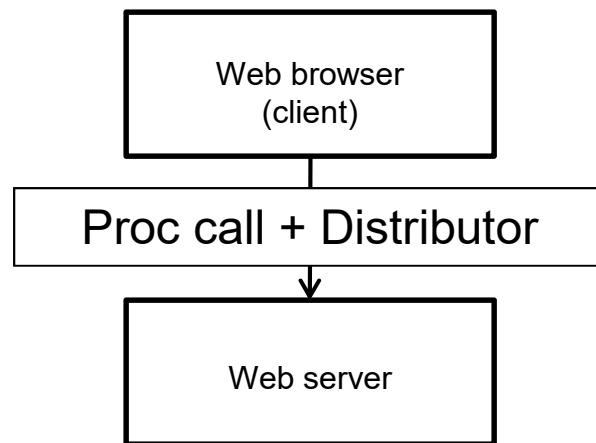
- Typical uses
 - Operating systems, network protocol
- Caveats
 - Strict virtual machines can be inefficient

Layered: Client-Server Style



- Variation of Layered
 - 2-tiered client-server
 - 3-tiered client-server
- Clients use services provided by the servers

Fat vs thin client?



Layered: Client-Server Style



- Components are clients and servers
- Servers do not know number or identities of clients
- Clients know server's identity
- Connectors are RPC-based network interaction protocols

Layered: Client-Server Style



- Typical uses
 - Where centralized data or centralized processing is needed
 - Usually business applications
- Caveat
 - Will not work well when you have a limited bandwidth and there are many client connections

Data-Flow Styles

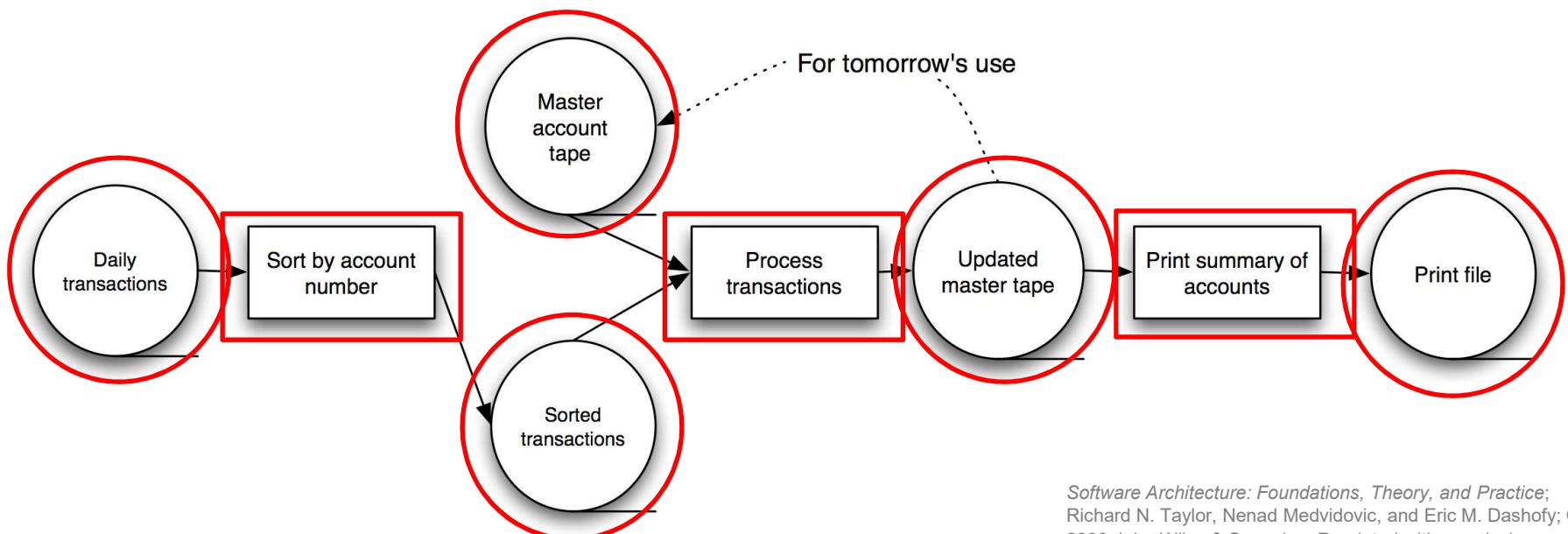


- Batch-Sequential
- Pipe and Filter

Batch-Sequential: A Financial Application

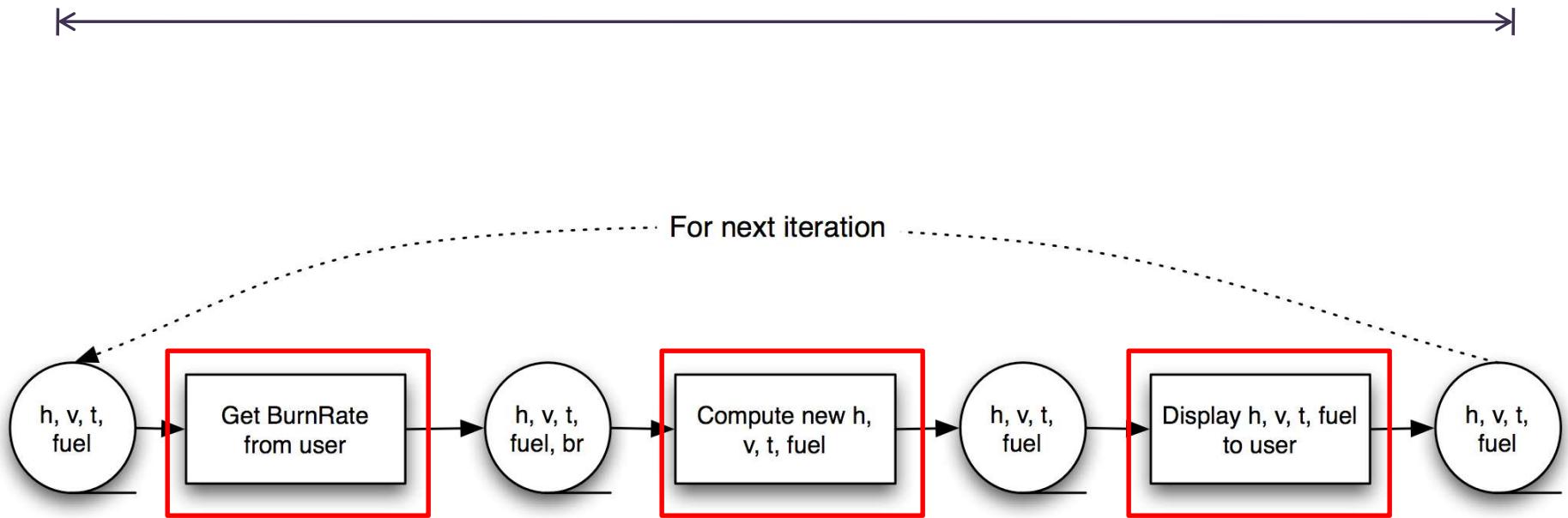


- Separate programs are executed in order; data is passed as an aggregate from one program to the next.



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Batch-Sequential LL



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Not a recipe for a successful lunar mission!

Batch Sequential

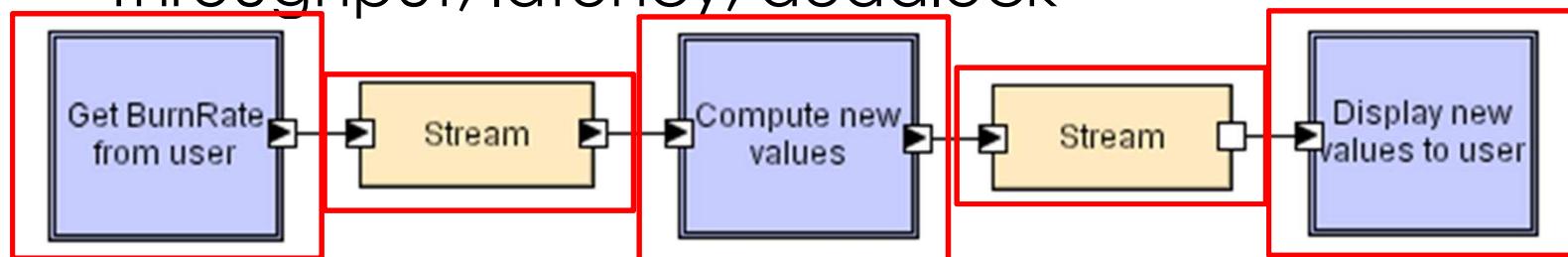


- Typical uses:
 - Transaction processing in financial systems. “The Granddaddy of Styles”
- Caveat
 - Not work well when interaction between components is required; when concurrency between components is possible or required

Pipe and Filter Style



- System behavior is a succession of component behaviors
- Filter addition, replacement, or reuse
 - Possible to hook any two filters together
- Concurrent execution
- Certain analyses
 - Throughput, latency, deadlock



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Pipe and Filter Style



- Components are filters
 - Transform input data streams into output data streams
 - Possibly incremental production of output
- Connectors are pipes
 - Conduits for data streams
- Style invariants
 - Filters are independent (no shared state)
 - Filter has no knowledge of up- or down-stream filters

Pipe and Filter (cont'd)



- Variations
 - Pipelines — linear sequences of filters
 - Bounded pipes — limited amount of data on a pipe
 - Typed pipes —strongly typed data

Pipe and Filter (cont'd)



- Typical uses
 - UNIX shell signal processing
 - Distributed systems parallel programming
 - Example: `ls invoices | grep -e August | sort`
- Caveats
 - Batch organization of processing
 - Interactive applications
 - Lowest common denominator on data transmission



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



ARCHITECTURE STYLES – PART II

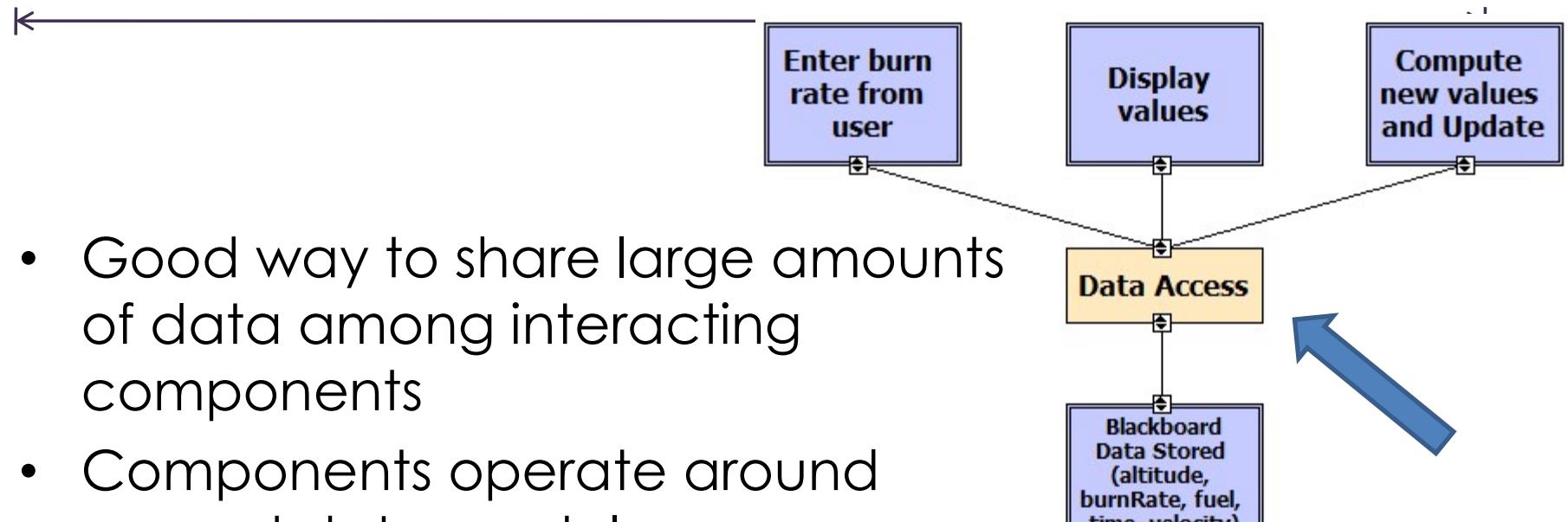
Portions taken and adapted from Richard N. Taylor (UCI)

Common Architectural Styles



- Main program and subroutines
- Object-oriented
- Virtual Machine
- Client-Server
- Batch Sequential
- Pipe and Filter
- Repository / Blackboard
- Rule-based
- Interpreter
- Mobile Code

Repository / Blackboard



- Good way to share large amounts of data among interacting components
- Components operate around agreed data model
- System control is entirely driven by the blackboard state

implicit invocation

Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Repository / Blackboard

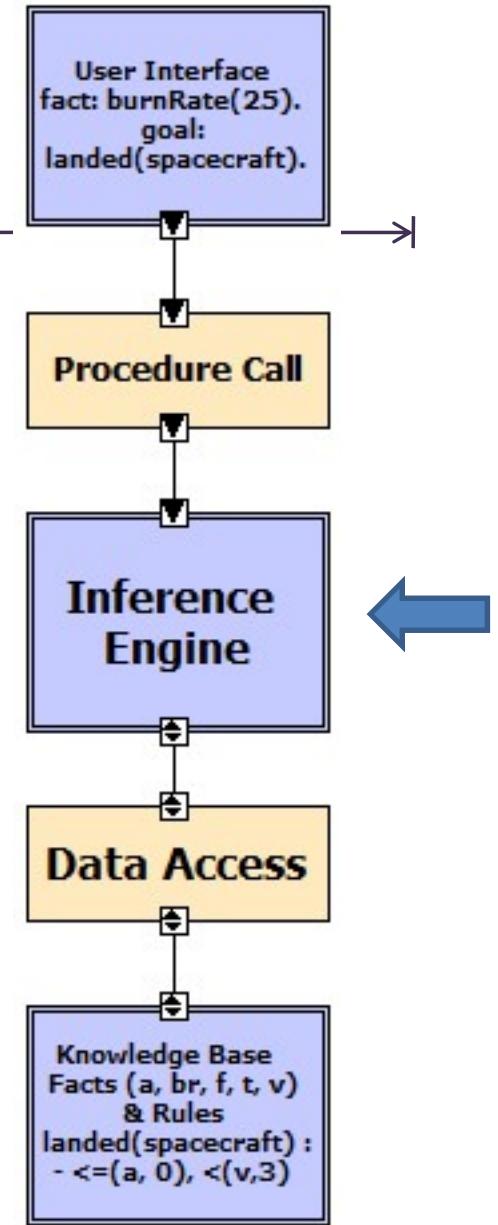


- Typical uses
 - AI systems
 - Integrated software environments
 - Compiler architecture
- Caveat
 - If the data model changes, have to reflect the change to all the components

Rule-Based Style



- Behavior of an application can be very easily modified through addition or deletion of rules from the knowledge base.
- Components?
- Connectors?
- Data elements?



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy ©
2008 John Wiley & Sons, Inc. Reprinted with permission



Rule-Based Style



- Typical use:
 - When a problem can be understood as a matter of repeatedly resolving a set of predicates
 - E.g., Semantic Web: Apache Jena

ab:craig	ab:homeTel	"(194) 966-1505" .	<i>Facts stored in knowledge base</i>
ab:craig	ab:email	"craigellis@yahoo.com" .	
ab:craig	ab:email	"c.ellis@usairwaysgroup.com" .	

subject predicate object

```
SELECT ?craigEmail  
WHERE  
{ ab:craig ab:email ?craigEmail . }
```

Query

Source:
<http://www.learningsparql.com/examples/index.html>

Rule-Based Style

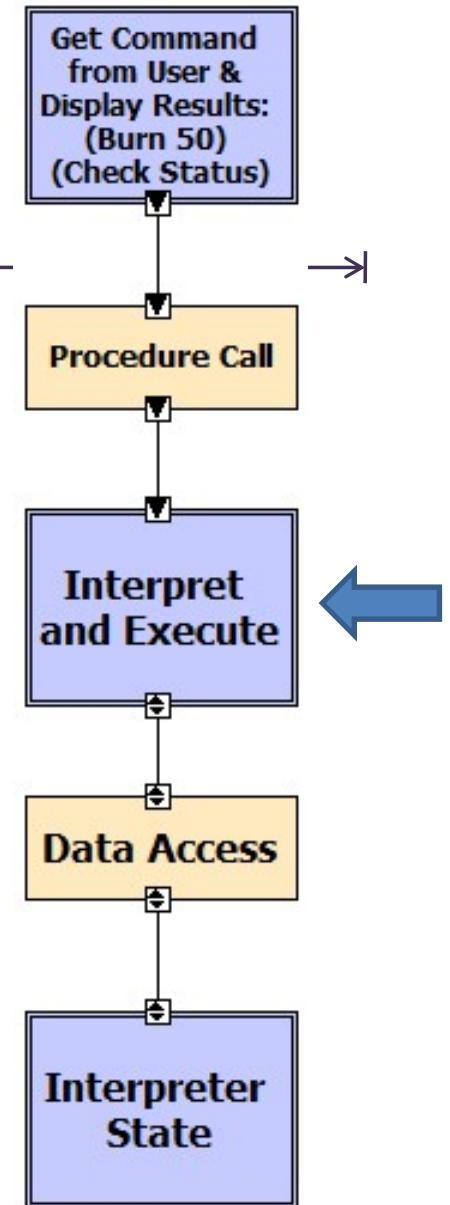


- Caveat:
 - When a large number of rules are involved understanding the interactions between multiple rules affected by the same facts can become very difficult.

Interpreter Style



- Interpreter parses and executes input commands, updating the state maintained by the interpreter
- Highly dynamic behavior possible, where the set of commands is dynamically modified. System architecture may remain constant while new capabilities are created based upon existing primitives.
- Components?
- Connectors?



Interpreter Style

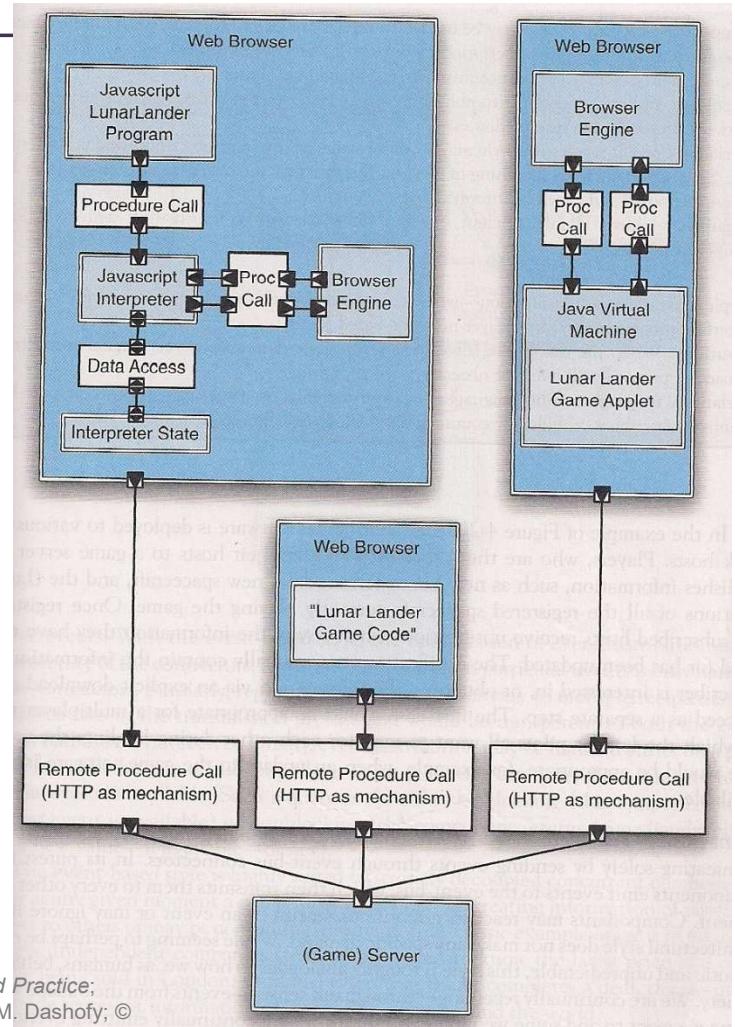


- Typical use
 - End-user programmability; supports dynamically changing set of capabilities
 - Lisp, Scheme, Excel functions
 - Example, see <https://repl.it/languages/scheme>
- Caveat
 - Interpreted code is slower than executable code
 - Memory management may be an issue, especially when multiple interpreters are invoked simultaneously

Mobile Code



- A data element (some representation of a program) is dynamically transformed into a data processing component.
- Components?
- Connectors?
- Data elements?



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Mobile-Code Style



- Typical uses
 - Process large data sets in distributed locations, more efficient to move the processing
 - Scripting languages (i.e. JavaScript, VBScript), ActiveX control, Grid computing
- Caveat
 - Security issues
 - when need to tightly control versions of deployed software
 - when network connections are unreliable
 - when cost of transmission exceeds the cost of computation
- Variants: Code-on-demand, remote evaluation, and mobile agent.

Mobile Code variants compared to Client-Server

Paradigm	Before	After
	S_A	S_B
<i>Client-Server</i>	A	know-how resources B
<i>Remote Evaluation</i>	know-how A	resources B
<i>Code on Demand</i>	resources A	know-how B
<i>Mobile Agent</i>	know-how A	resources

← → ← → ← →

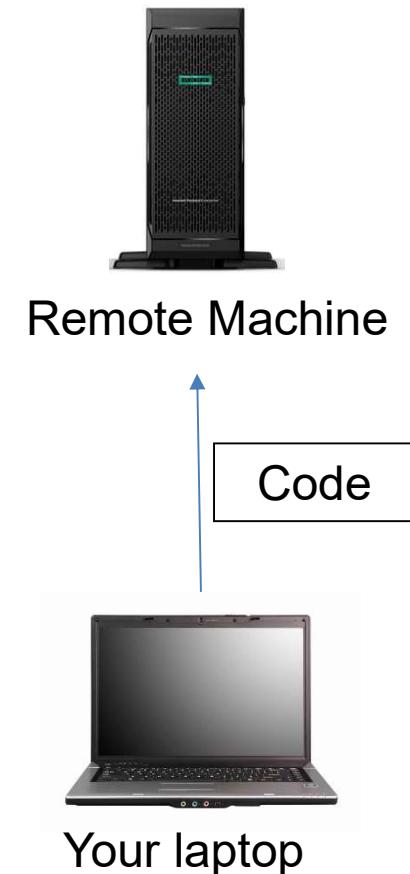
The table compares five mobile code paradigms (Client-Server, Remote Evaluation, Code on Demand, Mobile Agent) against a Client-Server paradigm. The 'Before' column shows the state before code execution, and the 'After' column shows the state after code execution. Red boxes highlight specific changes or differences in the 'After' state for each paradigm.

Source: Carzaniga, A., Picco, G.P., Vigna, G.,
Designing Distributed Applications with Mobile Code
Paradigms, ICSE 1997.

Mobile Code

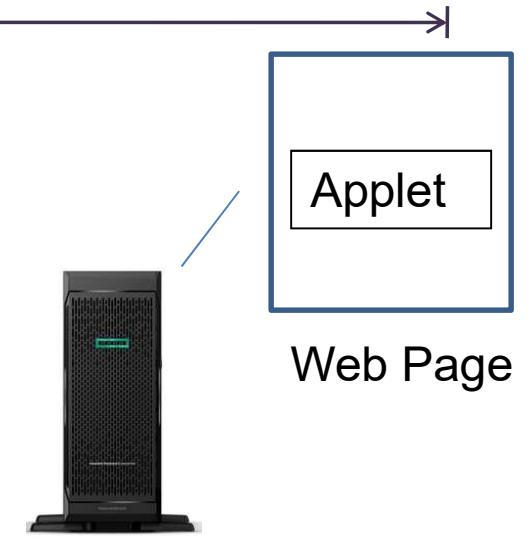


- Remote Evaluation
- Examples
 - Word processor & PostScript printer
 - code: postscript file sent to the printer
 - code executed by the PostScript interpreter hosted on the printer



Mobile Code

- Code on Demand
- Example: Java Applets
 - Resource to be rendered on browser, but client machine does not have software to display resource → browser downloads the software
 - Java program embedded in a web page & runs within a browser
 - <https://docs.oracle.com/javase/tutorial/deployment/applet/index.html>



Web Server



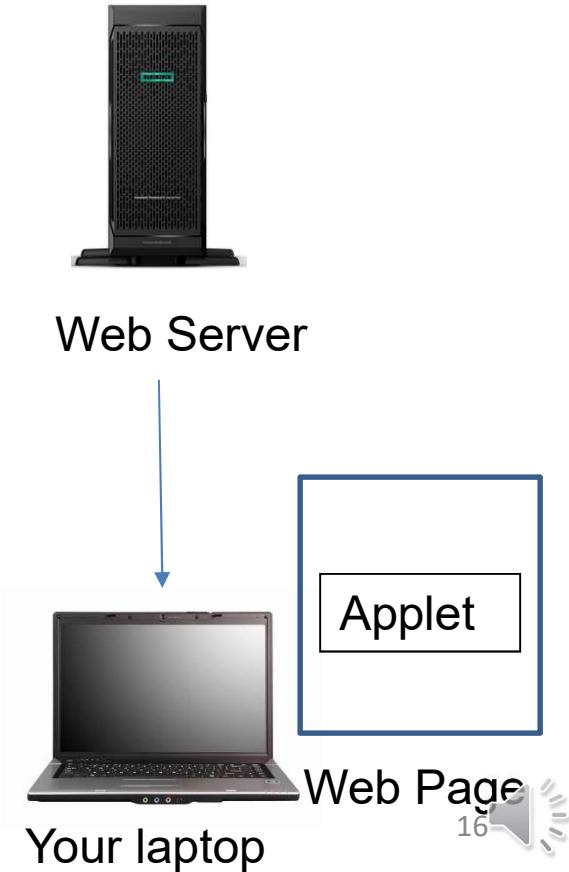
Your laptop



Code on Demand



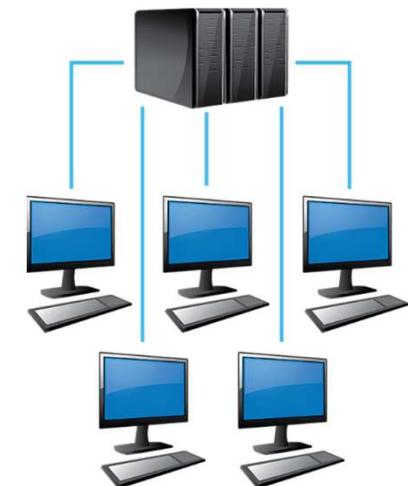
- Code on Demand
- Example: Java Applets
 - Resource to be rendered on browser, but client machine does not have software to display resource → browser downloads the software
 - Java program embedded in a web page & runs within a browser
 - <https://docs.oracle.com/javase/tutorial/deployment/applet/index.html>



Mobile Code



- Mobile Agent
 - The whole computational component together with its state, the code it needs, and some resources are moved to a remote site
 - E.g. Network diagnostic tool that travels along all faulty nodes



Questions:



- What is the difference between explicit and implicit invocation? Provide at least one architecture style that uses each type of invocation.
- When would you use a 2-tiered Client-Server vs a 3-tiered Client-Server? Give specific scenarios for each.
- When would you use a fat-client vs thin-client? Give specific scenarios for each.
- When would you use Mobile Code? Give specific scenarios for each.
- Is Repository the same as Client-Server? Why or why not?
- Which architecture styles support distributed systems?
- Is it easy to transform software built for standalone machine into a distributed setting? Why or why not?



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Outline



- Introduction to the CSS553
- Introduction to Software Architecture
- Course project



Part I

INTRODUCTION TO CSS553

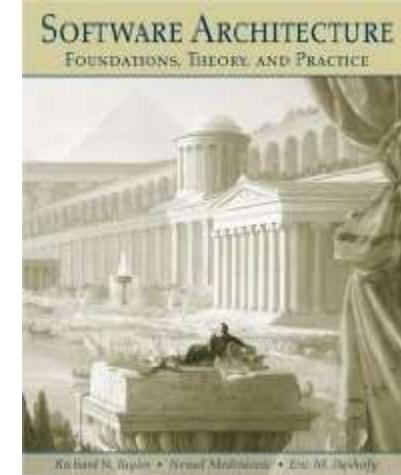
Syllabus



Class attendance and the book



- Topics covered in class are not necessarily in the book
 - Other reading materials assigned
- What is covered in class takes precedence over the book
- Come to class prepared to discuss what you have read



Course objectives



- Name and describe software architecture concepts, methods, and best practices
- Select and justify the use of software architecture styles or patterns for a given situation
- Apply software architecture styles or patterns on a small but realistic project
- Use CASE tools in creating, visualizing, and analyzing software architecture artifacts



Student survey



- Available online – Complete by tomorrow

Course Structure



- Refer to the Schedule
- “Hands-on” class
 - Combination of lecture, class discussions, course project, and two exams
 - Subset of lecture notes available before class
 - Links for in-class activities on Canvas

Writeups



- Post any writeup in recorded lecture in “Week x & y Write-up” Discussion board
 - Note: there’s a typo in the writeup submission for this week (Building Analogy)
- Only need one write-up submission every 2 weeks

Group Meetings



- Meet with your team at least 2 hours / week
- Schedule your meeting to coincide with office hours
- Poll: grader office hours
 - Take the poll later, after form groups
 - Use the “Raise Hand” feature in Zoom
 - <https://support.zoom.us/hc/en-us/articles/205566129-Raising-your-hand-in-a-webinar>

Advice from past students



- This is definitely a useful class. Be prepared before coming to class and ask questions if you have any.
- I will suggest them to check on the project document requirement carefully. It is easy to miss minor requirements that matters.

Advice from past students



- Prepare for the quizzes [exams] as much as you can, include all the detail as possible. Study more examples.
- Start the homework assignments early - they take quite a bit of time.

Advice from past students



- Please rehearse your presentation so that you can finish within the allocated time frame ...If you find yourself talking [more time], you should rework your slides to be more concise.

Questions?



Part II

COURSE PROJECT

Project List



- Refer to the website for possible projects

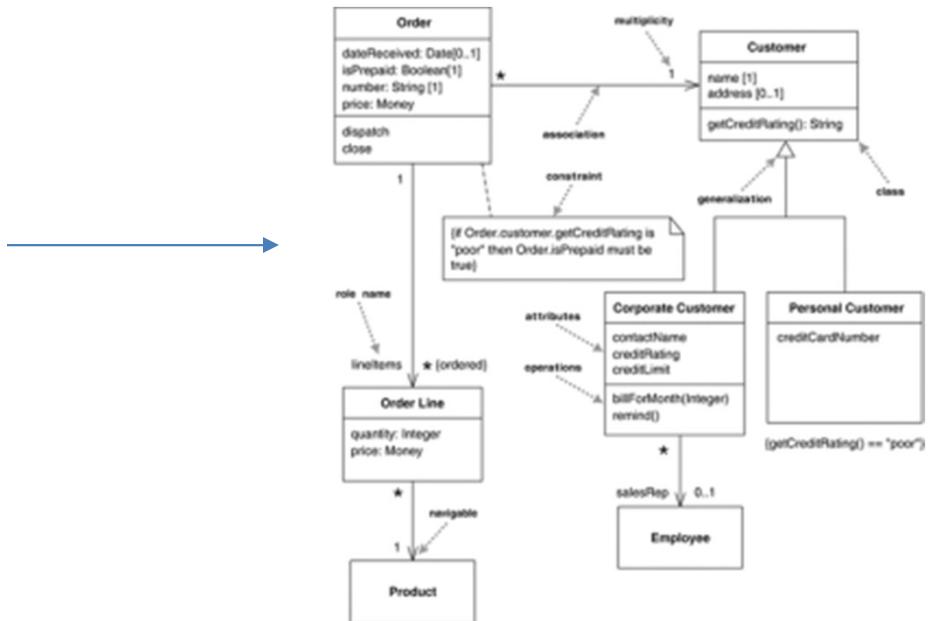
Reverse Engineering tools



- Tools automatically transform source code to a class diagram
 - E.g., Visual Paradigm, StarUML



Source Code



Class Diagram

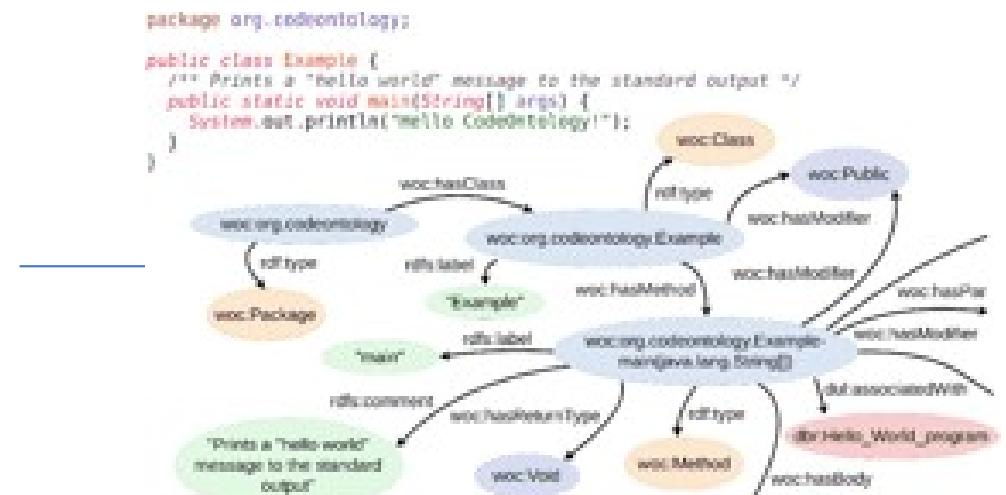
How to automatically find a design pattern?



- Transform code into a graph representation



Source Code



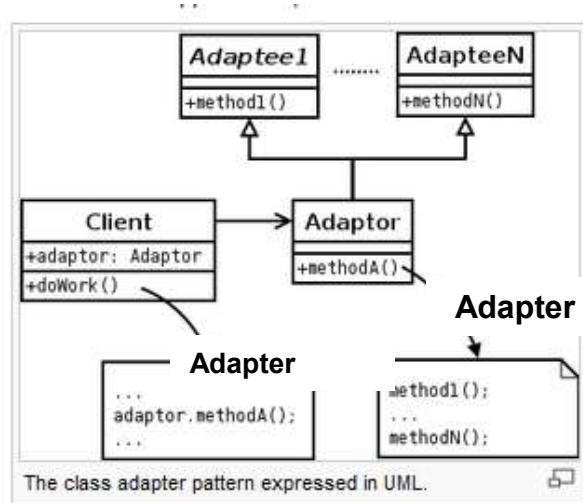
Atzeni, Mattia and Maurizio Atzori. "CodeOntology: RDF-ization of Source Code." SEMWEB (2017).

Tool: CodeOntology (.nt file)
RDF Graph
Semantic Web Technologies

How to automatically find a design pattern?



- Represent the design pattern as a query
 - Transform UML Class diagram into SPARQL
 - PatternScout – Research tool



Design Pattern

```
1 SELECT ?ClassA ?OperationA  
2 WHERE {  
3 ?ClassA a woc:Class .  
4 ?OperationA a woc:Method .
```

SPARQL query

http://en.wikipedia.org/wiki/Adapter_pattern

How to automatically find a design pattern?



- Run a query against the graph

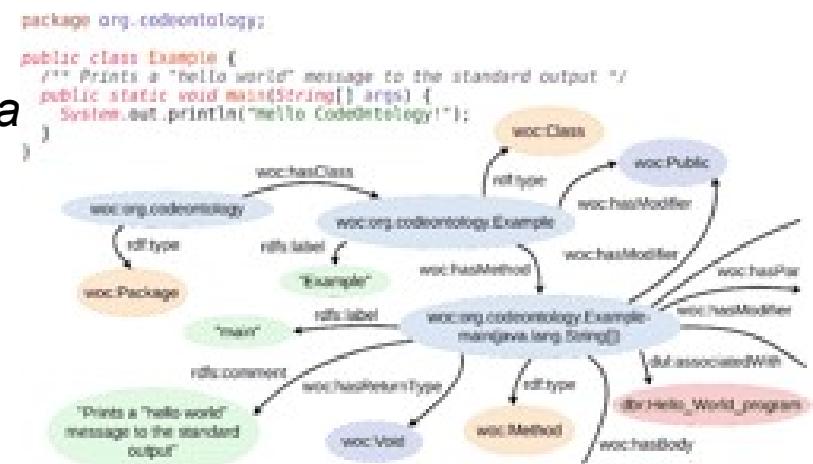
```
1 SELECT ?ClassA ?OperationA  
2 WHERE {  
3   ?ClassA a woc:Class .  
4   ?OperationA a woc:Method .
```

SPARQL query (.rq file)

Apache Jena



List of matches
Query results



Atzeni, Mattia and Maurizio Atzori. "CodeOntology: RDF-ization of Source Code." SEMWEB (2017).



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Part I

REVIEW

Why is software engineering difficult?



- What are the essential difficulties?
 - Complexity – each part of the sw is unique
 - Conformity – conform to different requirements (e.g. HIPAA) & regulations
 - Changeability – sw is easy to change and can affect performance of entire system
 - Invisibility – sw can't be represented as a physical object

Requirements



- Two types?
 - Non-functional – properties (correctness, reliability, -ilities) – determined by the design of system
 - Functional – features (or functions), behavior
- How satisfy each type? How would you test each type?
- SDLCs vs Twin Peaks?
 - SDLCs – designing/planning/testing – process
 - Twin Peaks – coarse-grained move into more fine-grained issues – back & forth between problem (requirements) and the solution (design)

Writeup: Analogy to Architecture of Buildings



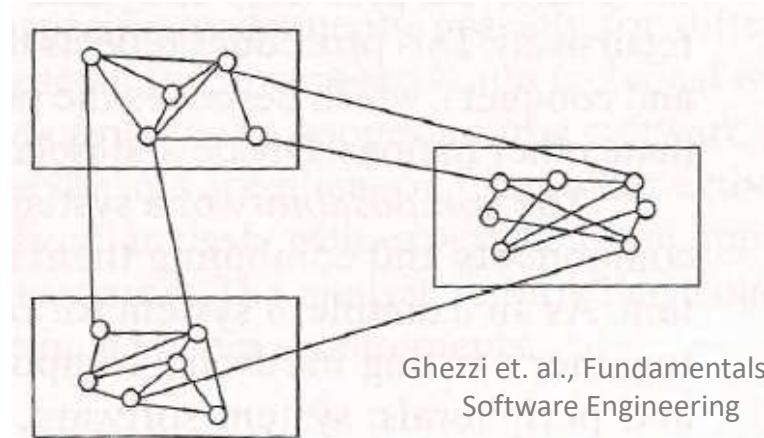
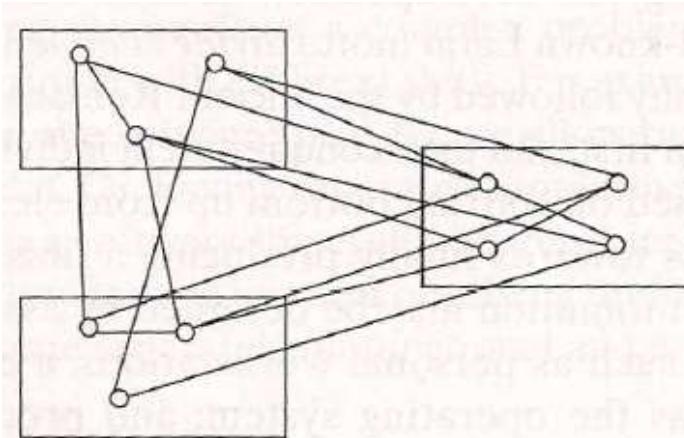
- Review section 1.1 in the book
 - What are the insights presented?
 - Are there other insights?
 - Do you agree with this analogy? Why or why not?
 - Where does the analogy break down?



Modularity



- High cohesion and low coupling
 - Cohesion: strength of relationship between elements within a module
 - Coupling: strength of relationship between different modules
- Which one?



Ghezzi et. al., Fundamentals of Software Engineering

Exercise: Making design decisions



- A requirements specification or a story contains lots of useful information to be leveraged during design
 - Nouns: modules/classes (Sometimes!)
 - Verbs: methods (Sometimes!)
 - Adjectives: properties/attributes/member variables (Sometimes!)
- Why?
 - To identify likely design elements
- Example:
 - “The system shall provide appropriate viewers for the user to read documents in the document store”



Making design decisions



- A requirements specification or a story contains lots of useful information to be leveraged during design
 - Nouns: modules/classes (Sometimes!)
 - Verbs: methods (Sometimes!)
 - Adjectives: properties/attributes/member variables (Sometimes!)
- Why?
 - To identify likely design elements
- Example:
 - “The system shall provide appropriate viewers for the user to read documents in the document store”



- Classes
 - viewers
 - Method: "read"
 - document
 - document store
 - user
- 

- Method
- read
- Adjective
- appropriate
- -----
- User --> read method
- read should not be declared in User class
- User class {
- public read {
- }
- }
- Security pattern?
- user - ability to access documents



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures

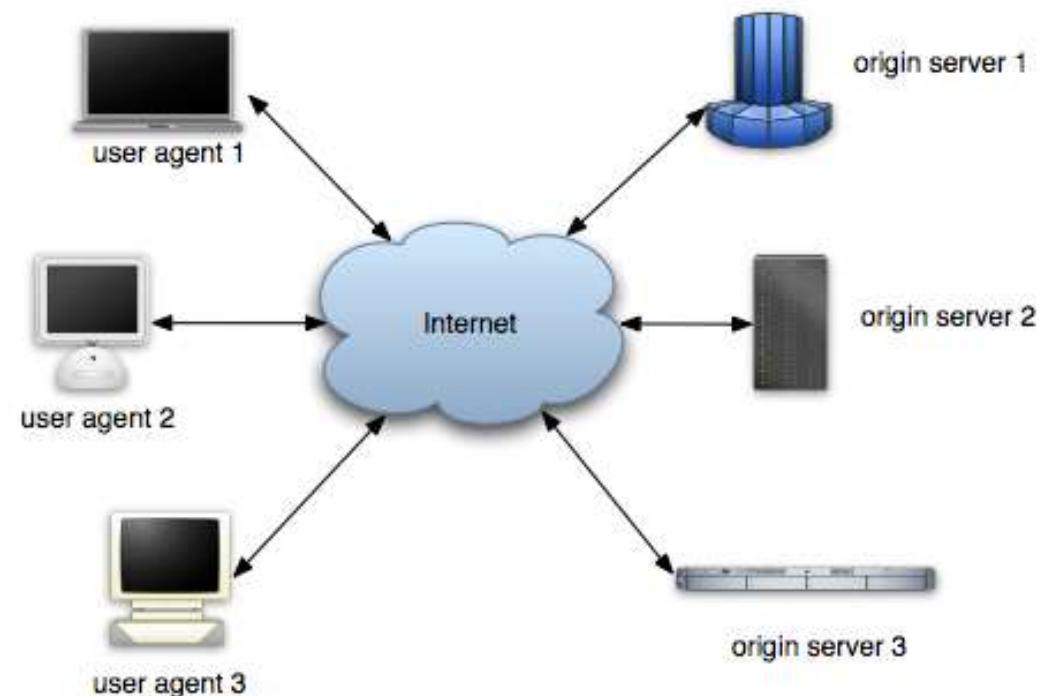


Hazeline Asuncion
Spring 2023

Review of Week 1



- Existing systems
 - What principal design decisions were made in the following systems?
 - Hide details - abstraction
 - Modularity
 - Communication via Internet
 - Client-Server
 - User agents (Clients)
 - Origin servers (Servers)

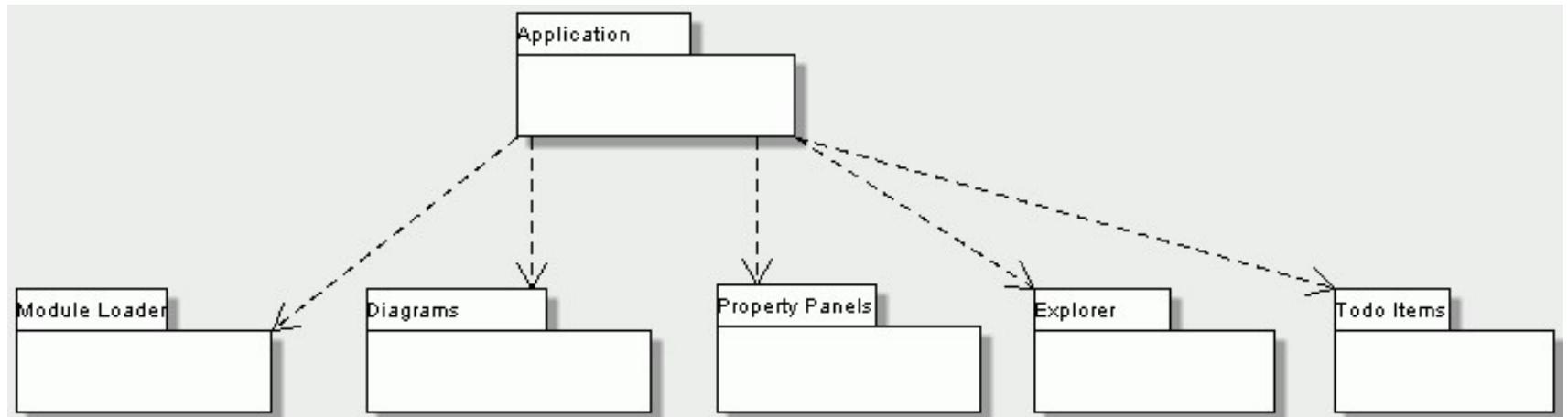


Each team prepare an answer for questions below. 5 mins

Review of Week 1



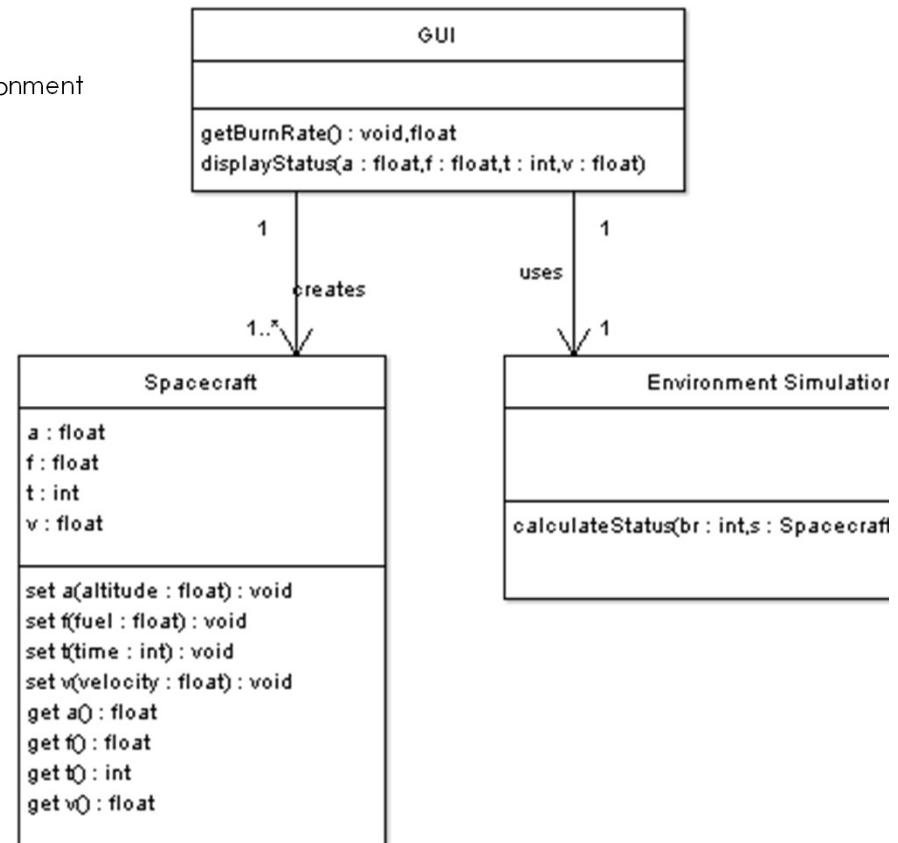
- Existing systems
 - What principal design decisions were made in the following systems?
 - Clustered
 - They all start in Main Application
 - No communication between components
 - Only between Main Application and other package



Review of Week 1



- Existing systems
 - What principal design decisions were made in the following systems?
 - Separate concerns
 - Functionality
 - GUI is starting point, creates SpaceCraft & uses Environment



Review of Week 1



- System under development
 - How to make design decisions?
 - First step is understand the requirements: functional & non-functional
 - Analyze, investigate **different technologies** ? –
 - Use design patterns
 - Collaborate & communicate – within team
 - Simple design
 - How to make *principal* design decisions?
 - Big decisions
 - Technologies (e.g., frameworks)
 - Requirements – behaviors, key connectors, best practices
 - What problems? Future issues?
 - NFRs
 - Difference between principal dd & dd? Principal dd – overall system
 - Twin Peaks Model?
 - Twin peaks vs Agile?
 - Should we always follow OOD for all products?

Review of Week 1



- Twin Peaks Model?
 - Model of development: develop requirements & arch together, start with vague requirements, cycle between the two, becoming more detailed as we move along
- Twin peaks vs Agile?
 - Agile – for software development (plan, design, implement,etc), Twin peaks is more focused on requirements & architecture
- Should we always follow OOD for all products?
 - No, has advantages/disadv
 - Adv: modularity, abstraction, reusability, inheritance,
 - » Easy to maintain
 - Disadv: sees everything as objects (encapsulation), connector is proc call,

OOD



- Should we always follow OOD for all products we design?
- What are pros & cons to OOD?
 - Pros
 - Cons
 - -

Review of Week 2

- 
- What are design tools? *The Boffins, Team 4, Ludus*
 - Abstraction – concept - simple machine
 - Separation of concerns – divide problem into independent parts. Think about tradeoffs
 - Refined experience – prior work (success/failure)
 - How do you design for novel systems (no similar system created before)? *To be decided, Magratheans, Dawn*
 - Brainstorm – different solutions
 - Change perspective of the problem (something closer to what was done before)
 - Literature – other solutions
 - Establish clear requirements, understand problem domain
 - Prototyping – help test ideas
 - Morphological charts – identify key functions
 - Assume different types of characters
 - Different outfit of characters based on types
 - Faces match, colors match
 - Analogy searching – find similar systems in other fields
 - Removing mental blocks – reframe the question to something more familiar
 - How do you recover design? *Team 9*

Review of Week 2



- How do you recover design? Team 9
 - Obtain source code –
 - Look for components (e.g. classes)
 - Look for relationships (calls, methods)
 - Look for design pattern

Questions from Reflection



- I had a couple of questions about architecture and design:
- How is specification related to architecture and design? Is the specification of a system the same as its prescriptive architecture?
- What criteria can we use to decide whether a design decision counts as a principal design decision?

ASSIGNMENTS

Upcoming



Group Assignments

- Design Presentation Sign-up – due before class
 - Suggest first presenters to send me a draft of presentation for feedback
- G1 – next week
 - Manual recovery
 - Automated recovery

ACTIVITIES

Exercise 2a: Component / Connector



- Take the design of the open source project you selected last time
- What are the components?
- Can you identify the connectors?

Exercise 2b: Making design decisions



- You are contracted to provide a turn-key software solution for the Cascadia / UWB Library Website. The website will allow librarians to enter new library items and it will allow library patrons to search for books, articles, various media items, etc. The website will also provide other functionality such as providing library information, supporting instant messaging (to ask librarians a question), and allowing patrons to check their accounts.

Making design decisions



- Has this type of problem been solved before?
- Brainstorm at least 5 design decisions you might make for this system (5 minutes)
- From the list of design decisions, which ones are “principal”?
- Sketch your architecture, representing these principal design decisions (10 mins)

READ THIS FIRST - 25 minutes

Topic: Basic Architecture Concepts

This activity is designed to

- Make design decisions based on a project description **or**
- Identify components and connectors on an open source project you selected
- Tie up loose ends regarding upcoming assignment(s)

Roles:

- **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.
- **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.
- **Note Taker** - takes record of the group’s discussion in the Google doc (below).
- **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity or the assignments due tonight, type in Red text

READ THIS FIRST - 25 minutes

Topic: Basic Architecture Concepts

This activity is designed to

- Make design decisions based on a project description **or**
- Identify components and connectors on an open source project you selected
- Tie up loose ends regarding upcoming assignment(s)

Roles:

- **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.
- **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.
- **Note Taker** - takes record of the group’s discussion in the Google doc (below).
- **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity or the assignments due tonight, type in Red text

Team Name: Dawn

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2b: Brainstorm at least 5 design decisions for the following system. From this list, which ones are principal? Sketch your architecture. You may add another slide if needed. (20 minutes)

You are contracted to provide a turn-key software solution for the Cascadia / UWB Library Website. The website will allow librarians to enter new library items and it will allow library patrons to search for books, articles, various media items, etc. The website will also provide other functionality such as providing library information, supporting instant messaging (to ask librarians a question), and allowing patrons to check their accounts.

Design decisions: Principal? (yes or no) (5mins)

#1 **Client-server** (yes): Client is the browser and the server is like a database containing the library items along with other information.

#2 **Communication** (yes): Client is communicating with the server by sending requests.

#3 **Security** (yes): Implement security best practices to protect user data and ensure the website's integrity, including secure authentication and authorization mechanisms, data encryption, and protection against common web vulnerabilities like SQL injection and cross-site scripting (XSS). The general public does not have access to the website, since only library patrons can search the website.

#4 **Modular architecture** (yes): Organize the website's functionality into modular components, such as a search module, instant messaging module, and account management module. This will allow for easier maintenance, updates, and future feature additions.

#5 **Scalability** (yes): Design the website to handle an increasing number of users and library resources over time.

List Roles and Student names

Moderator : Abdul

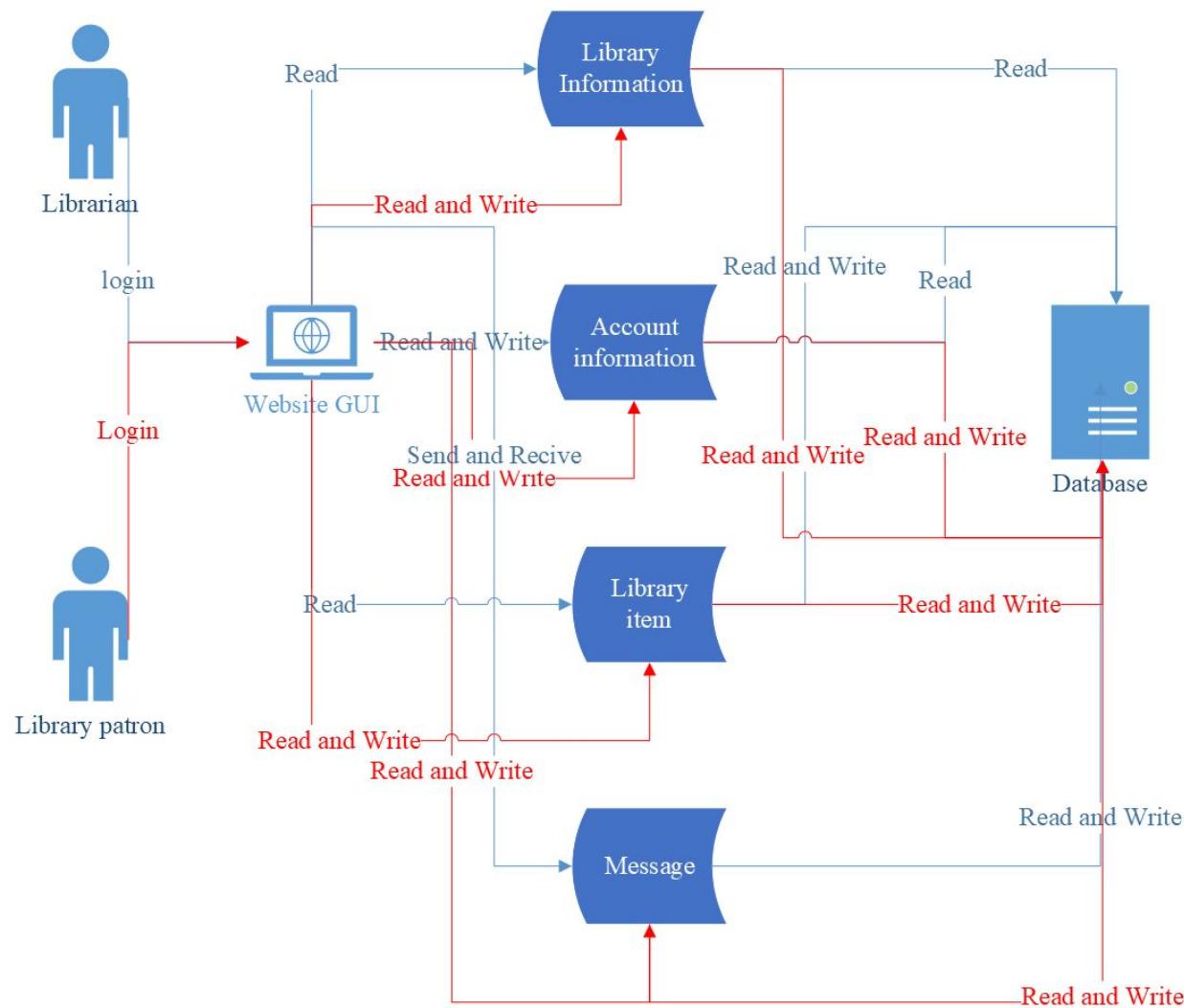
TimeKeeper: Luo

NoteTaker: Chloe

Reporter: Barack

Team Name: Dawn

Architecture Sketch



Team Name: The Boffins

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2b: Brainstorm at least 5 design decisions for the following system. From this list, which ones are principal? Sketch your architecture. You may add another slide if needed. (20 minutes)

You are contracted to provide a turn-key software solution for the Cascadia / UWB Library Website. The website will allow librarians to enter new library items and it will allow library patrons to search for books, articles, various media items, etc. The website will also provide other functionality such as providing library information, supporting instant messaging (to ask librarians a question), and allowing patrons to check their accounts.

Design decisions: Principal? (yes or no)

#1: **Principal Decision:** Client-Server architecture (separate into DB, Website, etc.)

#2: **Principal Decision:** MVC (Model-View-Controller) pattern

#3: **Principal Decision:** Semantic clustering of library items (books, articles, media, etc.)

#4: **Principal Decision:** Separation of concerns (separate the messaging interface from the search interface, for example)

#5: **Principal Decision:** Security - use SSL oAuth for AuthN.

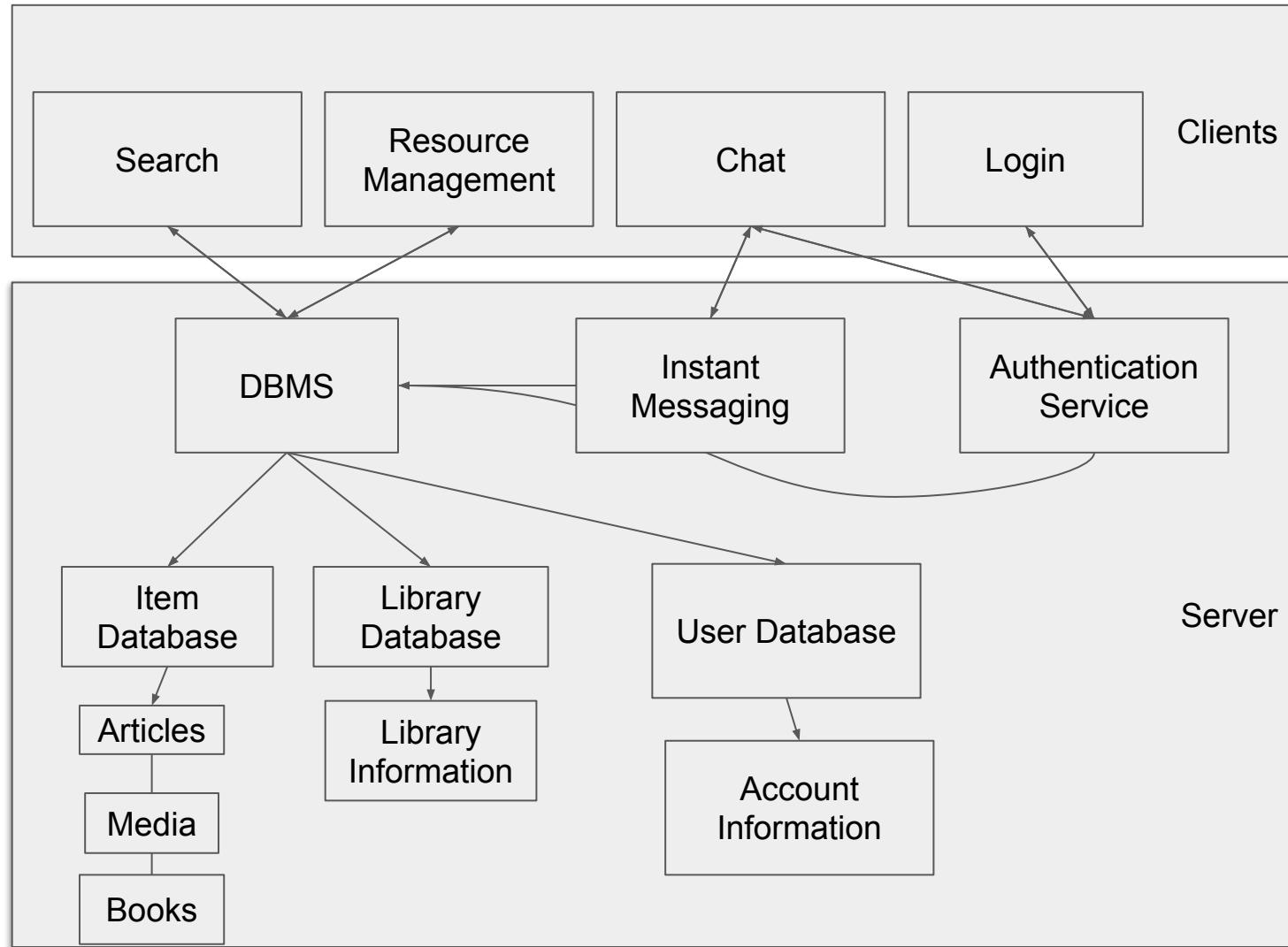
List Roles and Student names

Moderator : Ioana Preda

TimeKeeper: Holly East

NoteTaker: Sukeerth Vagaraju

Reporter: Jaron Wang



Team Name: To Be Decided

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1: If connectors aren't elicited, would we normally infer them (or omit)?

Ques #2:

Task 2a: Using the design diagram for your open source project, identify the components and connectors. You may copy the design diagram here and specify which ones are components and connectors. Most design diagrams may not contain connectors, so you might have to add this. You may add another slide if needed (20 mins). As before, specify name of OSS and url.

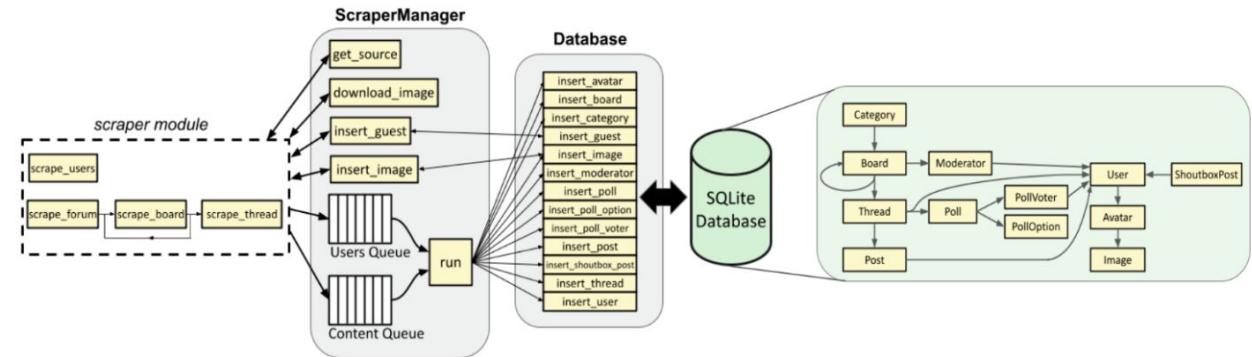
- This is a general diagram for a webscraper (**beautifulsoup** would be analyzed further at a later time)
- Although we plan to look into an HTML scraper, this diagram is more of a spider/crawler

Primary Components:

- Scraper module (collector)
- Scraper manager (analyzer)
- Database (storage)

Connectors:

- Queues (user, content)
- Manager (between module and database)
- database)
- Interface between Database and Manager



Reference:

<https://nrsyed.com/2021/11/26/asynchronously-web-scraping-a-proboards-forum-with-python-part-2/>

List Roles and Student names

Moderator : Sonal Yadav

TimeKeeper: Naima Noor

NoteTaker: Angelo Williams

Reporter: Noura Alroomi

Team Name: Team 4

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Task 2a: Using the design diagram for your open source project, identify the components and connectors. You may copy the design diagram here and specify which ones are components and connectors. Most design diagrams may not contain connectors, so you might have to add this. You may add another slide if needed (20 mins). As before, specify name of OSS and url.

Scrapy

Details in next page.

Reference: <https://docs.scrapy.org/en/latest/topics/architecture.html>

List Roles and Student names

Moderator : Warren Liu

TimeKeeper: Chengjun Xi

NoteTaker: Haihan Jiang

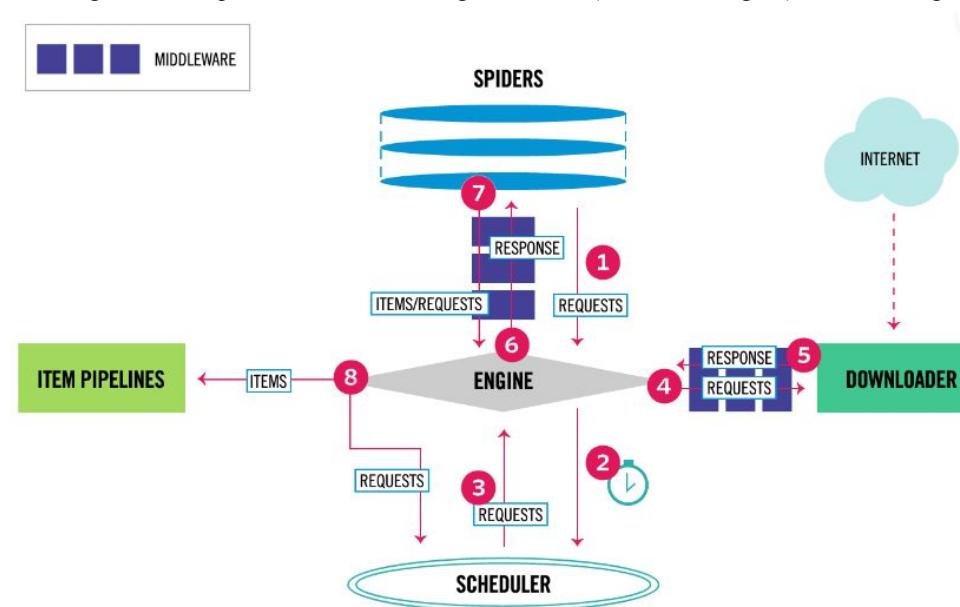
Reporter: Joshua Medvinsky

Components

1. **Scrapy Engine:** controlling the data flow between all components of the system, and triggering events when certain actions occur
2. **Downloader:** fetching web pages and feeding them to the engine which, in turn, feeds them to the spiders.
3. **Spiders¶:** custom classes written by Scrapy users to parse responses and extract items from them or additional requests to follow.
4. **Item Pipeline:** processing the items once they have been extracted (or scraped) by the spiders. Typical tasks include cleansing, validation and persistence (like storing the item in a database).

Connectors

1. **Downloader middlewares:** The downloader middleware is a framework of hooks into Scrapy's request/response processing. It's a light, low-level system for globally altering Scrapy's requests and responses.
2. **Spider middlewares:** The spider middleware is a framework of hooks into Scrapy's spider processing mechanism where you can plug custom functionality to process the responses that are sent to Spiders for processing and to process the requests and items that are generated from spiders.
3. **Scheduler:** receives requests from the engine and enqueues them for feeding them later (also to the engine) when the engine requests them.



Team Name: Ludus

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Task 2b: Brainstorm at least 5 design decisions for the following system. From this list, which ones are principal? Sketch your architecture. You may add another slide if needed. (20 minutes)

You are contracted to provide a turn-key software solution for the Cascadia / UWB **Library** Website. The website will allow **librarians** to enter new library items and it will allow library **patrons** to search for books, articles, various media items, etc. The website will also provide other functionality such as providing library information, supporting instant messaging (to ask librarians a question), and allowing patrons to check their accounts.

Design decisions: Principal? (yes or no)

#1. P - Modules:

- WFE (GUI, Accounts Controls, Admin Controls, Search Controls, Item Detail Controls)
- AppServer (Search, Common APIs, User management controller, Library management controller, Models, Notification Jobs)
- Backend(Storage, Sprocs)

#2. P - Use MVC architecture pattern

#3. P - Frameworks / Tech: React / C# .Net Core / MySQL

#4. P - AWS hosting Cloud Web Service w/ separate MySQL backend.

- Arch should support high availability and reliability.
- Don't want down time for students to be unable to access books or to mislabel things and make them late.

#5. D - Core classes: Item -> Book/Article/Media, Library, ItemManager, User -> Patron, Librarian

#6 - P - COTS IM integration. Build custom support for HOOPS.

#7 - P - Use/integrate COTS CMS

#8 - P - Authentication/authorization w/SSO support to UW auth servers (OAuth)

- Roles for admin and user

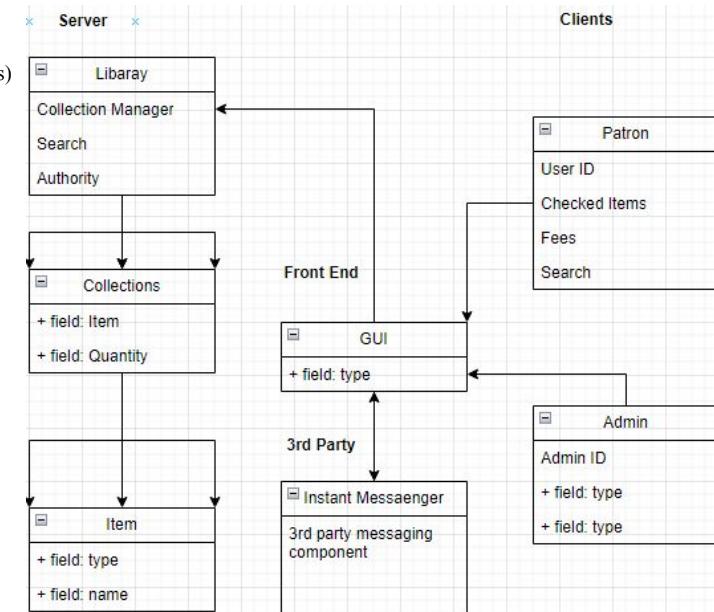
List Roles and Student names

Moderator : Austin

TimeKeeper: Agustin

NoteTaker: Tyson

Reporter: Shaun



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Do either Task 2a or 2b

Task 2a: Using the design diagram for your open source project, identify the components and connectors. You may copy the design diagram here and specify which ones are components and connectors. Most design diagrams may not contain connectors, so you might have to add this. You may add another slide if needed (20 mins). As before, specify name of OSS and url.

Task 2b: Brainstorm at least 5 design decisions for the following system. From this list, which ones are principal? Sketch your architecture. You may add another slide if needed. (20 minutes)

You are contracted to provide a turn-key software solution for the Cascadia / UWB Library Website. The website will allow librarians to enter new library items and it will allow library patrons to search for books, articles, various media items, etc. The website will also provide other functionality such as providing library information, supporting instant messaging (to ask librarians a question), and allowing patrons to check their accounts.

Design decisions: Principal? (yes or no)

#1

#2

#3

#4

#5

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: The Magratheans

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2b: Brainstorm at least 5 design decisions for the following system. From this list, which ones are principal? Sketch your architecture. You may add another slide if needed. (20 minutes)

You are contracted to provide a turn-key software solution for the Cascadia / UWB Library Website. The website will allow librarians to enter new library items and it will allow library patrons to search for books, articles, various media items, etc. The website will also provide other functionality such as providing library information, supporting instant messaging (to ask librarians a question), and allowing patrons to check their accounts.

Design decisions: Principal? (yes or no)

#1 Client/server Users (librarians and patrons) (Principal)

#2 The website will use a database that includes librarians, patrons, library items, and messages. (Principal)

#3 Polymorphic library items (all items support, for instance, addition by librarian and search) (Principal)

#4 Librarians and Users are identical with respect to instant messaging functionality (more polymorphism) (Principal)

#5 Security Integration with UW IT NetID authentication (Principal)

List Roles and Student names

Moderator : Jason SJ Lim

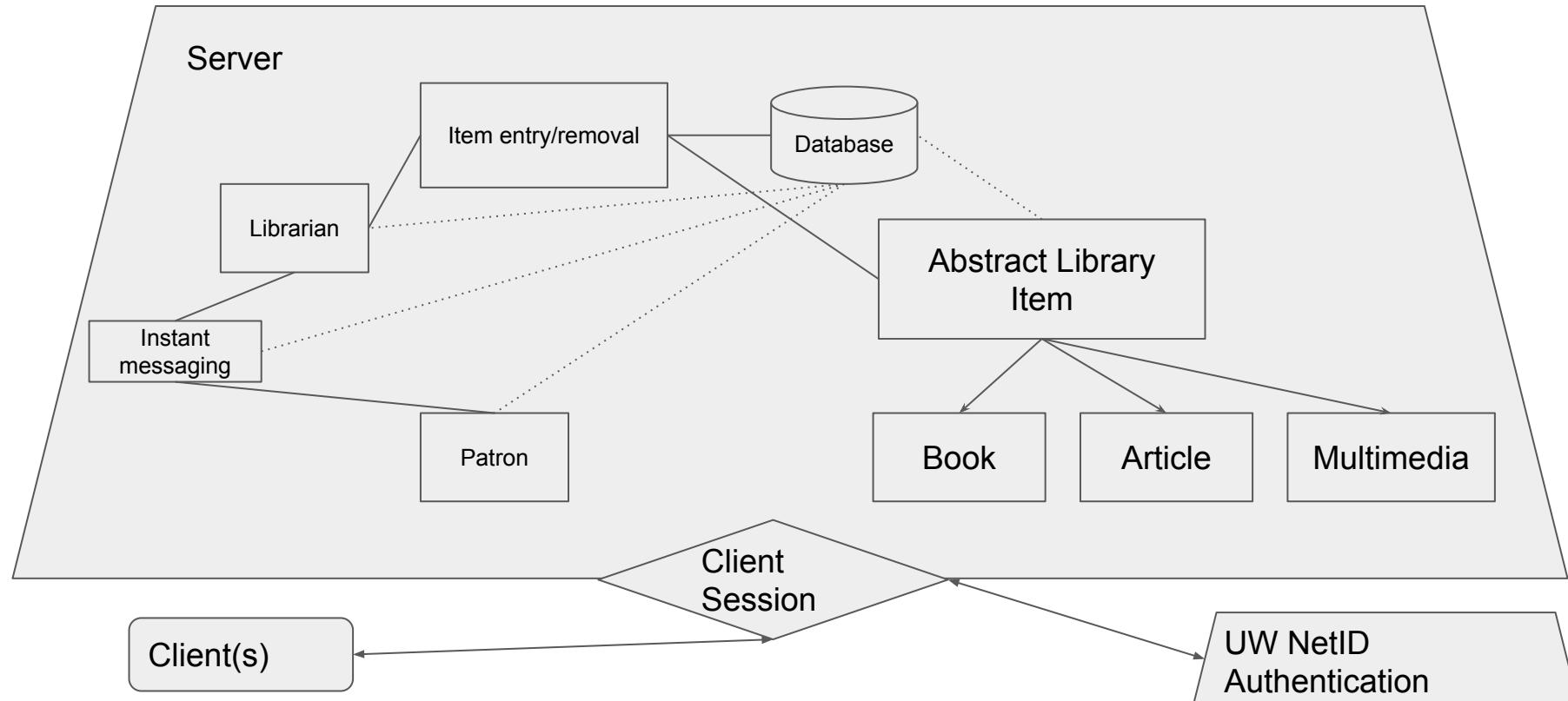
TimeKeeper: Karan Chopra

NoteTaker: Ryan Decker

Reporter: Thomas Pinkava

Notes

- client-server?
- hosted on a cloud, need web services
- Deciding the database items
 - Library items
 - Users
 - Librarians can enter new items, and they contacted
 - Polymorphism in terms of types of library
- Make an account for each person that accesses membership accounts)
 - Persistent session data
 - Permissions
- Nonfunctional requirements
- Supporting instant messaging



Team Name: Team 9

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Do either Task 2a or 2b

Task 2a: Using the design diagram for your open source project, identify the components and connectors. You may copy the design diagram here and specify which ones are components and connectors. Most design diagrams may not contain connectors, so you might have to add this. You may add another slide if needed (20 mins). As before, specify name of OSS and url.

Task 2b: Brainstorm at least 5 design decisions for the following system. From this list, which ones are principal? Sketch your architecture. You may add another slide if needed. (20 minutes)

You are contracted to provide a turn-key software solution for the Cascadia / UWB Library Website. The website will allow librarians to **enter new library items** and it will allow library patrons to **search for books, articles, various media items, etc.** The website will also provide other functionality such as **providing library information, supporting instant messaging** (to ask librarians a question), and allowing **patrons to check their accounts**.

Design decisions: Principal? (yes or no)

#1 User Interface Design: Color scheme, page layout.

Principle: No, the UI is important for the usability but it is not going to have an impact on the overall system as a whole

#2 Database Schema Design: Using NoSQL vs SQL databases

Principle: Yes, the way and format in which the database is used will have significant impact on the rest of the system, whether which type of data can be stored to its efficiency.

#3 Search Algorithm Design: Which search algorithm are we using, bubble search, sequential, hashing

Principle: Yes, with a more efficient algorithm it will be faster which “search” function could be a main bottleneck in a database service.

#4 Security Design: How are information being stored, authentication methods, etc

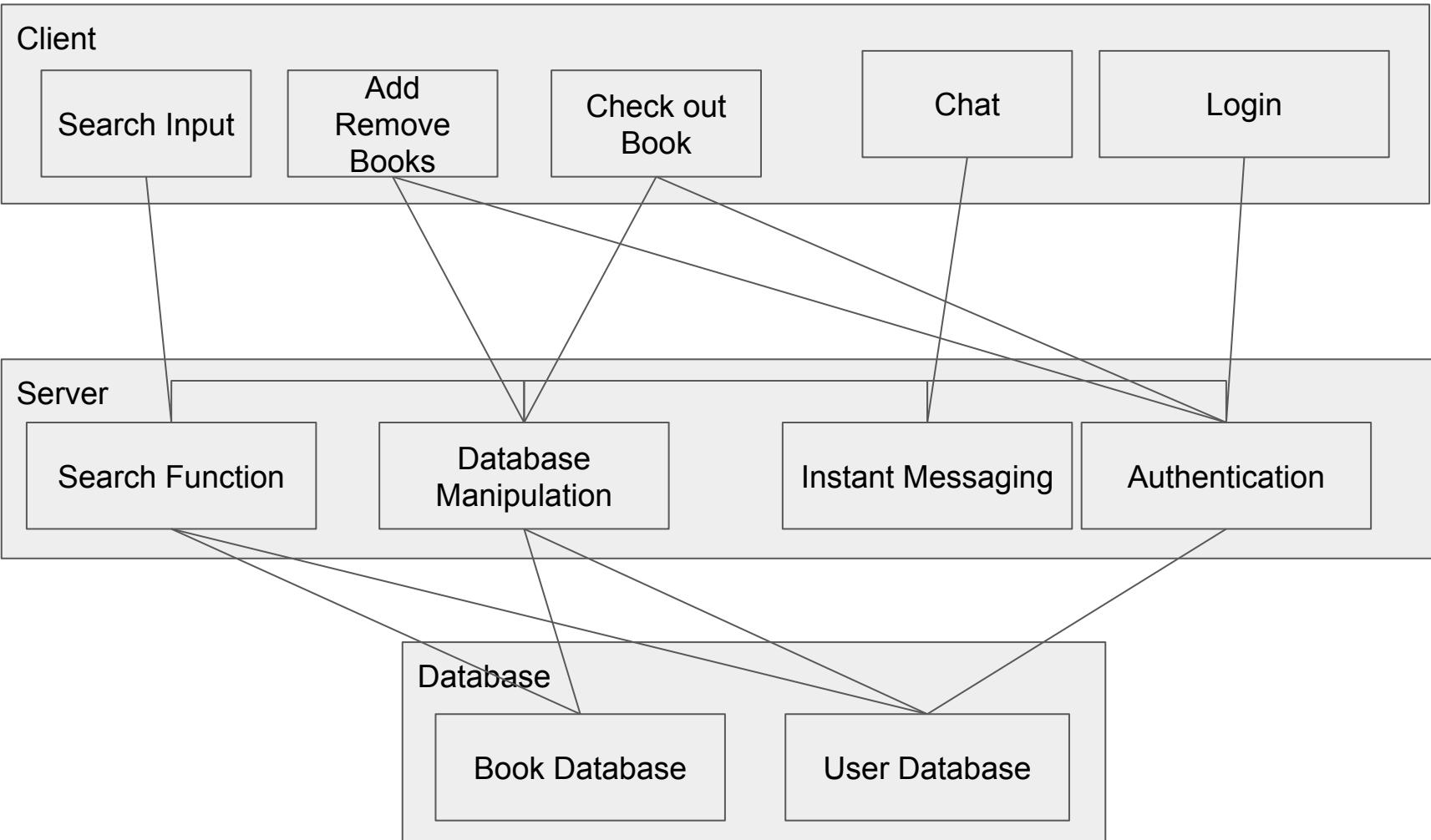
Principle: Yes, because website where we allow client to connect, one of a webserver's main functions must be its ability to implement authentication and access control.

#5 Message Format Design: How are the messages being kept or sent, which format is it sent as, json raw text, etc.

Principle: No, because messaging is just a part of the messaging system, which does not affect the main functionality of library database access.

[List Roles and Student names](#)

Moderator : Pragati Dode



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Do either Task 2a or 2b

Task 2a: Using the design diagram for your open source project, identify the components and connectors. You may copy the design diagram here and specify which ones are components and connectors. Most design diagrams may not contain connectors, so you might have to add this. You may add another slide if needed (20 mins). As before, specify name of OSS and url.

Task 2b: Brainstorm at least 5 design decisions for the following system. From this list, which ones are principal? Sketch your architecture. You may add another slide if needed. (20 minutes)

You are contracted to provide a turn-key software solution for the Cascadia / UWB Library Website. The website will allow librarians to enter new library items and it will allow library patrons to search for books, articles, various media items, etc. The website will also provide other functionality such as providing library information, supporting instant messaging (to ask librarians a question), and allowing patrons to check their accounts.

Design decisions: Principal? (yes or no)

#1

#2

#3

#4

#5

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Open Source Web Services

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Do either Task 2a or 2b

Task 2a: Using the design diagram for your open source project, identify the components and connectors. You may copy the design diagram here and specify which ones are components and connectors. Most design diagrams may not contain connectors, so you might have to add this. You may add another slide if needed (20 mins). As before, specify name of OSS and url.

List Roles and Student names

Moderator : Nur Dincer

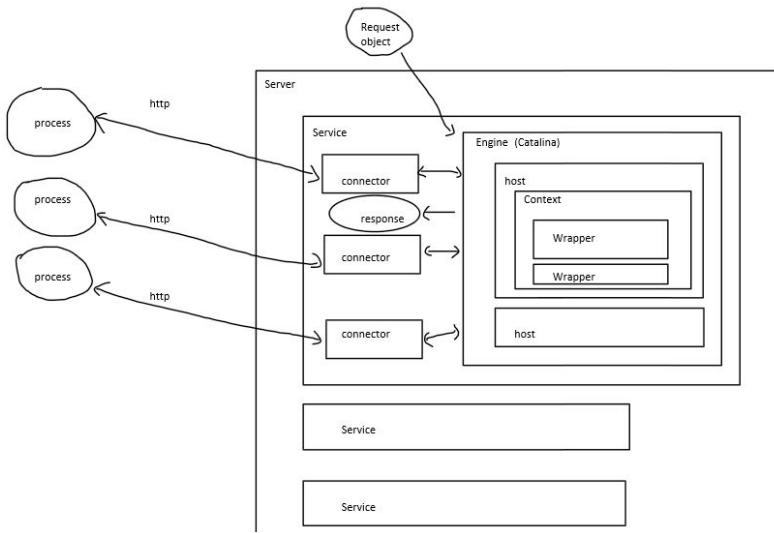
TimeKeeper: Anthony Bustamante

NoteTaker: Yang Yu

Reporter: Sunjil Gahatraj

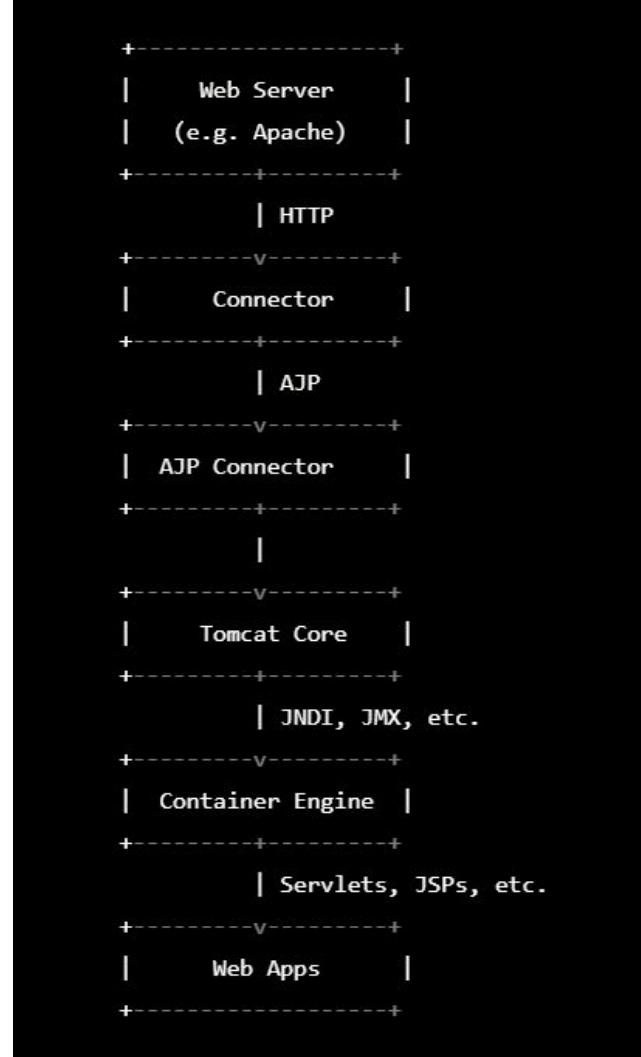
Name of OSS: Apache Tomcat

URL: <https://github.com/apache/tomcat>



Connectors: Connector, AJP Connector, HTTP, AJP

Components: Web server, connector, Web apps, servlet containers, Engine(Catalina), JNDI, JMX, JSPs



Web Server: This layer represents a front-end web server, such as Apache HTTP Server, which can handle static content, SSL/TLS encryption, load balancing, and other tasks.

Connector: This layer represents a connection between the web server and Tomcat, typically using the HTTP or AJP (Apache JServ Protocol) protocols. The Connector can handle incoming requests from the web server and forward them to the Tomcat Core.

AJP Connector: This is a special Connector that is optimized for communication between Apache and Tomcat using the AJP protocol.

Tomcat Core: This layer represents the heart of Tomcat, which provides a JNDI (Java Naming and Directory Interface) service, a JMX (Java Management Extensions) agent, and other services that are required by the Container Engine.

Container Engine: This layer represents the Servlet Container, which is responsible for managing the lifecycle of web applications and their components (such as Servlets, JSPs, Filters, and Listeners), and providing a runtime environment for executing web requests.

Web Apps: This layer represents the actual web applications that are deployed on Tomcat. Each web application has its own context path and contains a set of Servlets, JSPs, and other resources that are processed by the Container Engine.

JSP: JSP is a component in the Tomcat architecture, not a connector. JSP is a technology that allows developers to create dynamic web pages by embedding Java code in HTML pages, it doesn't take charge of any connection.

JNDI and JMX are not connectors because they do not directly handle the communication between clients and the Tomcat server. While JNDI and JMX provide important functionality for managing and accessing resources in Tomcat, they are not considered connectors because they do not directly handle the communication between clients and the Tomcat server. Connectors, such as the HTTP connector and the AJP connector, provide the low-level communication protocols that enable clients to connect to the Tomcat server and request resources.



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Questions – 5 mins to prep for an answer

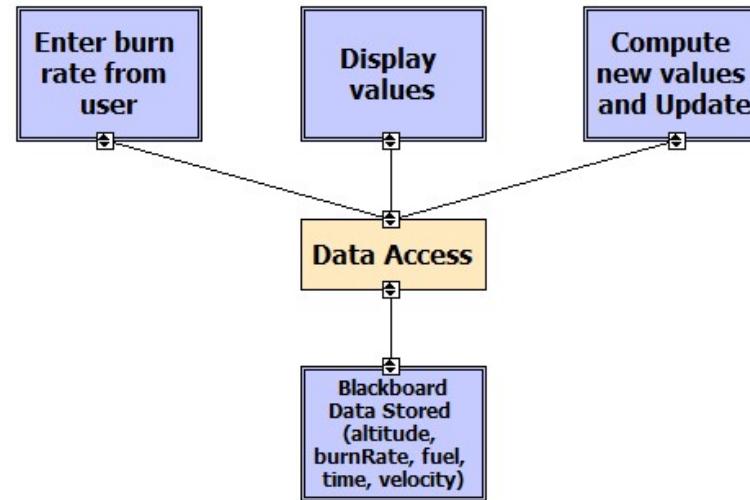


- What is the difference between explicit and implicit invocation? Provide at least one architecture style that uses each type of invocation. **Dawn**
- When would you use a 2-tiered Client-Server vs a 3-tiered Client-Server? Give specific scenarios for each. **The Boffins**
- When would you use a fat-client vs thin-client? Give specific scenarios for each. **To be decided**
- When would you use Mobile Code? Give one specific scenario for each. **Team 4, Team 9**
- Is Repository the same as Client-Server? Why or why not? **Ludus**
- Which architecture styles support distributed systems? **The Magratheans**
- Is it easy to transform software built for standalone machine into a distributed setting? Why or why not? **Open Source Web Services**

Questions



- What is the difference between explicit and implicit invocation? Provide at least one architecture style that uses each type of invocation. **Dawn**
- Implicit invocation – methods indirectly called – eg. repository
- Explicit invocation – invoke methods directly – object-oriented architecture



Questions



- When would you use a 2-tiered Client-Server vs a 3-tiered Client-Server? Give specific scenarios for each.

The Boffins

- 2-tiered client-server
 - No need to store data
 - Internet with home camera – no video is being stored - Pub-sub service
 - Some data stored – not much
- 3-tiered client-server
 - Data intensive
 - Game application – lots of data to store

Questions



- When would you use a fat-client vs thin-client? Give specific scenarios for each. **To be decided**
 - Fat-client
 - Processing in both client & server
 - Interactive intensive apps
 - Unreliable connection to server
 - Require more support
 - Powerful server is not needed
 - Thin-client
 - Powerful servr - required
 - Processing on the server side
 - Security type of applications – don't want to have data on the client-side
 - Small applications on client-side, mobile devices (not a lot of resources)
 - Reliable connection to the server

Questions



- When would you use Mobile Code? Give specific scenarios for each. **Team 4, Team 9**
- Explain differences for each scenario
 - Web application – Code on demand
 - IoT network – camera & image detection – sent to the server – Remote evaluation

Mobile Code variants compared to Client-Server

Paradigm	Before		After	
	S_A	S_B	S_A	S_B
<i>Client-Server</i>	A	know-how resources B	A	know-how resources B
<i>Remote Evaluation</i>	know-how A	resources B	A	<i>know-how</i> resources B
<i>Code on Demand</i>	resources A	know-how B	resources <i>know-how</i> A	B
<i>Mobile Agent</i>	know-how A	resources	—	<i>know-how</i> resources A

Source: Carzaniga, A., Picco, G.P., Vigna, G.,
 Designing Distributed Applications with Mobile Code
 Paradigms, ICSE 1997.

Mobile code



- Remote evaluation
 - when have large amounts of data and cheaper to move code than data
 - Centralized business logic, but still needs additional logic somewhere else
- Code on demand
 - Data is on local machine, but want to do processing on local side (data can't be moved), private data
 - Plug-ins, extension (ide, browser, etc)
- Mobile agent
 - viruses, network diagnostics
- Cloud Computing - does this use mobile code?

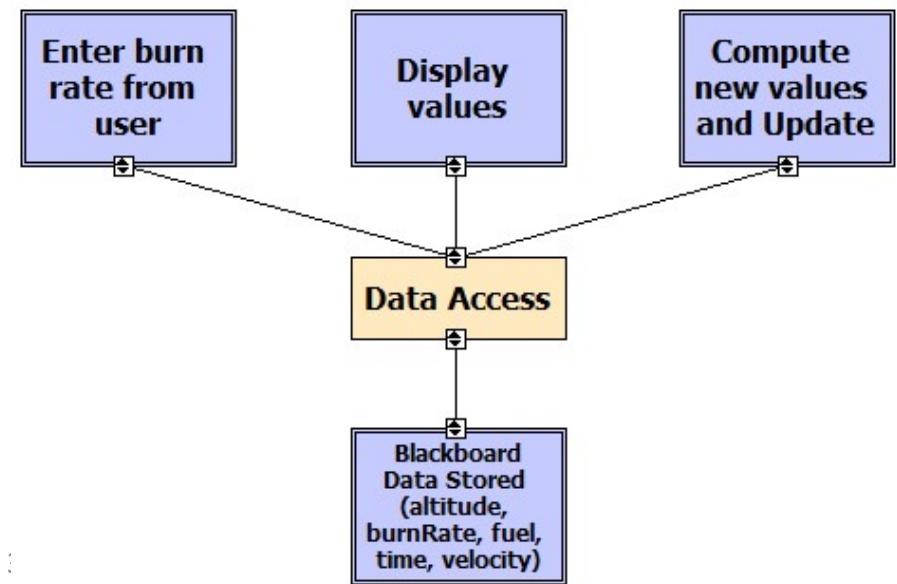
Questions



- Is Repository the same as Client-Server? Why or why not?

Ludus

- No. Repository is more about storing info on a central place; uses implicit invocation
 - Implicit invocation – get notifications, alerts, components decide when to respond
- Client-Server – focus is more on services that server offers. Server offers services to Client, uses explicit invocation



Questions



- Which architecture styles support distributed systems? **The Magratheans**
 - Batch-sequential, pipe & filter – more efficient
 - Mobile code, client-server, repository
 - Object-oriented needs middleware
 - Main-program & subroutines – calls need to be supported over network connection
- Is it easy to transform software built for standalone machine into a distributed setting? Why or why not? **Open Source Web Services**
 - **No, not easy –**
 - Change the architecture of software
 - Require appliances, for memory storage
 - Coordinate distributed systems- maintain consistency
 - Implement – remote calls/procedures
 - Security – re-assign security of system
 - Network delay (*latency*), other network issues
 - Failover, redundancy

Reflection question



- Fat-client vs Code-on-Demand Mobile code
 - Presumably a fat client is already possessed of all of the necessary logic it uses, but when one considers a modern web browser, say, where the browser is a large and sophisticated machine being fed large and sophisticated JavaScript applets to run, it's clear that the lines are being blurred a bit.

DESIGN PRESENTATION

REMINDER & ACTIVITIES

Reminders



- G1 – due tomorrow before midnight
 - It will take about a week to grade
- G1 group assessment – only submit if have issues with teammates
 - Detect team issues early in quarter
- Quiz – next week
- Website updated
 - All group assignments are posted
 - All readings and recorded lectures are posted
 - Grader office hours are posted – Welcome Page

Apply Arch Styles (Task 2a)



- Each group will be given a system
- Your task is to select an architecture style
- Sketch your arch diagram

Assigned systems – OK to make assumptions



- Onboard flight systems – Open Source WS
- Interactive TV – The Boffins
- Patient Records System - Dawn
- Stopwatch - Magratheans
- Emergency response system – Team 9
- Sprinkler System – Team 4
- Online library – To be decided
- Personal Contacts Manager – Team 3
- Video Stream - Ludus
- Flight Reservation System

READ THIS FIRST - 20 minutes for each activity

Topic: Architecture Styles

This activity is designed to

- Give you practice in designing
- Apply architecture styles you learned this week
- Tie up loose ends regarding upcoming assignment(s)

Roles:

- **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.
- **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.
- **Note Taker** - takes record of the group’s discussion in the Google doc (below).
- **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion
3. All groups will do Task 2a for first 20 min. Then I will assign your group for either Task 2b or 2c for the 2nd 20 mins

If you have any questions about this activity or the assignments due tonight, type in Red text

Team Name: Dawn (Patient Record System)

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)

The software system should be able to manage patient medical records, including medical history, medications, test results, and other important information. The architecture of such a system must be designed to ensure the confidentiality, integrity, and availability of patient data, as well as to provide an efficient and user-friendly interface for healthcare providers.

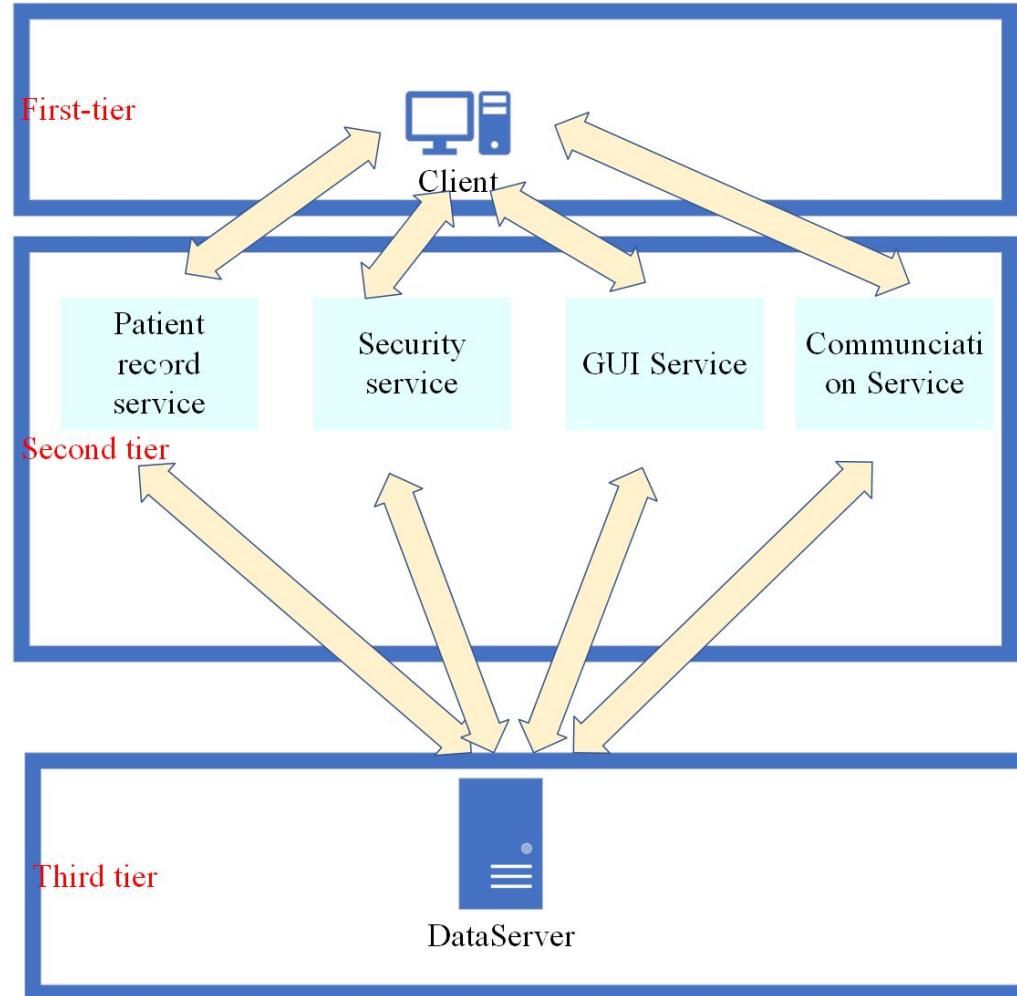
List Roles and Student names

Moderator : Abdul-Muizz Imtiaz

TimeKeeper: Luo

NoteTaker: Chloe

Reporter: Barrack



Team Name: The Boffins

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1: For G1, do we turn in one document per group or do we turn turn the document in individually?

[Prof] Yes, one document per group

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)

Topic: interactive TV (smart TV, Netflix program)

Notes:

- Definition of interactive TV
- streaming
- fat-client, lots of interaction
- rule based interpreter to scale user input
- mobile code, large dataset needs to be located

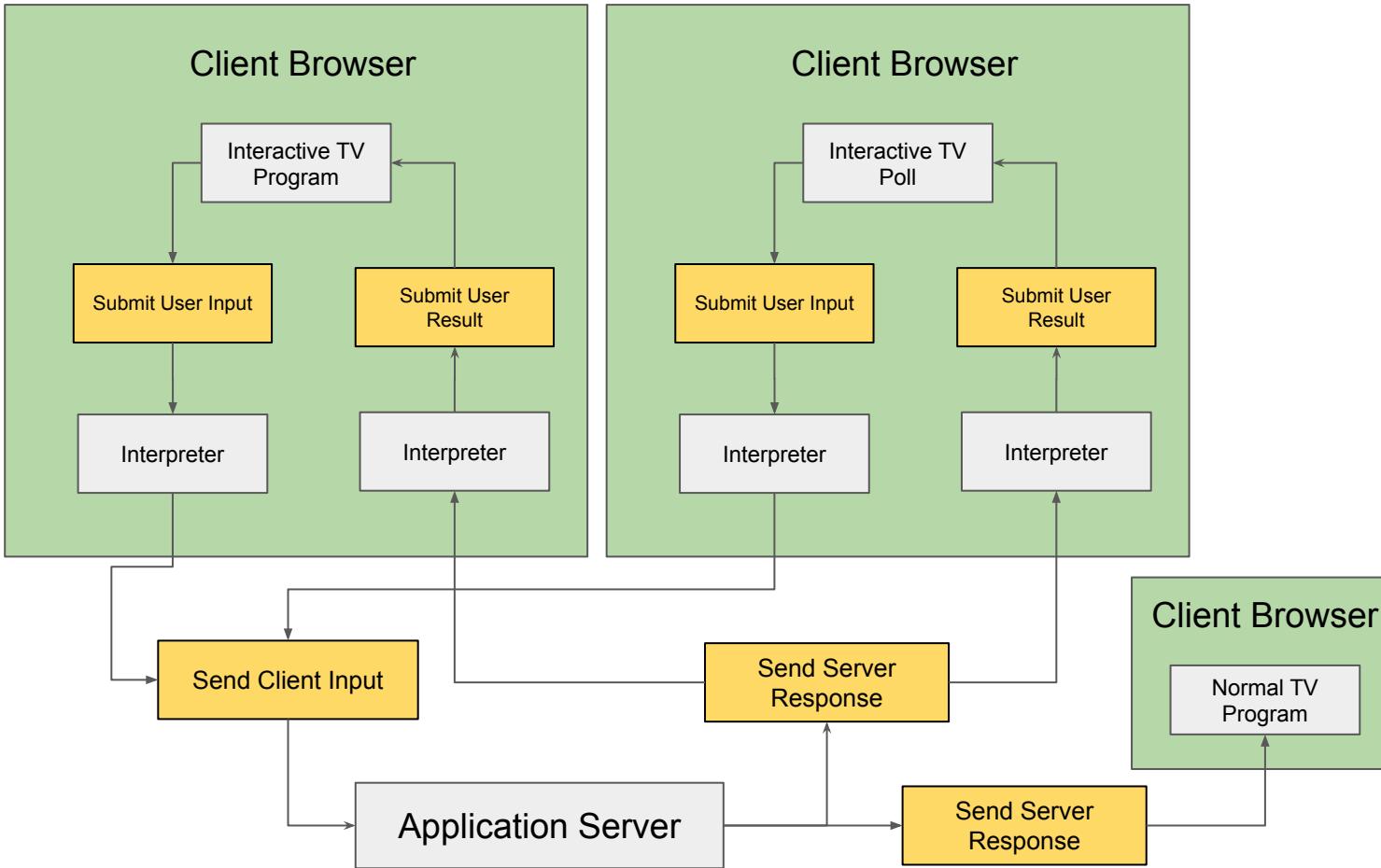
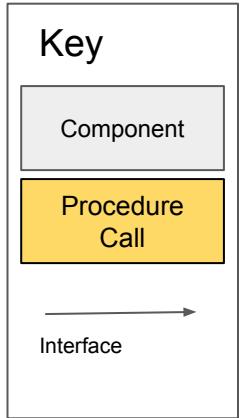
List Roles and Student names

Moderator : Jaron Wang

TimeKeeper: Sukeerth Vagaraju

NoteTaker: Holly East

Reporter: Ioana Preda



Team Name: To be decided - Online Library

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1: For typical server-third party interactions, are there any common patterns in terms of Mobile Code?

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)

The architecture resembles an MVC pattern (not included in the common list but worthy of consideration) in ways since the implementation of one of the components may have this type of pattern embedded within it. Consider when you render the HTML in the browser and how it produces a view. The client could also be seen as a controller depending on the desired goal.

There are elements of Code on Demand, as JavaScript is included as part of the resources. There may be a video player implemented as one of its resources.

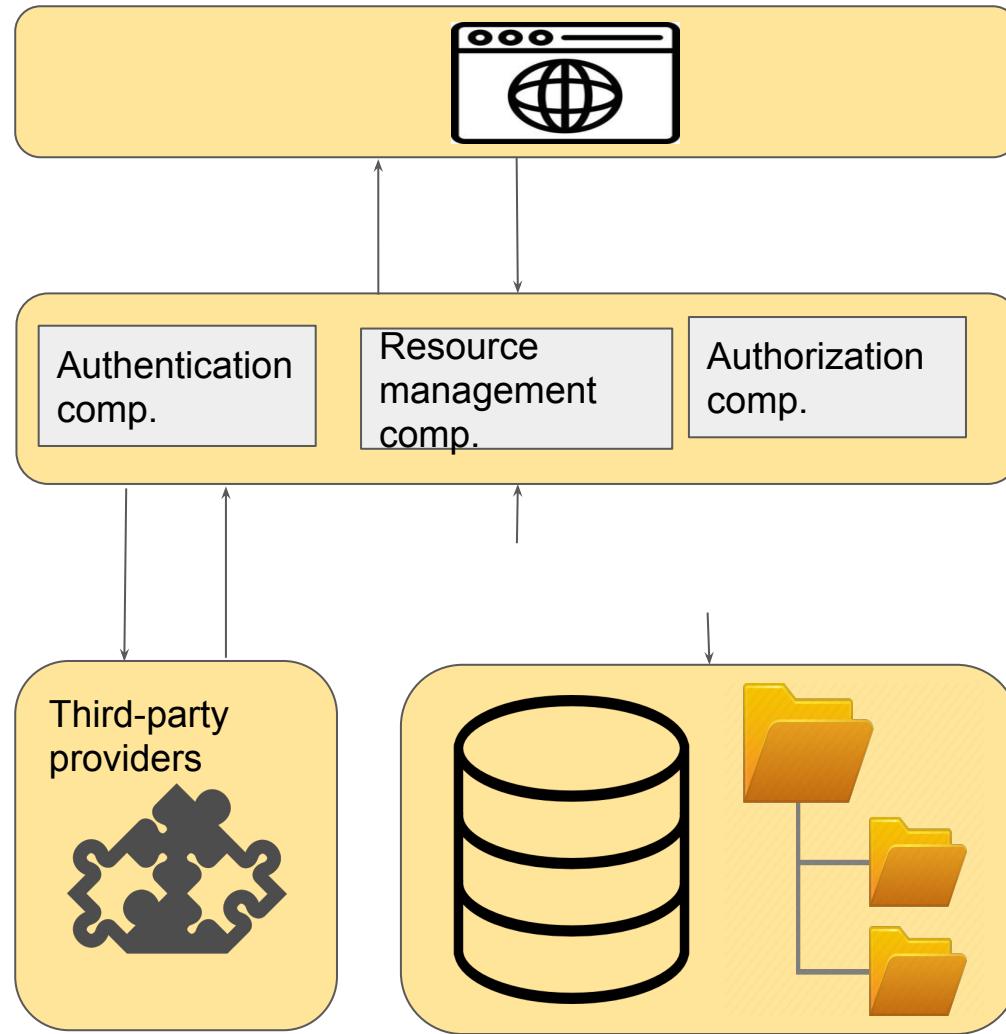
List Roles and Student names

Moderator :

TimeKeeper: Naima Noor

NoteTaker: Angelo Williams

Reporter: Noura Alroomi



Team Name: Team 4

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes) System is the sprinkler system.

- Sensor, timer -> input
- Input -> Control Engine (contains the pre-defined rules, evaluates the inputs and rules) -> send evaluated rules
- evaluated rules-> controller -> send signal to whether activate, deactivate, or nothing the sprinkler system
- Pipeline-> schedule transport from source to sprinkler

And

Task 2b: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

.....using a pencil and paper...

...be able to explain its meaning...

...and money is no object...

Take a picture of your design and paste it here (or you can directly create it on this slide). You may add another slide if needed. . (15 minutes)

Or

Task 2c: Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

...using whatever method you like...

...Cost is a concern. Make sure that the cost of implementing your design is reasonable...You may add another slide if needed. . (15 minutes)

List Roles and Student names

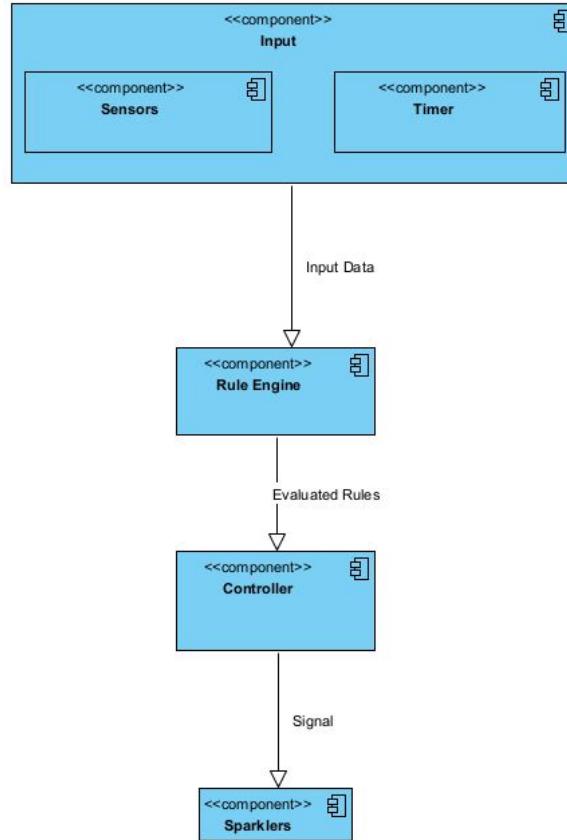
Moderator :

TimeKeeper: Joshua Medvinsky

NoteTaker:

Reporter:

Team 4 Diagram



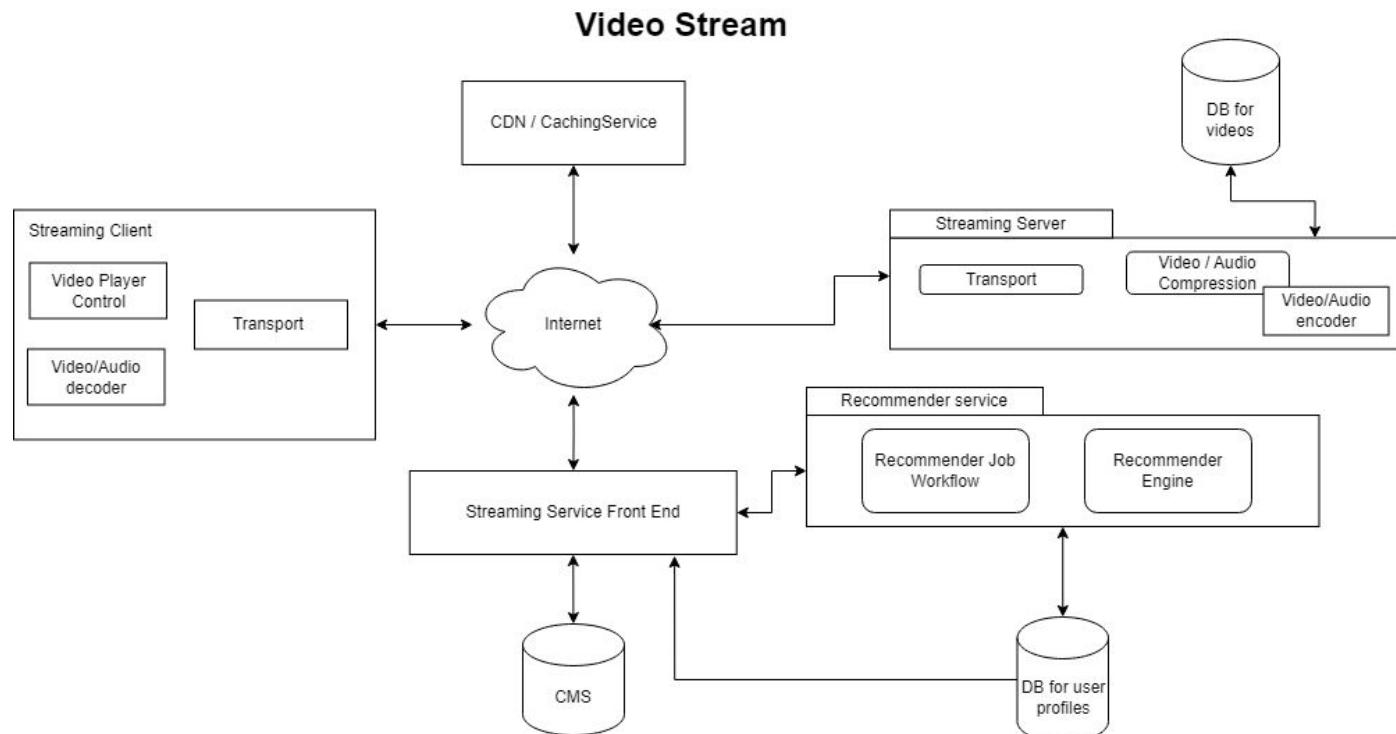
Team Name: Ludus - Video Stream

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)



List Roles and Student names

Moderator : Shaun

TimeKeeper: Austin

NoteTaker: Tyson

Reporter: Agustin

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)

And

Task 2b: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

.....using a pencil and paper...

...be able to explain its meaning...

...and money is no object...

Take a picture of your design and paste it here (or you can directly create it on this slide). You may add another slide if needed. . (15 minutes)

Or

Task 2c: Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

...using whatever method you like...

...Cost is a concern. Make sure that the cost of implementing your design is reasonable... You may add another slide if needed. . (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: The Magratheans

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)

<See next slide for details>

List Roles and Student names

Moderator : Thomas

TimeKeeper: Karan

NoteTaker: Jason

Reporter: Ryan

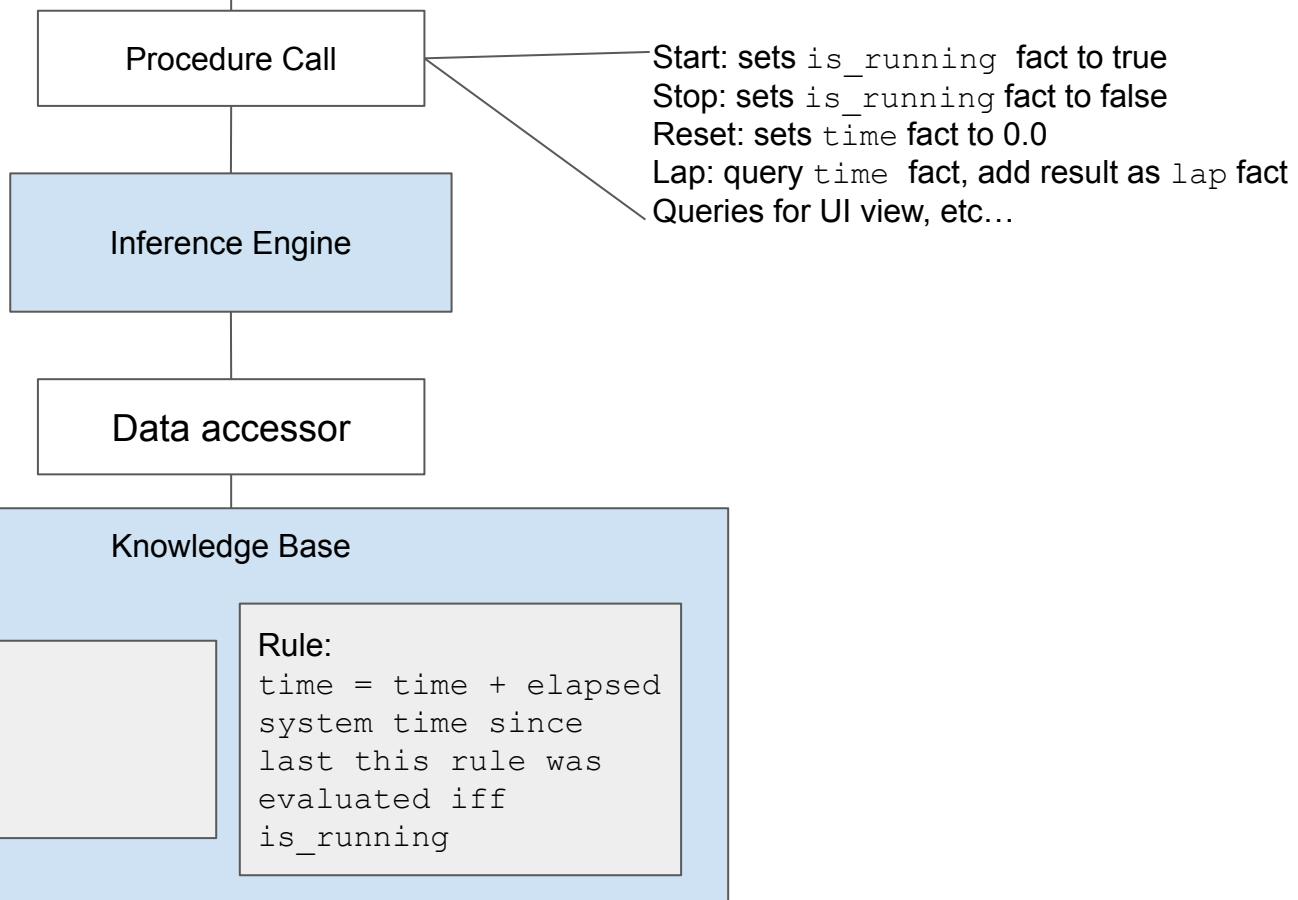
User Interface
(Digital/analog representation, start, stop, lap, and settings button)

Legend

Component

Subcomponent

Connector



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)

And

Task 2b: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

.....using a pencil and paper...

...be able to explain its meaning...

...and money is no object...

Take a picture of your design and paste it here (or you can directly create it on this slide). You may add another slide if needed. . (15 minutes)

Or

Task 2c: Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

...using whatever method you like...

...Cost is a concern. Make sure that the cost of implementing your design is reasonable... You may add another slide if needed. . (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Team 9

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes) **Emergency response system**

And

Task 2b: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

.....using a pencil and paper...

...be able to explain its meaning...

...and money is no object...

Take a picture of your design and paste it here (or you can directly create it on this slide). You may add another slide if needed. . (15 minutes)

Or

Task 2c: Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

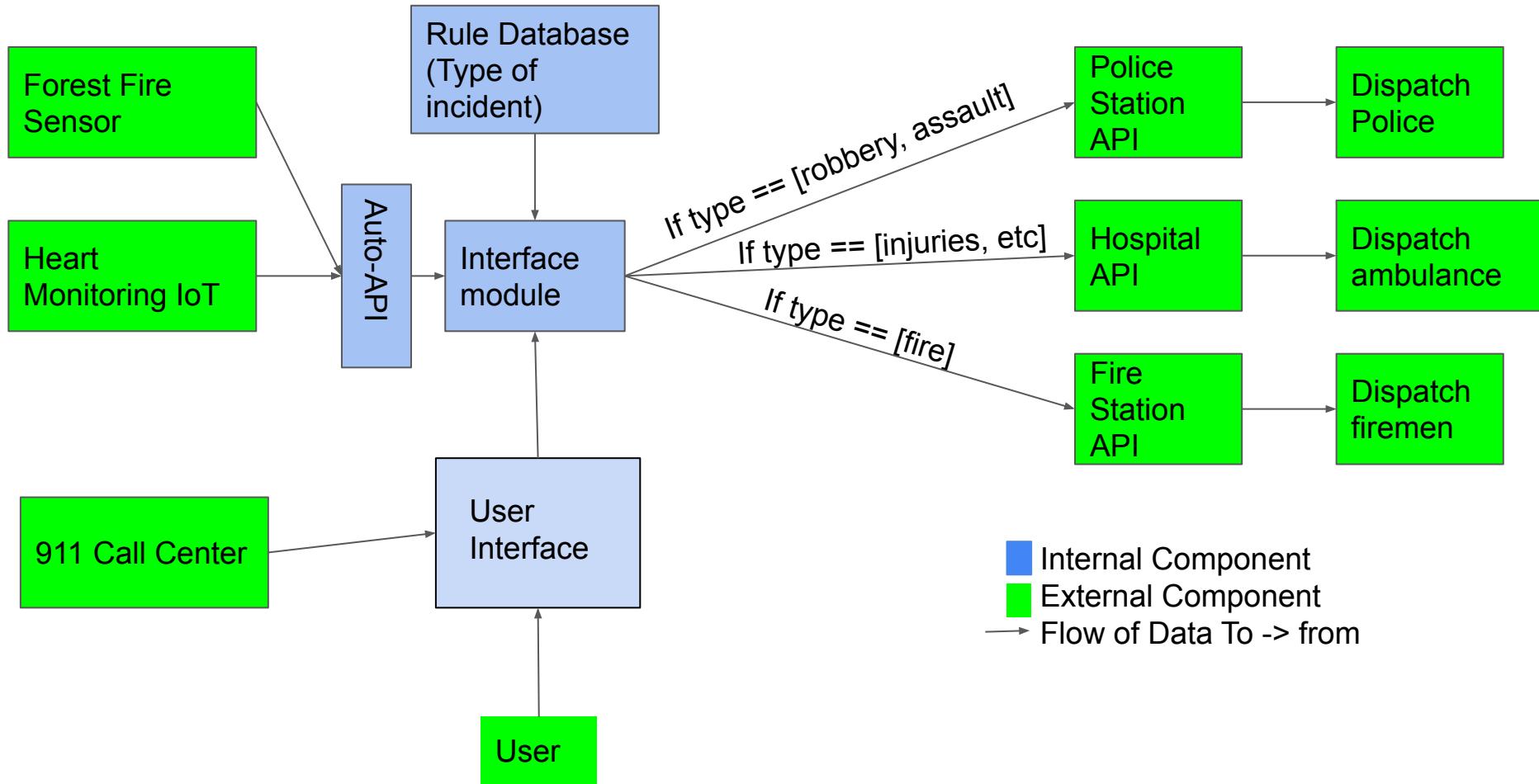
...using whatever method you like...

...Cost is a concern. Make sure that the cost of implementing your design is reasonable.. You may add another slide if needed. . (15 minutes)

List Roles and Student names

TimeKeeper: Pragati

Reporter: Will



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)

And

Task 2b: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

.....using a pencil and paper...

...be able to explain its meaning...

...and money is no object...

Take a picture of your design and paste it here (or you can directly create it on this slide). You may add another slide if needed. . (15 minutes)

Or

Task 2c: Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

...using whatever method you like...

...Cost is a concern. Make sure that the cost of implementing your design is reasonable... You may add another slide if needed. . (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Open Source Web Services

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a: Based on the system you are provided, choose the most appropriate architecture style. Sketch your architecture. **Do not** write your architecture style. We will ask the class if they recognize the style. You may add another slide if needed. (15 minutes)

And

Task 2b: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

.....using a pencil and paper...

...be able to explain its meaning...

...and money is no object...

Take a picture of your design and paste it here (or you can directly create it on this slide). You may add another slide if needed. . (15 minutes)

Or

Task 2c: Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

...using whatever method you like...

...Cost is a concern. Make sure that the cost of implementing your design is reasonable... You may add another slide if needed. . (15 minutes)

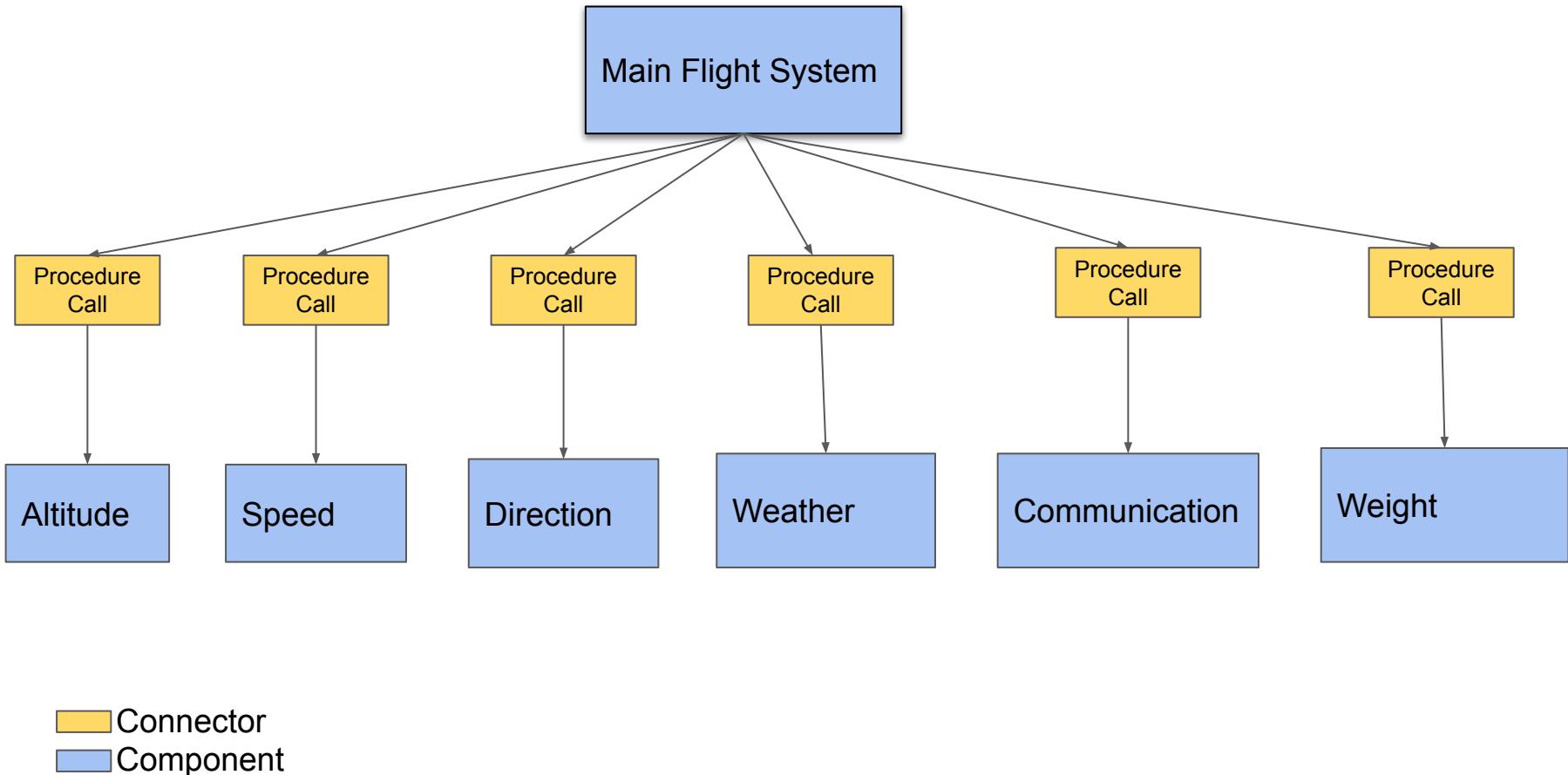
List Roles and Student names

Moderator : Yang Yu

TimeKeeper:

NoteTaker: Nur Dincer

Reporter: Sunjil Gahatraj





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



DESIGN PATTERNS

Portions taken and adapted from Gamma et. al. Design Patterns Elements of Reusable Object-Oriented Software, Pressman's Software Engineering, A Practitioner's Approach,

Common Patterns



- Common patterns are used throughout computer science and software engineering to solve macro and micro problems
- <http://hillside.net/patterns>

Different Types of Patterns



- Architectural Styles or Patterns
 - Covered previously
- Design Patterns
- Security Design Patterns
- UI Patterns
- Workflow Patterns
- ...and others

Different levels of Design

← →

- Analogy

Frame (gray parts) – arch style

Rooms – sub-arch style

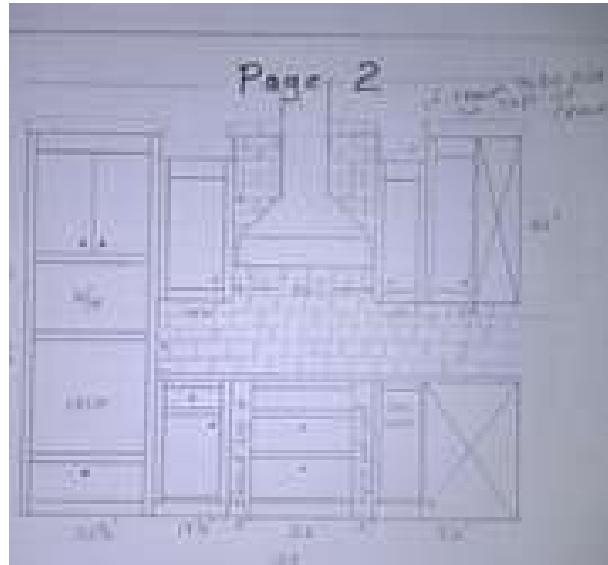


Different levels of Design



- Analogy

Cabinet Layouts – design patterns



Source: <https://creativecommons.org/>

Design Problem



- Need to use two existing classes, but their interfaces do not match

Design Pattern: Adapter



- Intent: convert the interfaces of a class into another interface clients expect
- Also known as wrapper



American electrical plug



“Wrappers”



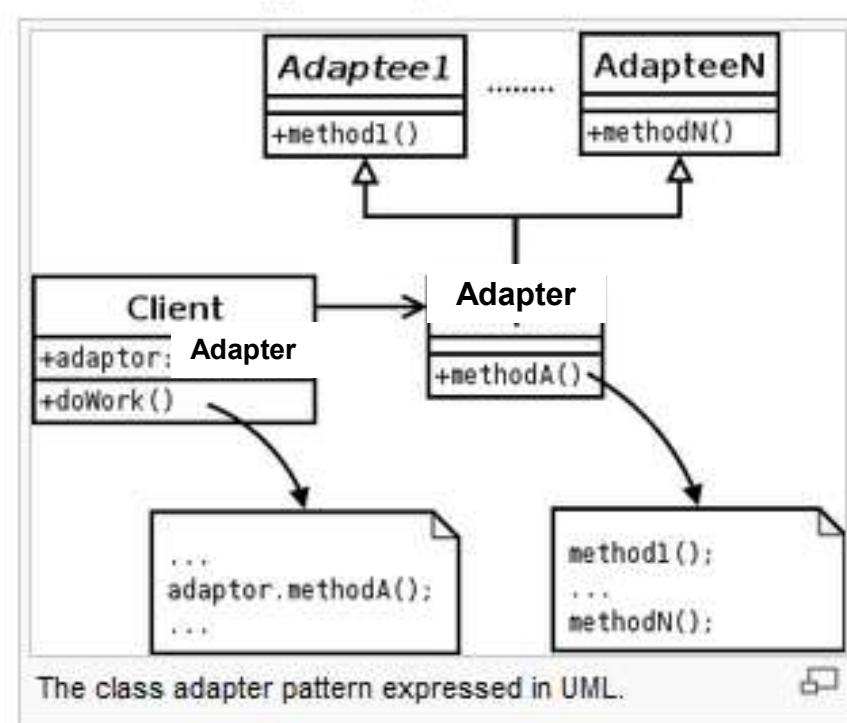
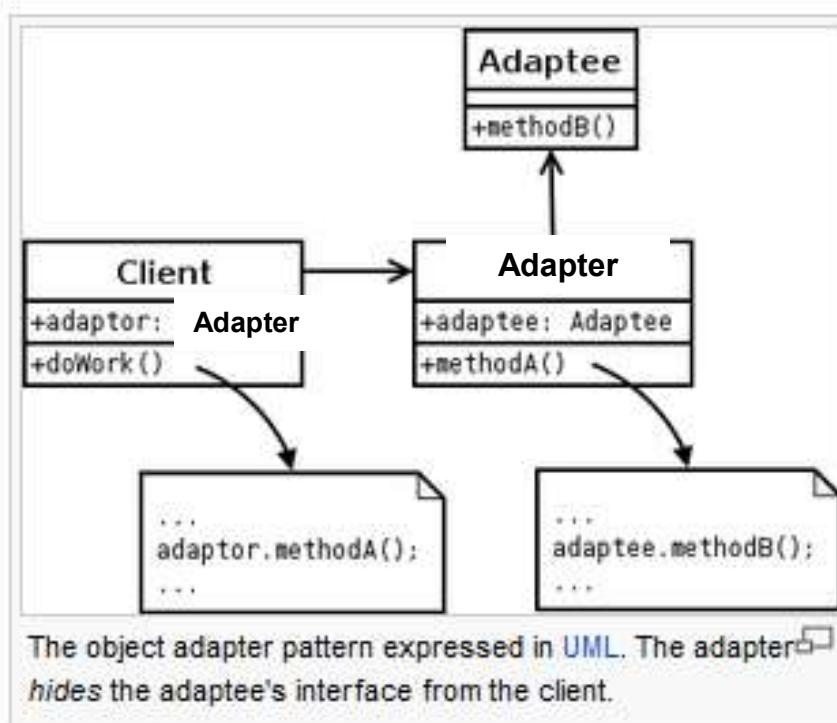
European electrical outlet

Design Pattern: Adapter



- Use when:
 - You want to use an existing class, and its interface does not match the one you need
 - You want to create a reusable class that cooperates with unrelated or unforeseen classes, that is, classes that don't necessarily have compatible interfaces

Design Pattern: Adapter



http://en.wikipedia.org/wiki/Adapter_pattern

Design Problem



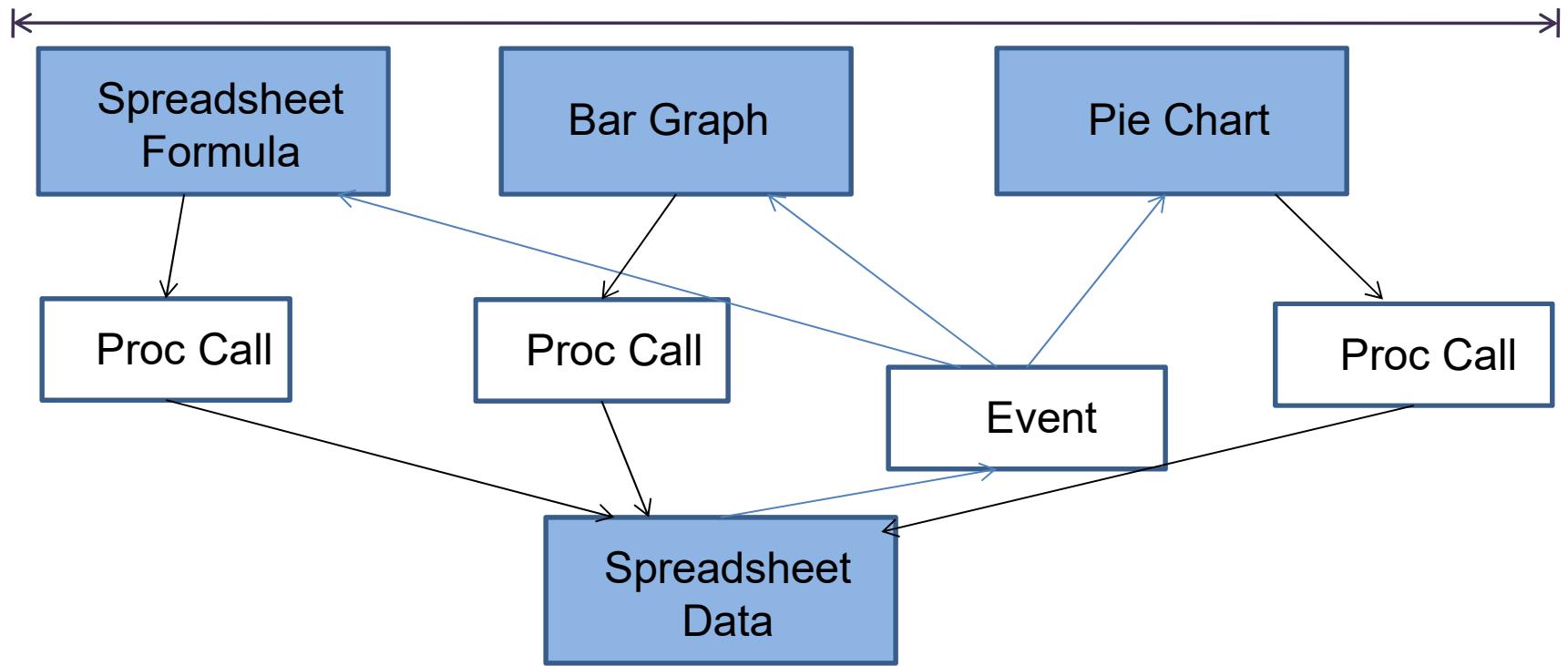
- Different components need to immediately respond to each state change
- Solution must be scalable

Design Pattern: Observer



- Intent: defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically
- Also known as dependents, publish-subscribe
- Use when:
 - Keep classes consistent without making the objects tightly coupled

Design Pattern: Observer



Subject	• Spreadsheet Data	Send notify signal to observer object whenever data changes
Observer	• Spreadsheet Formula • Bar Graph • Pie Char	Request subject for change information in order to update itself accordingly

<http://www.cs.clemson.edu/~malloy/courses/patterns/observer.html>

Design Problem



- Need to make sure that a class is only instantiated once

Design Pattern: Singleton



- Intent: ensures a class only has one instance, and provide a global point of access to it
- Use when:
 - Important for some classes to exactly have one instance (e.g., one printer spooler, one operating systems)

Design Pattern: Singleton



Sample code:

```
public class Singleton {  
    private static final Singleton instance = new Singleton();  
  
    // Private constructor prevents instantiation from other classes  
    private Singleton() {}  
  
    public static Singleton getInstance() {  
        return instance;  
    }  
}
```

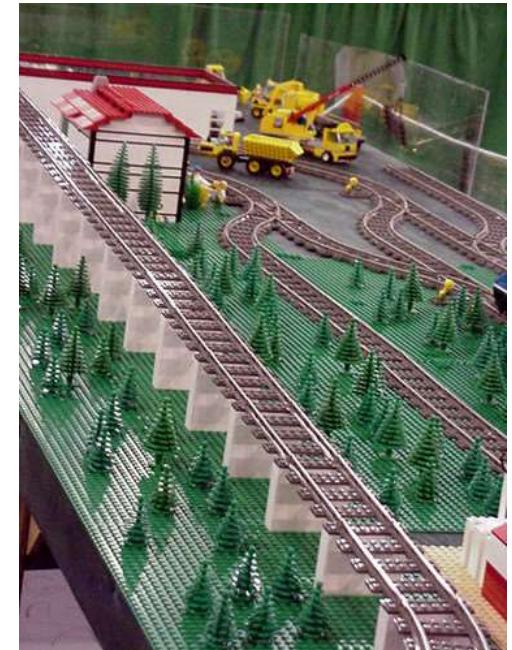
http://en.wikipedia.org/wiki/Singleton_pattern



Design Problem



- Create a complex object that may have different representations depending on how they are assembled



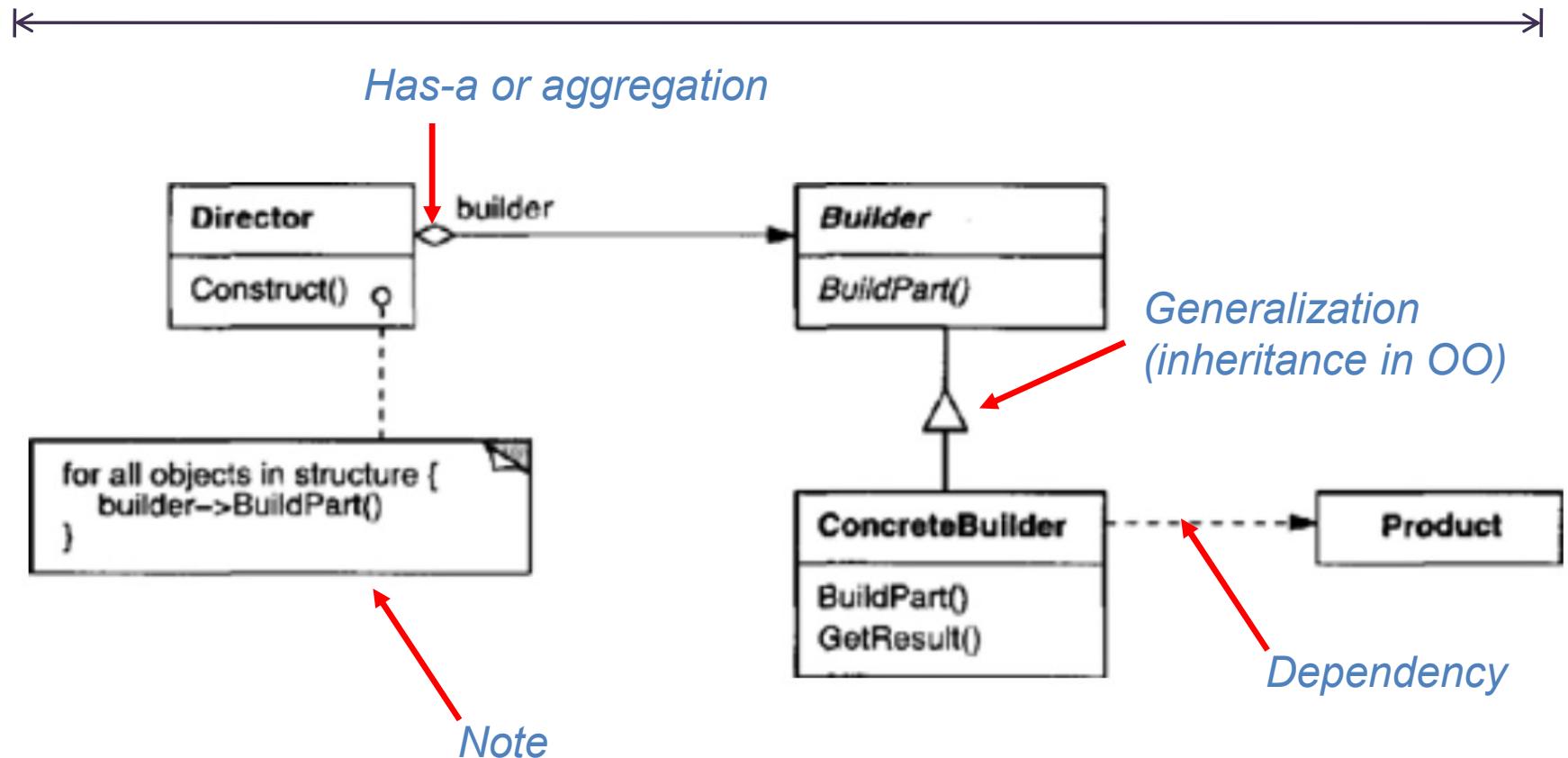
Source: <https://creativecommons.org/>

Design Pattern: Builder



- Motivation: A file reader should be able to convert to many different file formats, but new file formats might come up in the future
- Use when:
 - The algorithm for creating a complex object should be independent of the parts that make up the object and how they're assembled
 - Construction process allow for different representations for the object that's constructed

Design Pattern: Builder



Design Pattern: Builder



Builder

```
class MazeBuilder {  
public:  
    virtual void BuildMaze() { }  
    virtual void BuildRoom(int room) { }  
    virtual void BuildDoor(int roomFrom, int roomTo) { }  
  
    virtual Maze* GetMaze() { return 0; }  
protected:  
    MazeBuilder();  
};
```

Product

```
Maze* MazeGame::CreateMaze (MazeBuilder& builder) {  
    builder.BuildMaze();  
  
    builder.BuildRoom(1);  
    builder.BuildRoom(2);  
    builder.BuildDoor(1, 2);  
  
    return builder.GetMaze();  
}
```

Director

Design Problem



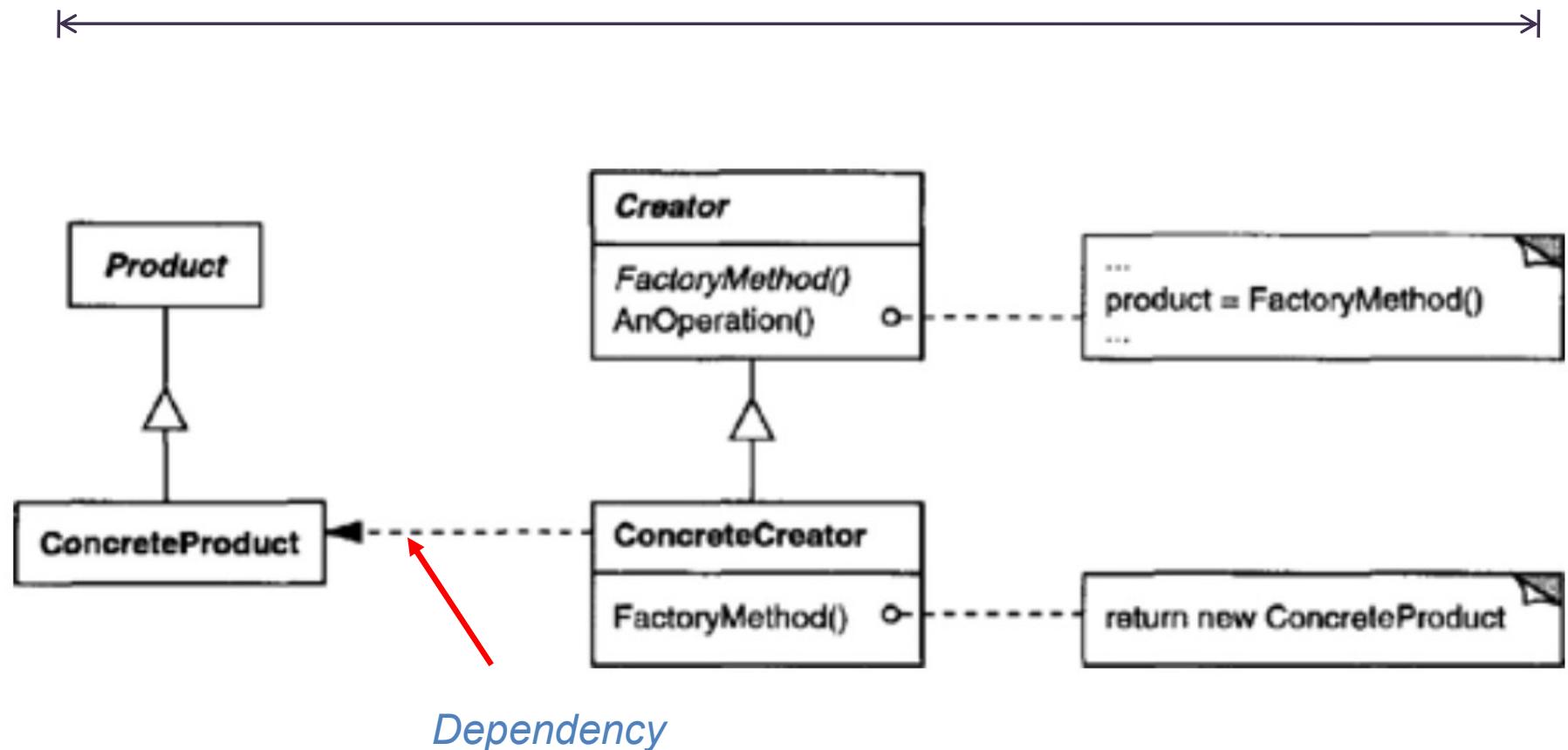
- Need to create objects, but the type of objects to create is unknown or yet to be determined

Design Pattern: Factory Method



- Also known as Virtual Constructor
- Intent: Define an interface for creating an object, but let subclasses decide which class to instantiate
- Use when:
 - A Class can't anticipate the class of objects it must create
 - A class wants its subclasses to specify the objects it creates
 - Classes delegate responsibility to one of the several helper subclasses, and want to localize the knowledge of which helper subclass is the delegate

Design Pattern: Factory Method



Design Problem



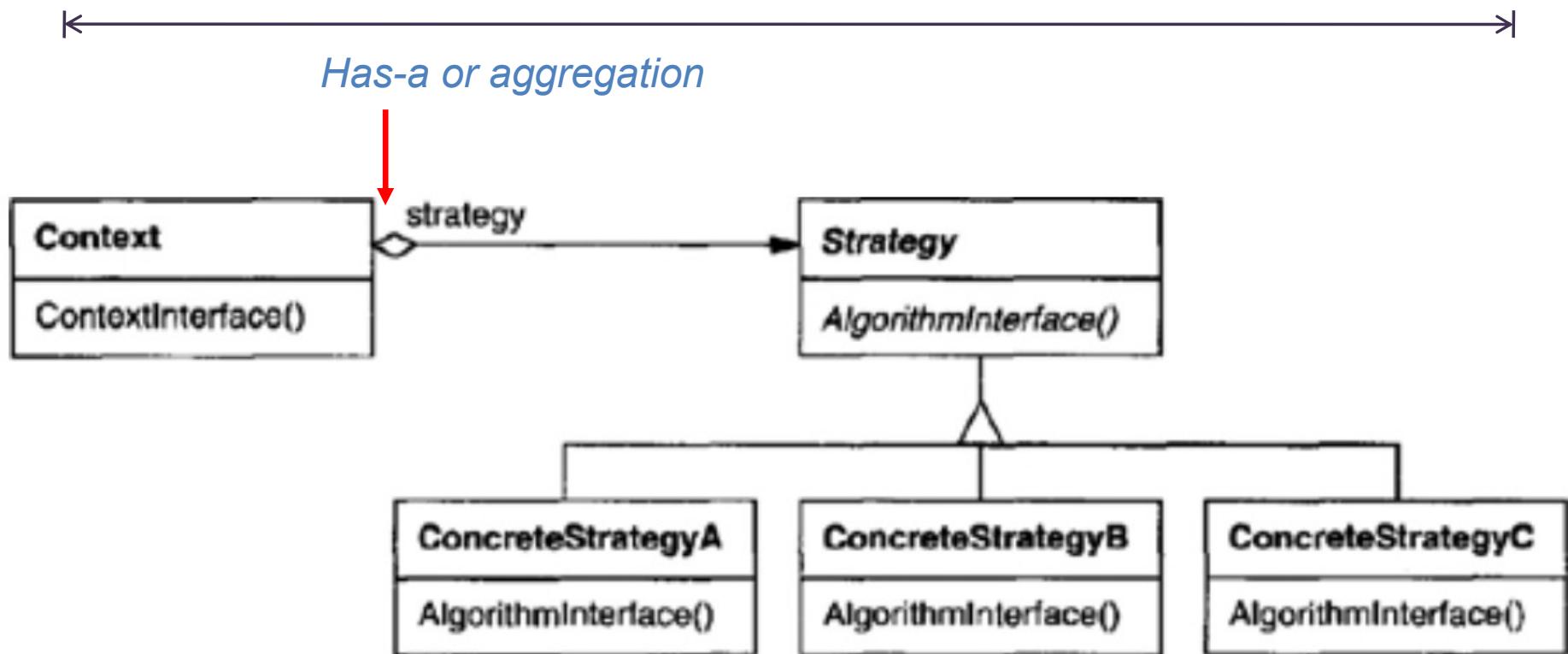
- Need to create a set of algorithms that can be interchangeably used, depending on the situation

Design Pattern: Strategy

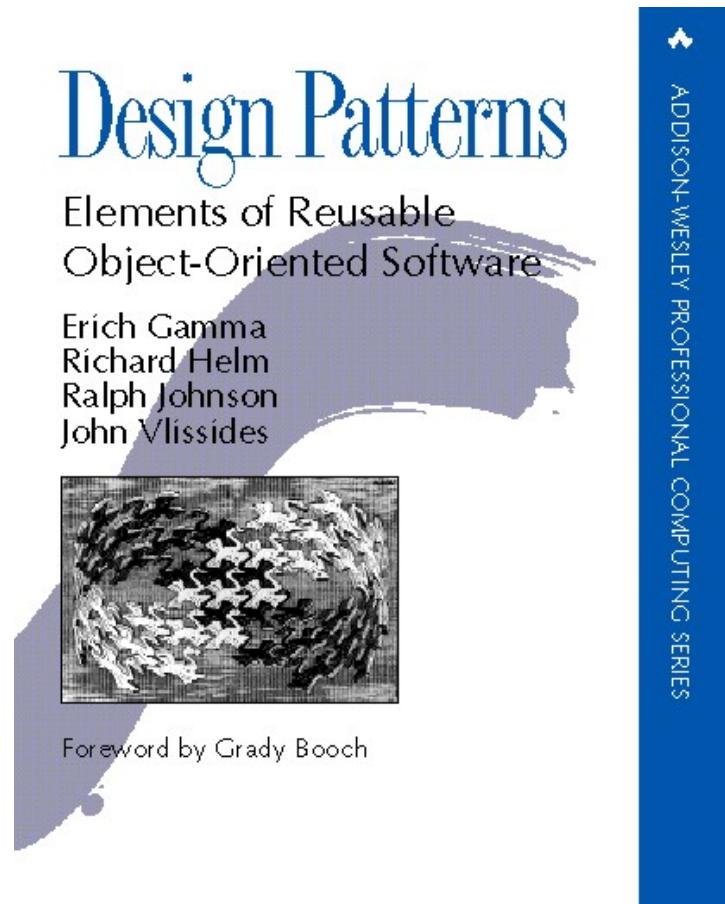


- Also known as Policy
- Motivation: Many algorithms exist for breaking a stream of text into lines. Hard-coding algorithms into code that use them isn't desirable
- Use when:
 - Many related classes differ only in their behavior
 - Need different variants of an algorithm
 - Avoid exposing complex, algorithm-specific data structures
 - A class define many behaviors and appear as multiple conditional statements. Move those conditional branches into their own Strategy class

Design Pattern: Strategy



Design Patterns Book





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



SECURITY DESIGN PATTERNS

Portions taken and adapted from Dougherty, Chad et al, Secure Design Patterns, Technical Report CMU/SEI-2009-TR-010, Software Engineering Institute

Cybersecurity Vocabulary



- Asset - a resource of value (data, service, etc.)
- Threat - a potential for compromise/damage
- Vulnerability - a weakness that makes the threat possible
- Attack/Exploit - an action that damages an asset
- Countermeasure - addresses a threat or mitigates risk

Cybersecurity Background: Threat Classes



- Threats to confidentiality
 - Disclose information to people/programs who have no authorization to access info
- Threats to integrity
 - Damage or corrupt the software/data
- Threats to availability
 - Restrict access to system or data for authorized users

Cybersecurity Background



- Security vulnerabilities
 - <http://cwe.mitre.org/top25>
 - <https://cve.mitre.org/>

Security Design Patterns



- Descriptions or templates describing general solution to a security problem that can be applied in many different situations

Security Design Patterns



- Design patterns – generally focus on object-oriented style
- Security Patterns – not limited to this style
 - There are many security patterns that are not at the “Design level”
 - Specific implementation
 - See [Implementation level patterns](#)

Design Problem



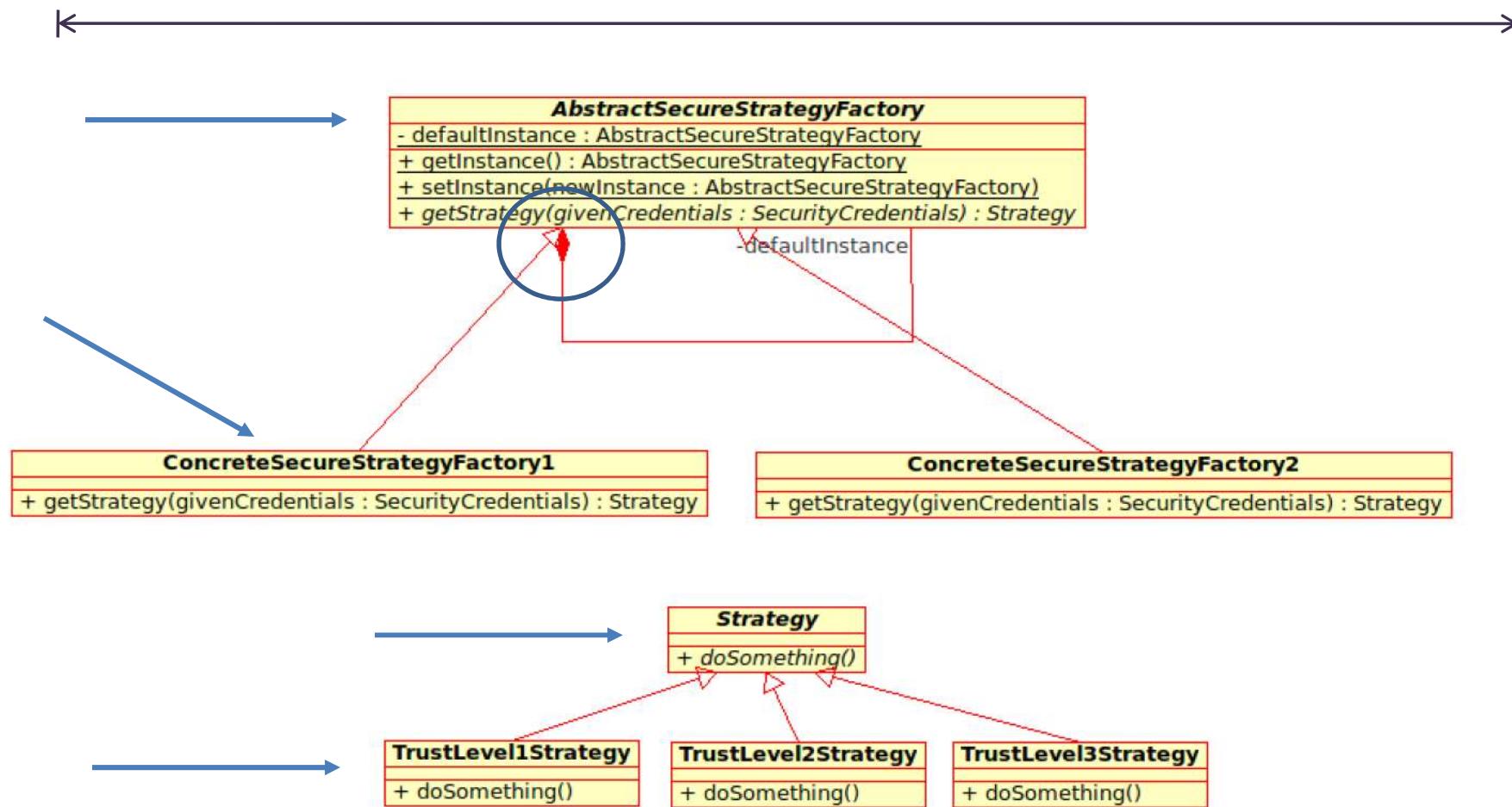
- Run a specific task or algorithm based on the credentials of a user or environment

Design Pattern: Secure Strategy Factory



- Intent: Operates as follows:
 - Caller asks for implementation of Secure Strategy Factory pattern to perform a function given security credentials
 - Secure Strategy Factory implementation uses credentials to select the appropriate object implementing the Strategy pattern
- Use when:
 - Similar to Strategy pattern but the variation of algorithm is based on security credentials

Design Pattern: Secure Strategy Factory



Design Problem



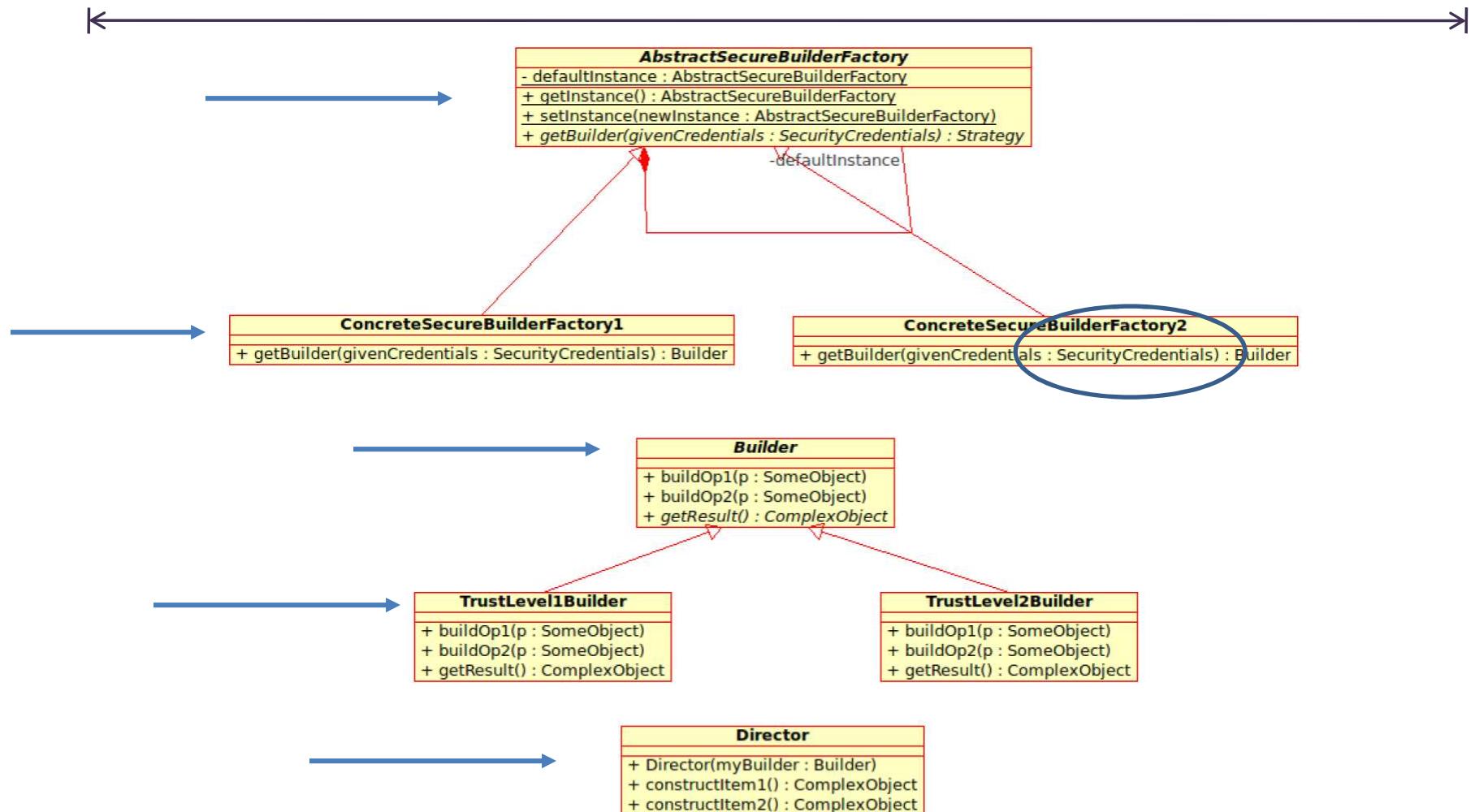
- Separate security dependent rules in creating a complex object from the steps in actually creating the object

Design Pattern: Secure Builder Factory



- Motivation: Use complex objects whose allowable contents are dictated by the level of trust in a user or environment (e.g., allowable queries to a database may be dictated by trust level)
- Use when:
 - Similar to Builder pattern
 - ... but the behavior of the Builder depends on security credentials or
 - ... construction of complex object is based on security credentials

Design Pattern: Secure Builder Factory







Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



COMMON ARCHITECTURAL STYLES

Portions taken and adapted from Richard N. Taylor (UCI)

Common Architectural Styles



- Implicit Invocation
 - Publish-subscribe
 - Event-Based
- Peer to peer

Implicit Invocation Style



- Event announcement instead of method invocation
 - “Listeners” register interest in and associate methods with events
 - System invokes all registered methods implicitly
- Component interfaces are methods and events
- Style invariants
 - “Announcers” are unaware of their events’ effects
 - No assumption about processing in response to events

Implicit Invocation



- Advantages
 - More scalable
 - Component reuse
 - System evolution
 - Both at system construction-time & run-time
- Disadvantages
 - Counter-intuitive system structure
 - Components relinquish computation control to the system
 - No knowledge of what components will respond to event
 - No knowledge of order of responses

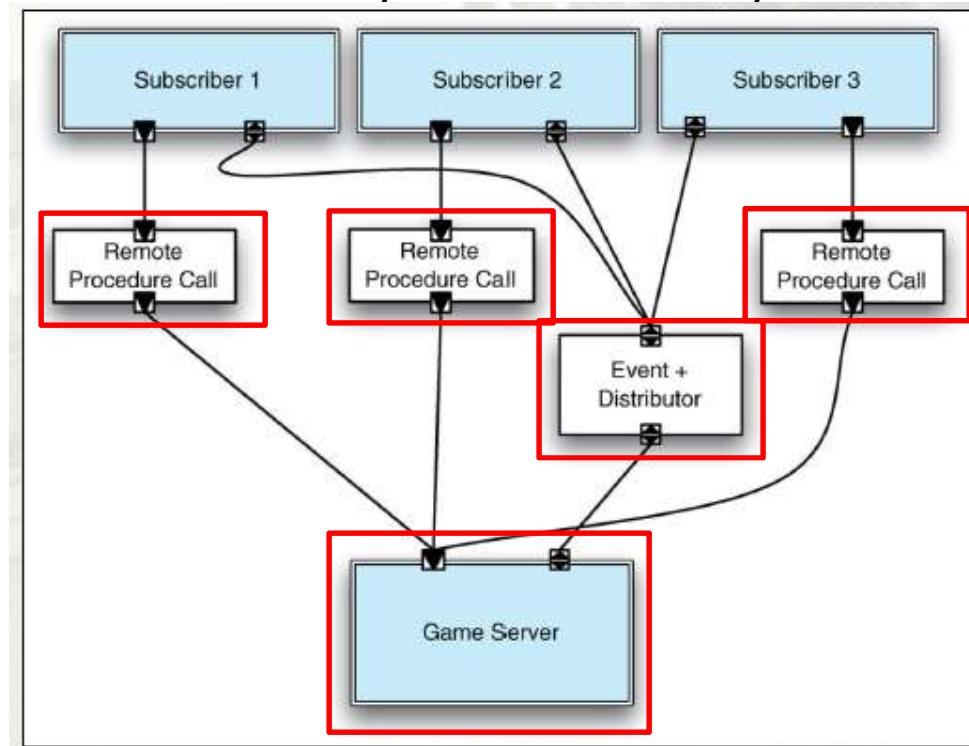
Publish-Subscribe

Subscribers register/deregister to receive specific messages or specific content. Publishers broadcast messages to subscribers either synchronously or asynchronously.

Components?

Connectors?

Data elements?



Publish-Subscribe



- Topology: Subscribers connect to publishers either directly or may receive notifications via a network protocol from intermediaries
- Quality yielded: Highly efficient one-way dissemination of information with very low-coupling of components



Publish-Subscribe



- Typical uses
 - News dissemination, GUI programming, multi-player network-based games
- Caveat
 - Very large number of subscribers for a single data item, may need specialized broadcast protocol

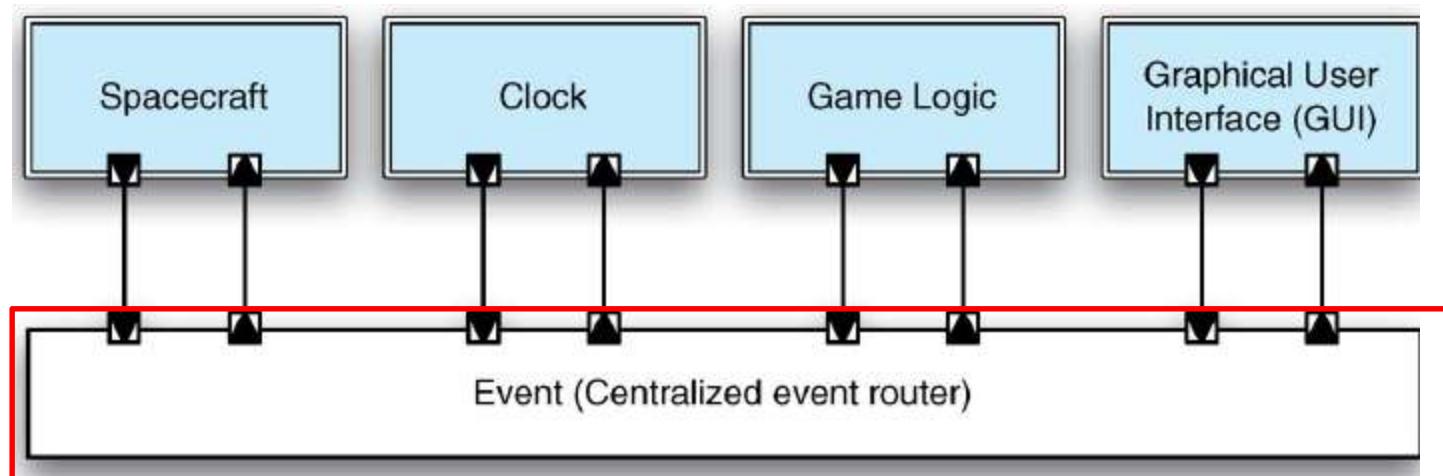
Publish-Subscribe Example



- Google Cloud Pub-Sub
 - Messaging service – may be used to send and receive messages between independent applications anywhere in the world
 - Servers run in multiple data centers
 - <https://cloud.google.com/pubsub/architecture>

Event-Based Style

- Independent components asynchronously emit and receive events communicated over event buses
- Components?
- Connectors?
- Data elements?



Event-Based Style



- Topology: Components communicate with the event buses, not directly to each other.
- Variants: Component communication with the event bus may either be push- or pull-based.
- Highly scalable, easy to evolve, effective for highly distributed applications.

Event-based Style

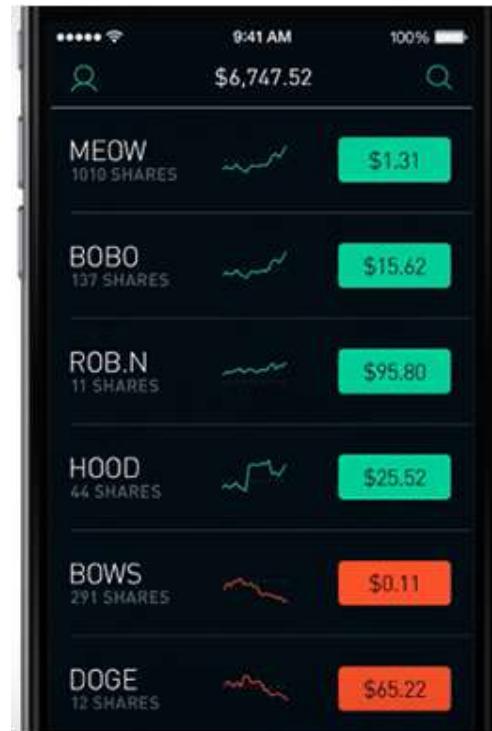


- Typical uses
 - User interface software, wide area applications involving independent parties (e.g. financial markets, sensor network)
- Caveat
 - No guarantees if or when an event will be processed

Event-based Example



- Stock Trading
 - Receive notifications
 - Send notifications

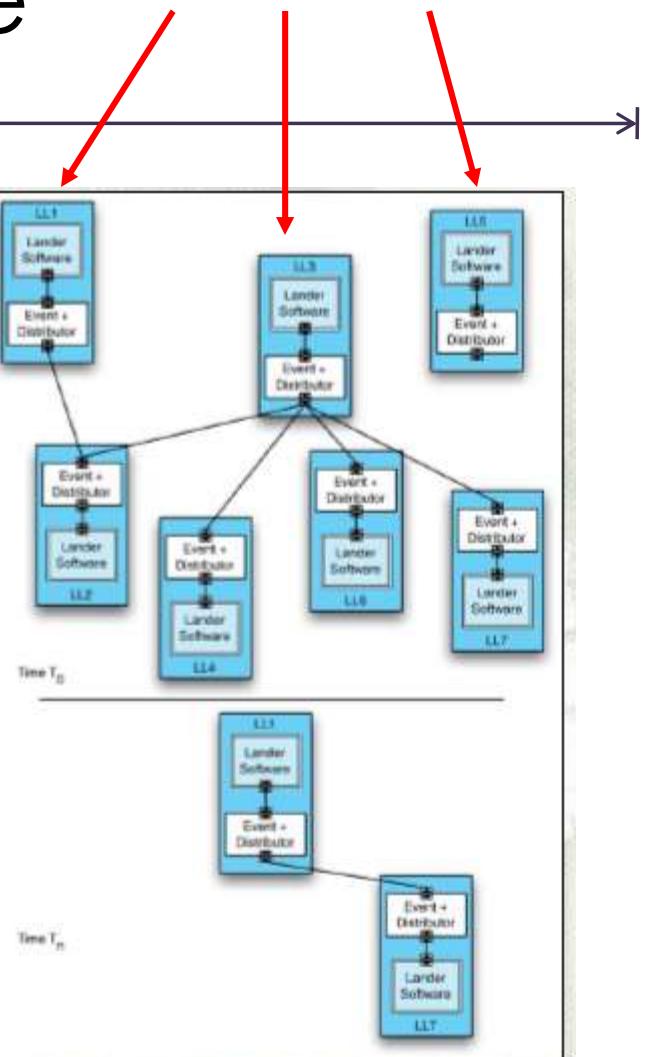


<http://cdn.blessthisstuff.com/imagens/stuff/robinhood-stock-trading-app-3.jpg>

Peer-to-Peer (P2P) Style



- State and behavior are distributed among peers which can act as either clients or servers.
- Supports decentralized computing
- Components?
- Connectors?
- Data elements?



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Peer-to-Peer Style



- Topology: Network (may have redundant connections between peers); can vary arbitrarily and dynamically
- Qualities:
 - decentralized computing
 - highly robust in the face of failures
 - scalable

Peer-to-Peer Style

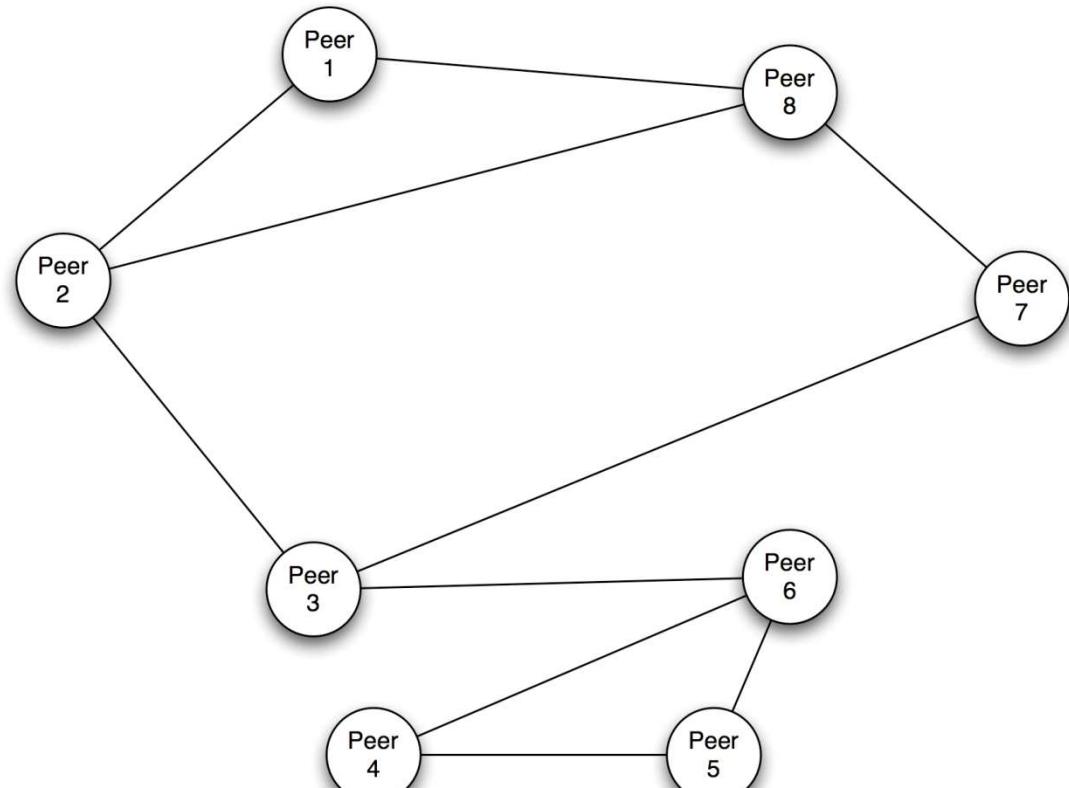


- Typical uses
 - Contexts where sources of information and operation are distributed and network is ad-hoc
- Caveat
 - Latency imposed by protocol
 - Need to manage security

Peer-to-Peer Example



- Gnutella
 - File-sharing protocol



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Questions



- Give one example of a system that uses P2P style. In this system, state one important system qualities (or non-functional property). How does P2P support this quality/property?
- Compare/contrast architecture patterns with design patterns
- What are key differences between event-based systems and publish-subscribe?
- Are security patterns available only at the design-level? Why or why not?
- A co-worker tells you “Design patterns are all we need when designing software”. Do you agree with this statement? Why or why not?



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Agenda



- Review / Questions
- Quiz answers
- Reminders
- Design Exercise from last week

Questions



- Compare/contrast Factory Pattern with Builder pattern **Dawn**
- Give one example of a system that uses P2P style. In this system, state one important system qualities (or non-functional property). How does P2P support this quality/property? **The Boffins, Team 4**
- Are security patterns available only at the design-level? Why or why not? **To be decided**
- Compare/contrast architecture patterns with design patterns **Team 9**
- What are key differences between event-based systems and publish-subscribe? **Ludus, Open source Web services**
- A co-worker tells you “Design patterns are all we need when designing software”. Do you agree with this statement? Why or why not? **Magratheans**

Questions



- Compare/contrast Factory Pattern with Builder pattern **Dawn**
 - Factory pattern: subclasses define the object, good when don't know what class is needed in the future (from top to bottom)
 - Builder: build a complex object by putting together different parts (like Lego pieces) (from bottom to top)
- Give one example of a system that uses P2P style. In this system, state one important system qualities (or non-functional property). How does P2P support this quality/property? **The Boffins, Team 4**
 - Multi-player games – allows global scalability, even though don't have powerful server
 - Distributed file system – robust, preserving several copies of same data
- Are security patterns available only at the design-level? Why or why not? **To be decided**
 - no, not only at design-level – entire software development lifecycle-implementation
 - Check the CMU reading

Questions



- Compare/contrast architecture patterns with design patterns

Team 9

- Architecture patterns- higher level (overall system)
 - Design patterns – about specific component in an architecture (or a subsystem)

- What are key differences between event-based systems and publish-subscribe? **Ludus, Open source Web services**

- Pub-sub: need to have subscriber to listen to publishers
 - Event-based – everyone sends message
 - Event-based – decentralized, distributed among several parties
 - Pub-sub – designed around a set of topics, dependent on publisher

- A co-worker tells you “Design patterns are all we need when designing software”. Do you agree with this statement? Why or why not? **Magratheans**

- No, depends on how complex is the system
 - Style – more abstract, can help organize different patterns (theme of a show)

Design patterns – specific implementation problems (individual episode / show))

CSS 553, Spring 2023, 4d Activities

Qualities and Arch styles



- Make a note of qualities provided in each arch style...
 - Midterm: what arch styles satisfy following qualities? Performance, scalability, etc...

Arch Style vs Arch Pattern



- Treat them the same for the purpose of the class, in application to real life

Announcements



- Mid-course survey
- G2 – next week
- Survey – Time schedule for CSS553 next week

QUIZ ANSWERS

ACTIVITIES

Design Exercise #2a



- Suppose we are to give out an award for excellence in MS project. Create a design for this award...
- ...using a pencil and paper...
- ...be able to explain its meaning...
- ...and money is no object...
- Take a picture of your design and submit the google slides

Reflection Questions



- How hard was it to think of the design for the award?
- How hard was it to create the design for the award?
- Which representation did you use for your design, and why?
- How does your design reflect its meaning?
- How do you think the recipient will react to the award's design?

Design Exercise #2b



- Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...
- ...using whatever method you like...
- ...Cost is a concern. Make sure that the cost of implementing your design is reasonable...
- Provide your answer in the Google Slide

Reflection Questions



- How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?
- Who did you keep in mind when making your design?
- What was your goal with your design?
- Did you have more than one goal?
- Did you reach the goal(s) with your design?



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



Outline



- Architecture to Implementation
- 1.x way mapping



Part I

ARCHITECTURE TO IMPLEMENTATION MAPPING

Slides taken and adapted from Yongjie Zheng's PhD Thesis, "Enhancing Architecture-Implementation Conformance with Change Management and Support for Behavioral Mapping", UC Irvine, 2012
CSS 553, Spring 2023, 5b Architecture
Implementation Mapping



Implementation



- You have designed an architecture for an application
 - What do you call this architecture?
 - *prescriptive architecture*
- Now, you have to go implement the system
 - Assumptions during design will be tested
 - Your architecture may end up slightly different—a common phenomenon:
 - *descriptive architecture* != *prescriptive architecture*

Implementation Challenges

- ↔
- Although your architectures are going to be different, they have some commonalities
 - Components
 - Interconnections (links, possibly explicit connectors)
 - Explicit provided and required interfaces
 - Your target platform (e.g., Java) is mismatched
 - Objects
 - Pointers/references everywhere
 - Optional provided interfaces
 - How do we bridge the gap?

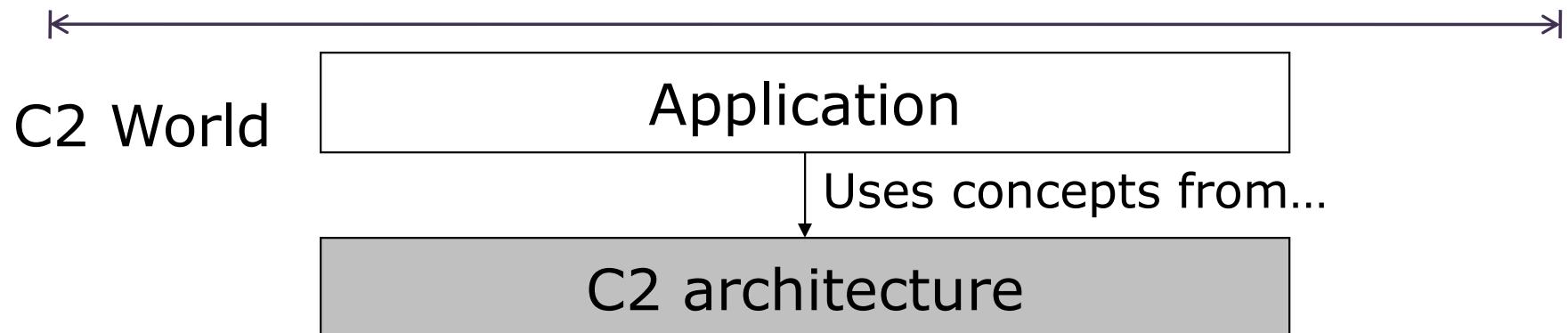


An Architecture Framework



- An architecture framework is software that bridges the gap between the concepts of an architectural style and the capabilities of a given platform (programming language + operating system + runtime)

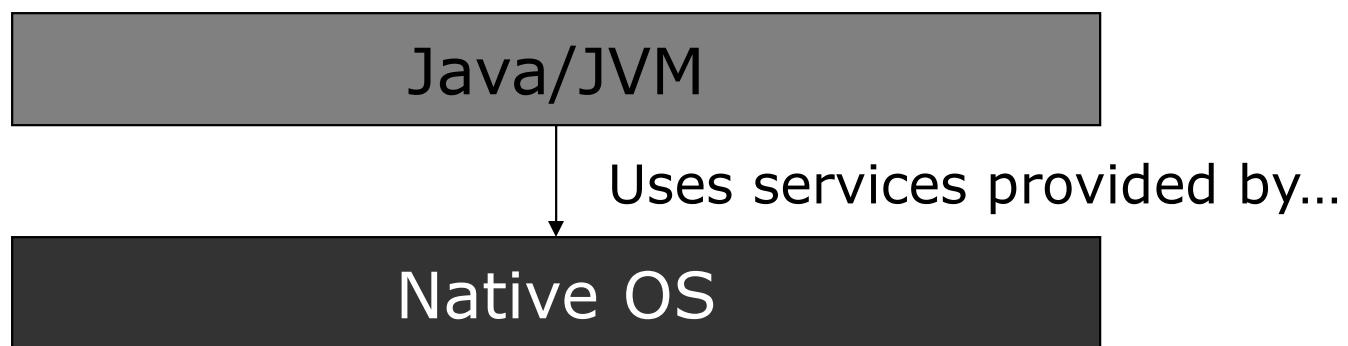
Implementing a C2 architecture



???

How do we bridge this gap?

Object-
oriented
World

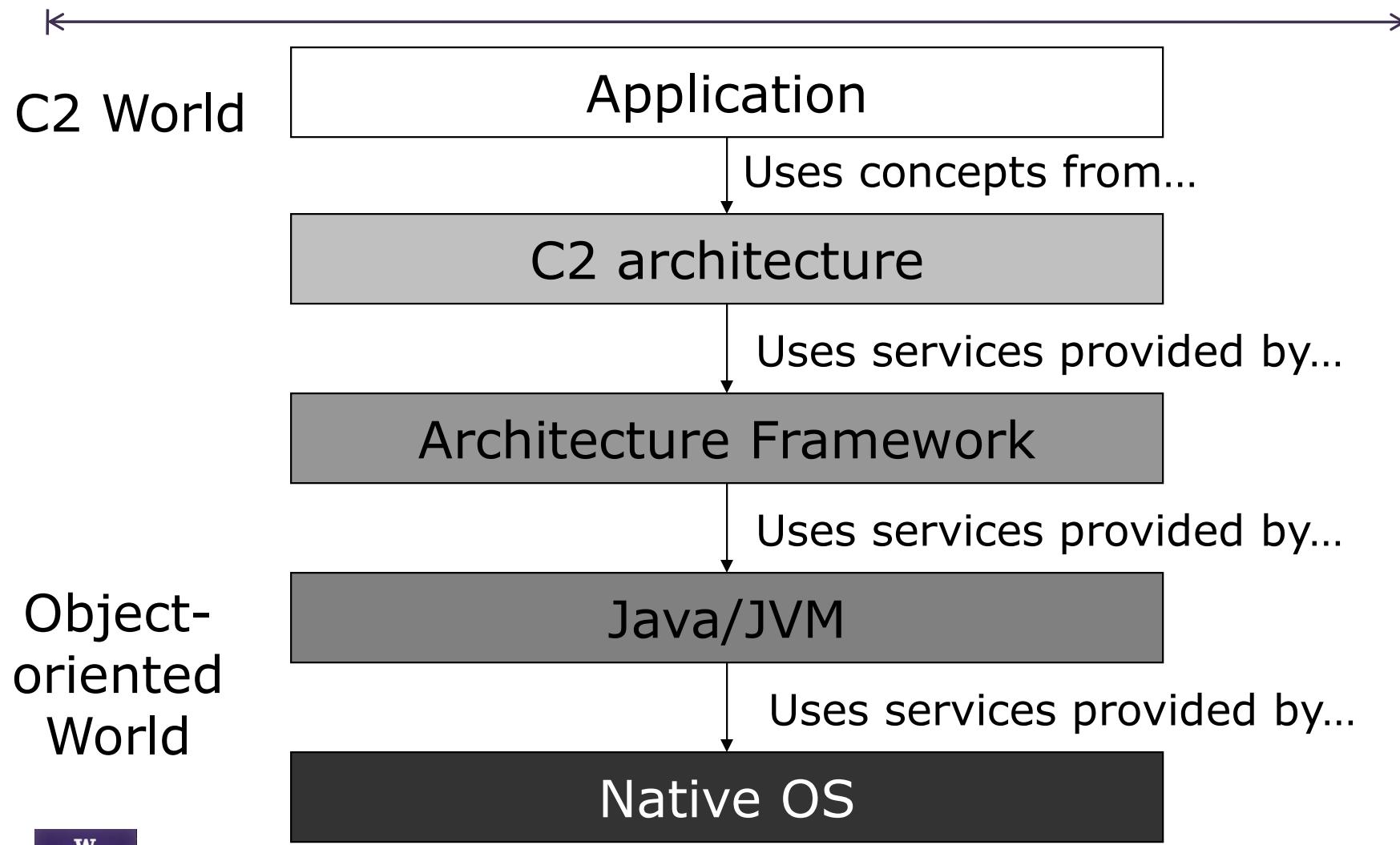


Example



- C2 World
 - Components
 - Connectors
 - Events
 - Links
 - Many threads
 - Asynchronous communication
- Java World
 - Objects
 - Method calls
 - Parameters
 - References
 - Few threads
 - Synchronous communication

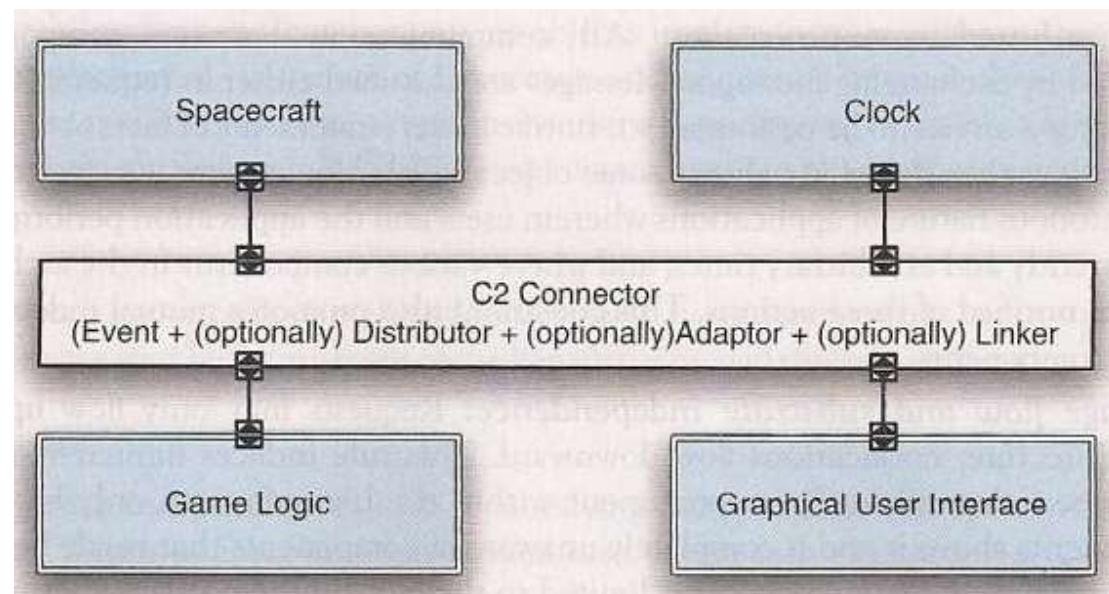
Implementing a C2 architecture



C2 Style



- Autonomous, event-based communication
- Hierarchical applications
 - Notifications fall
 - Requests rise



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Enter c2.framework



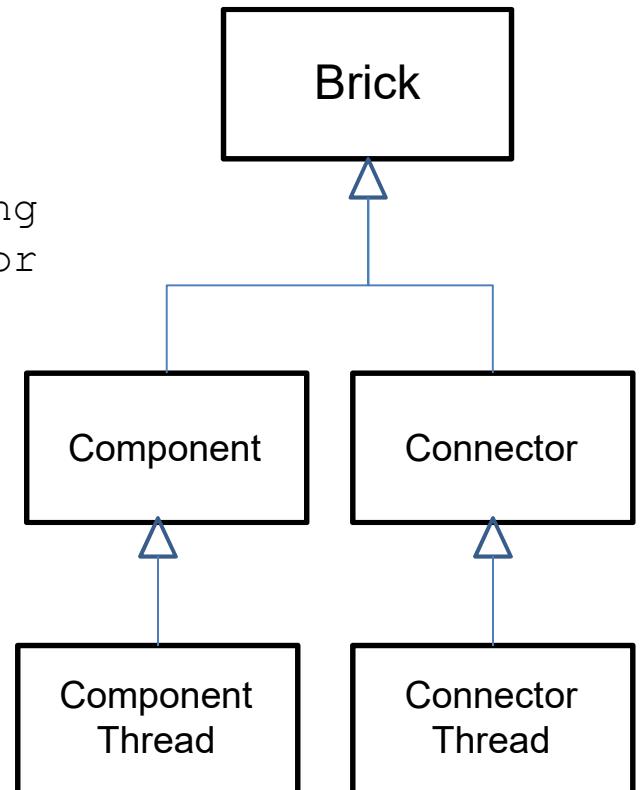
- c2.framework is an architecture framework for the C2 style built in Java, providing:
 - Abstract base classes for components, connectors
 - Reusable connectors (local & network)
 - (Pluggable) topology management
 - (Pluggable) threading policies
- Essentially, c2.framework does the hard work, components and connectors simply implement behaviors.

Lifecycle Methods of Brick

```
public void create() {
    //instantiates the object
}

public void start() {
    //called when component/connector is running
    //should be called before component sends or
    receives messages
}

public void finish() {
    //immediately stops the execution of the
    component or connector
    //even if it has outstanding messages to
    process
}
```



Your basic c2.framework component...

```
package edu.uci.ics.mypackage;

//standard import
import c2.framework.*;

int altitude = 0;
int fuel = 0; //plus other internal state variables
```

```
public class GameLogic extends ComponentThread {  
}
```

Implements lots of boilerplate functionality

Your basic c2.framework component...

```
← →  
package edu.uci.ics.mypackage;  
  
//standard import  
import c2.framework.*;  
  
int altitude = 0;  
int fuel = 0;           //plus other internal state variables  
  
public class GameLogic extends ComponentThread {  
    public GameLogic() {  
        super.create("gameLogic", FIFOPort.class)  
    }  
  
    public void start() {  
        super.start();  
        Request r = new Request("getGameState");  
        send(r);  
    }  
}
```



Your basic c2.framework component...

```
public class GameLogic extends ComponentThread {  
    :  
    protected void handle(Notification n) {  
        if (n.name().equals("gameState")) {  
            if(n.hasParameter("altitude")) {  
                this.altitude = ((Integer)n.getParameter("altitude"))  
                    .intValue();  
            }  
            if(n.hasParameter("fuel")) {  
                this.fuel = ((Integer)n.getParameter("fuel"))  
                    .intValue();  
            }  
        :  
        :  
    }
```



Your basic c2.framework component...

```
public class GameLogic extends ComponentThread {  
    :  
    protected void handle(Request r) {  
        //component does not handle requests  
    }  
  
}
```



Pointers



- C2 Framework
 - https://isr.uci.edu/architecture/JavaFrameworkDoc/c2.framework.Brick.html#_top
- Lunar Lander in C2
 - <http://www.softwarearchitecturebook.com/svn/main/code/lander-c2.zip>

Part II

1.X WAY MAPPING

Slides taken and adapted from Yongjie Zheng's PhD Thesis, "Enhancing Architecture-Implementation Conformance with Change Management and Support for Behavioral Mapping", UC Irvine, 2012
CSS 553, Spring 2023, 5b Architecture
Implementation Mapping

1.x-way mapping

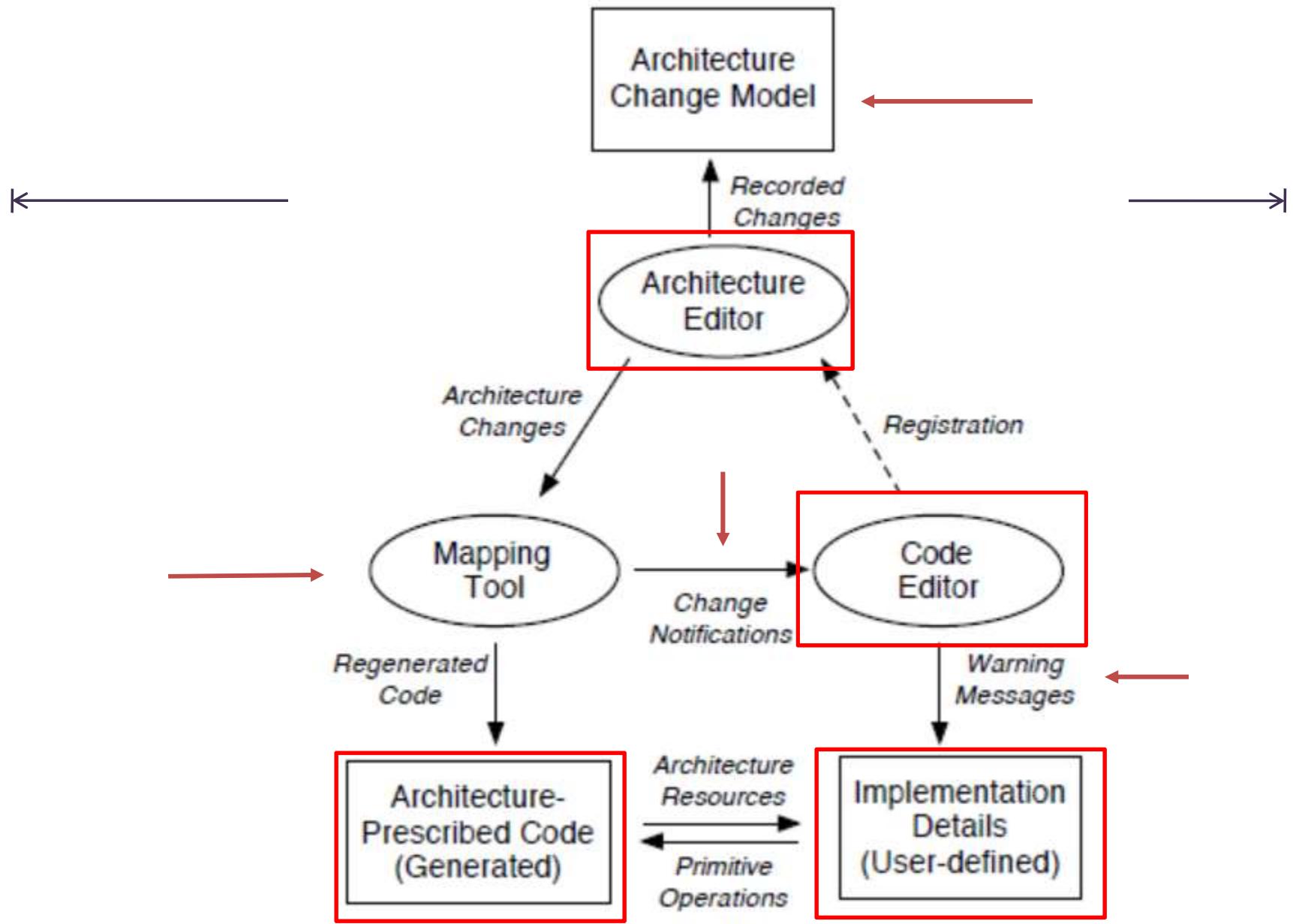


- Manual changes initiated in the architecture (“1”) and a separated portion of the code (“.x”)
- Tool: xMapper

Core mechanisms



- Deep separation
- Architecture change model
- Architecture-based code generation
- Architecture change notification



Video Demo



- <https://www.youtube.com/watch?v=J2ikbq76oAg>

Questions



- Compare C2 with **one** previous architecture styles/patterns covered in class. What are similarities with that style? Differences?
- How can one address the issue of latency with C2?
- What types of applications can benefit from using C2?
- Concepts in a programming language are often different from architecture concepts. How do you bridge this gap?
- Explain how components communicate with each other in C2



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



C2 STYLE

Portions taken and adapted from Richard N. Taylor and Eric Dashofy (UCI)

Heterogeneous Styles



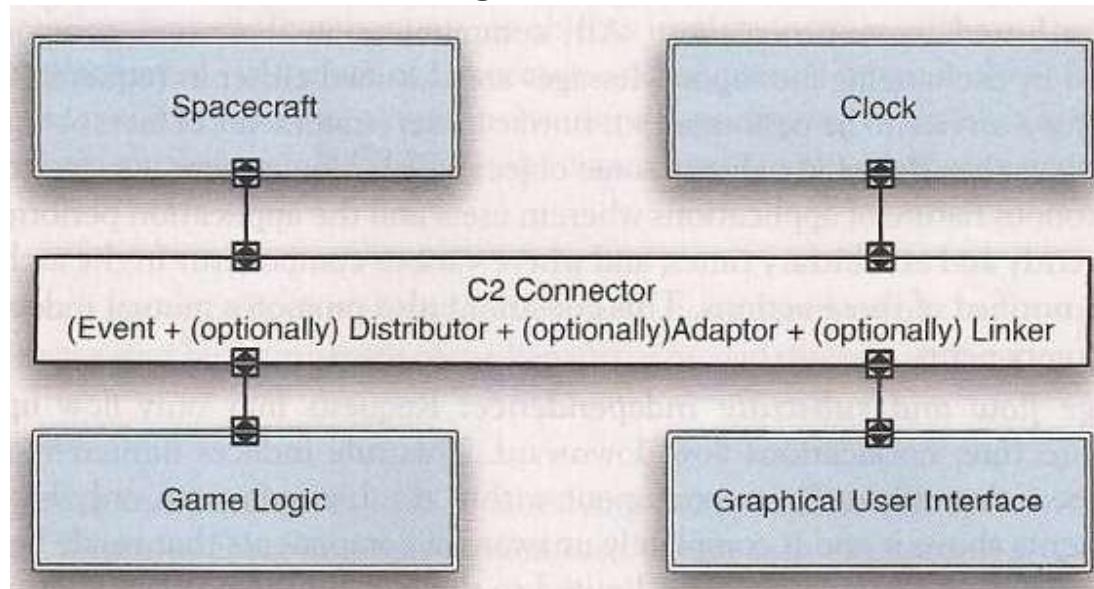
- More complex styles created through composition of simpler styles
- REST (discuss later)
 - Complex history presented later in course
- C2
 - Implicit invocation + Layering + other constraints
- Distributed objects
 - OO + client-server network style
 - E.g., Java RMI



C2 Style



- Read Component- and Message-Based Arch Style for GUI Software (Canvas/Reading/Taylor_Component)
- A hybrid style—event based style + layered style
- Outgrowth of GUI development, MVC pattern
 - To support coarse-grained reusability



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.



C2 Style



- Components: Independent, potentially concurrent message generators and/or consumers
- Connectors: Message routers that may filter, translate, and broadcast messages of two kinds: notifications and requests.
- Data Elements: Messages – data sent as first-class entities over the connectors. Notification messages announce changes of state. Request messages request performance of an action.
- Topology: Layers of components and connectors, with a defined “top” and “bottom”, wherein notifications flow downwards and requests upwards.



Intellectual Heritage

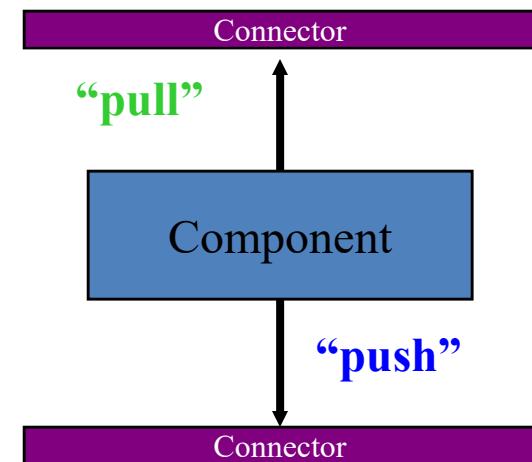


- Domain-specific software architectures
 - concurrent, loosely coupled, distributed, dynamic applications
- Smalltalk's model-view-controller paradigm
- Field's event-based integration
- Abstraction and separation of concerns
- OO typing
- Lisp, et.al. dynamic properties

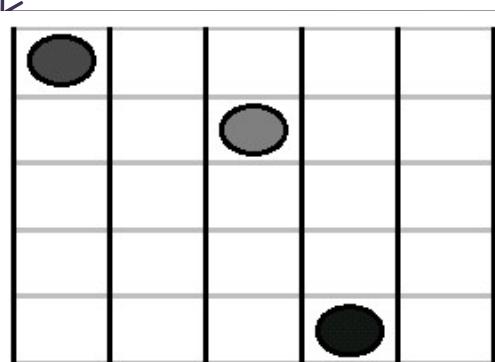
A Second Look at the C2 Style



- Asynchronous, event-based communication among autonomous components, mediated by active connectors
- No component-component links
- Event-flow rules
- Hierarchical application
Notifications fall
Requests rise



Example: KLAX Video Game



http://www.youtube.com/watch?v=M-xVg_dZjoc



KLAX Chute

- Tiles of random colors drop one cell at a time, starting at random times and locations

KLAX Palette

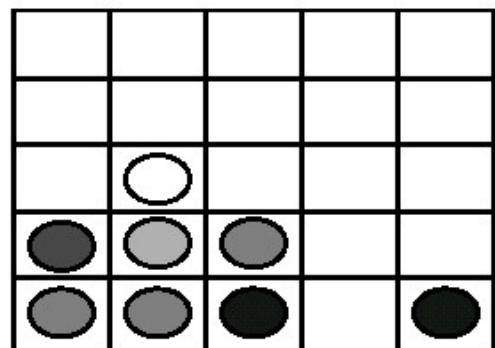
- Palette manipulated to catch tiles coming down the Chute and to drop them into the Well

KLAX Well

- Horizontal, vertical and diagonal sets of three or more consecutive files of the same color are removed, and any tiles above them collapse down to fill in the newly-created empty spaces

KLAX Status

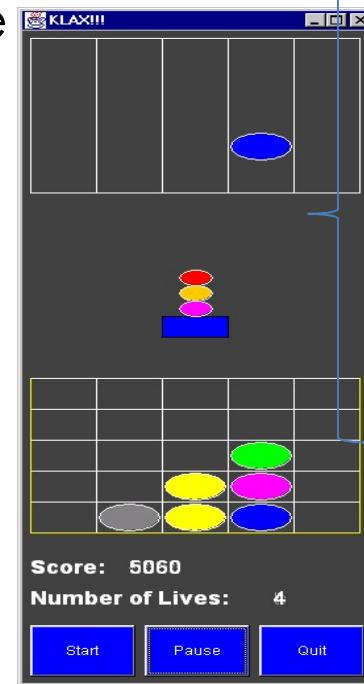
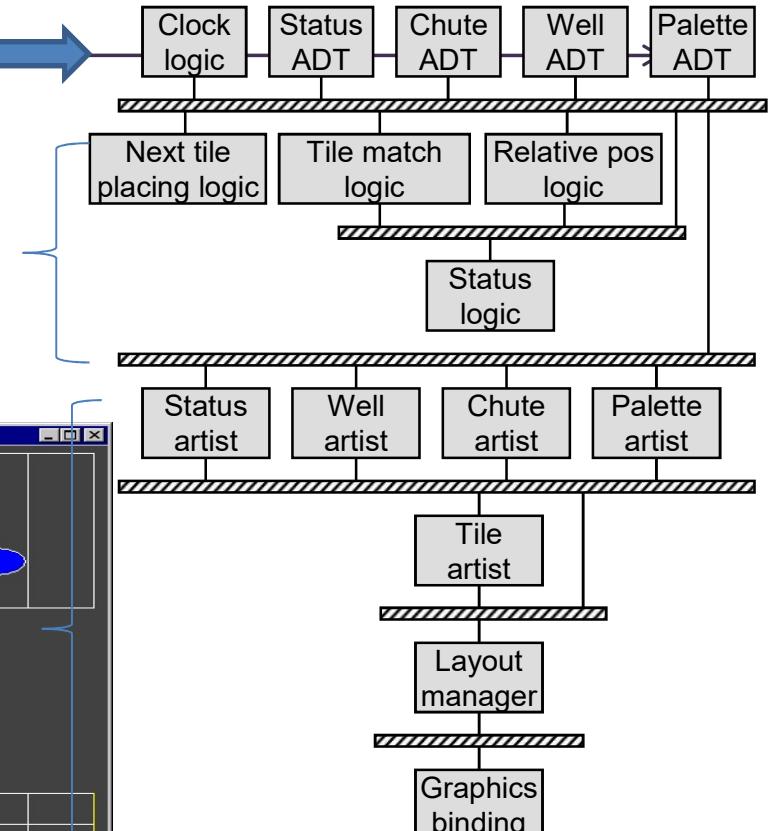
- Points scored as sets are formed
- Lives lost as Well or Palette spills over



Score: 100
Lives: 2

A Simple Example: KLAX

- KLAX™ game
 - 16 components, approx. 4k LOC, 100kb compiled
 - implemented from scratch in the C2 architectural style [Taylor et. Al. 1996]

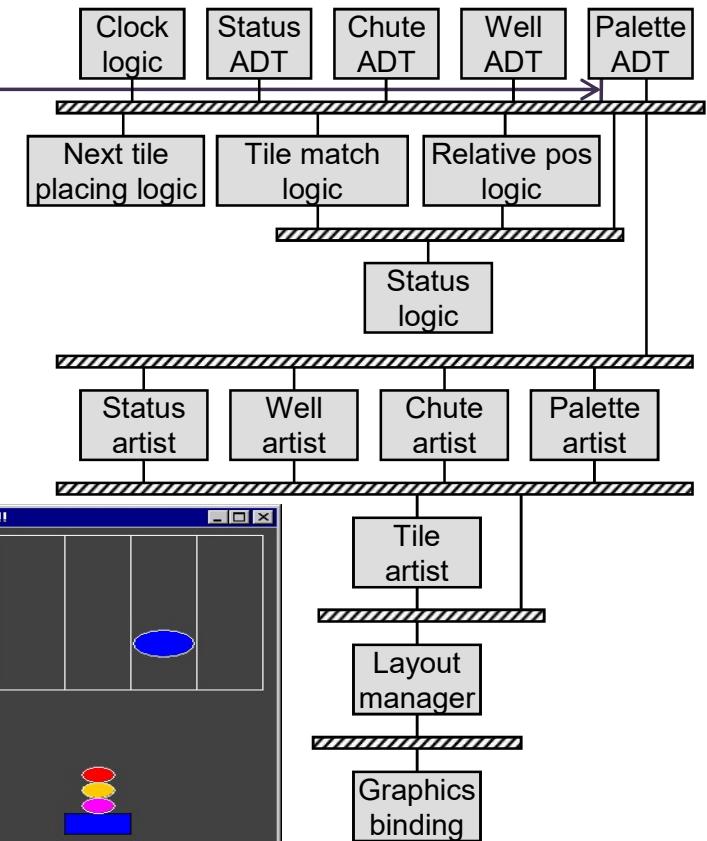


CSS 553, Spring 2023, 5a C2

KLAX: Adaptability

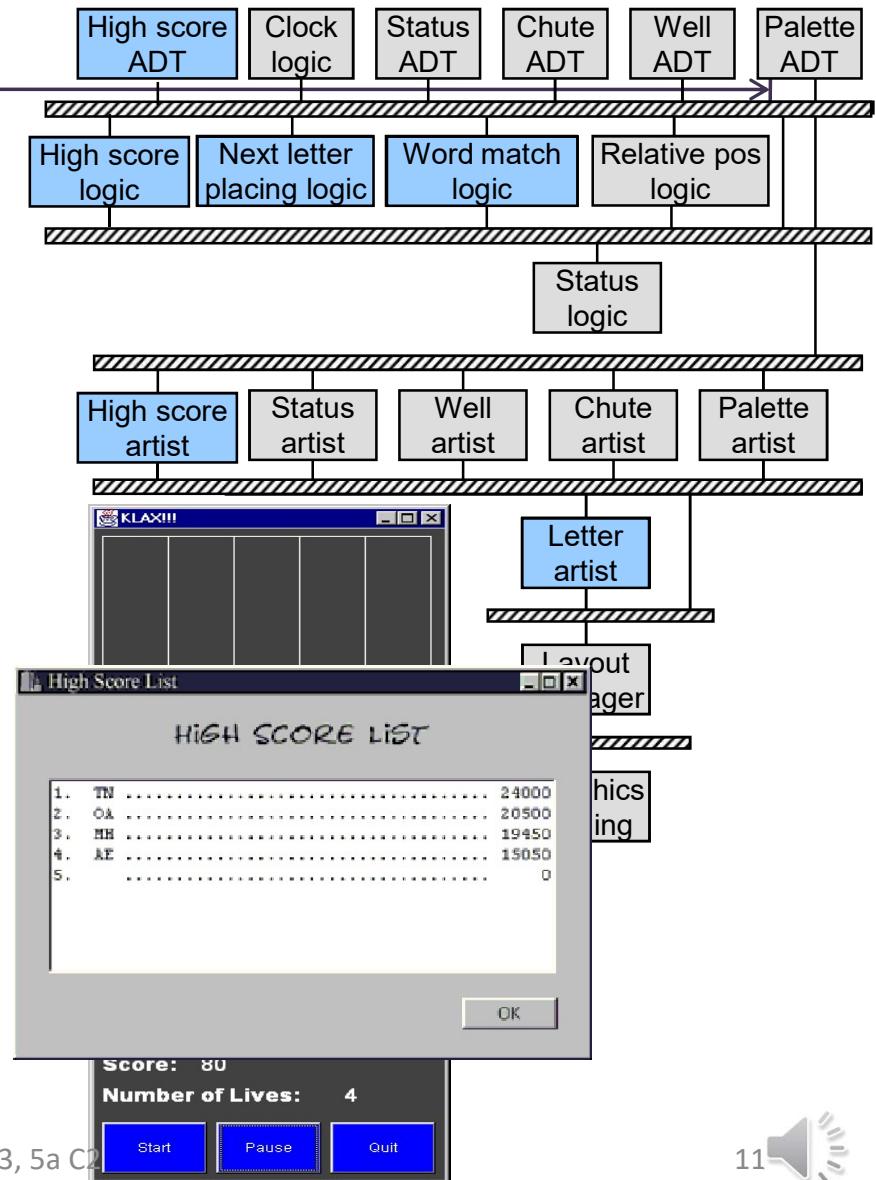


- Spelling KLAX
 - spell words from falling letters
 - replaces 3 components
- High score list
 - adds 3 new components
- Multi-player networked KLAX
 - a match adds a tile to opponents' chute



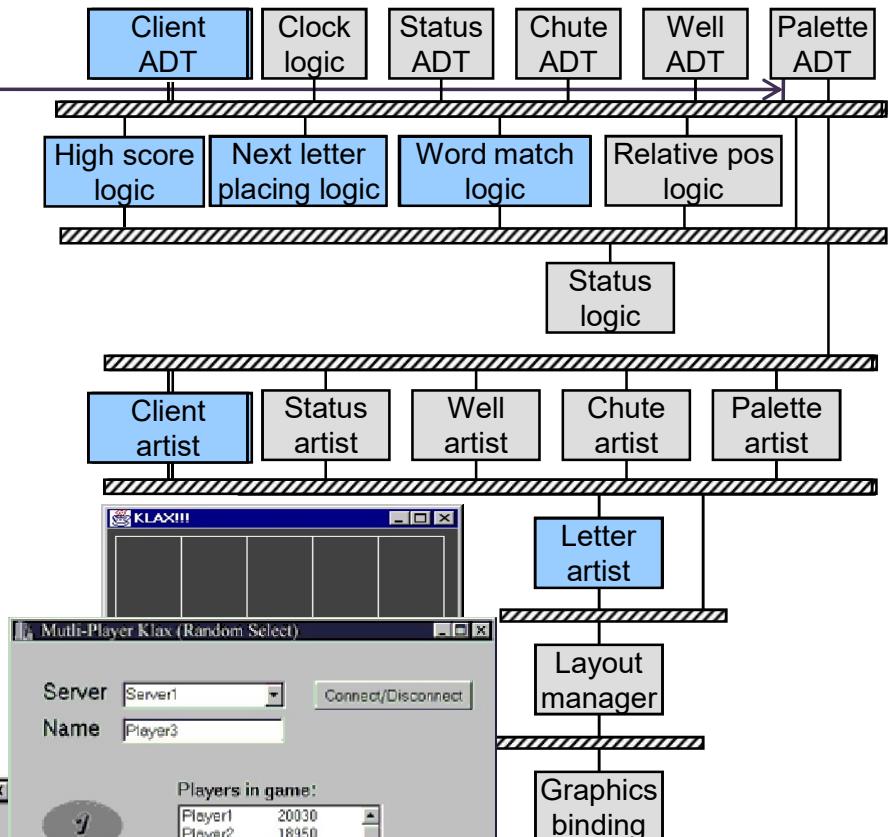
KLAX: Adaptability

- ← • Spelling KLAX
 - spell words from falling letters
 - replaces 3 components
- • High score list
 - adds 3 new components
- Multi-player networked KLAX
 - a match adds a tile to opponents' chute

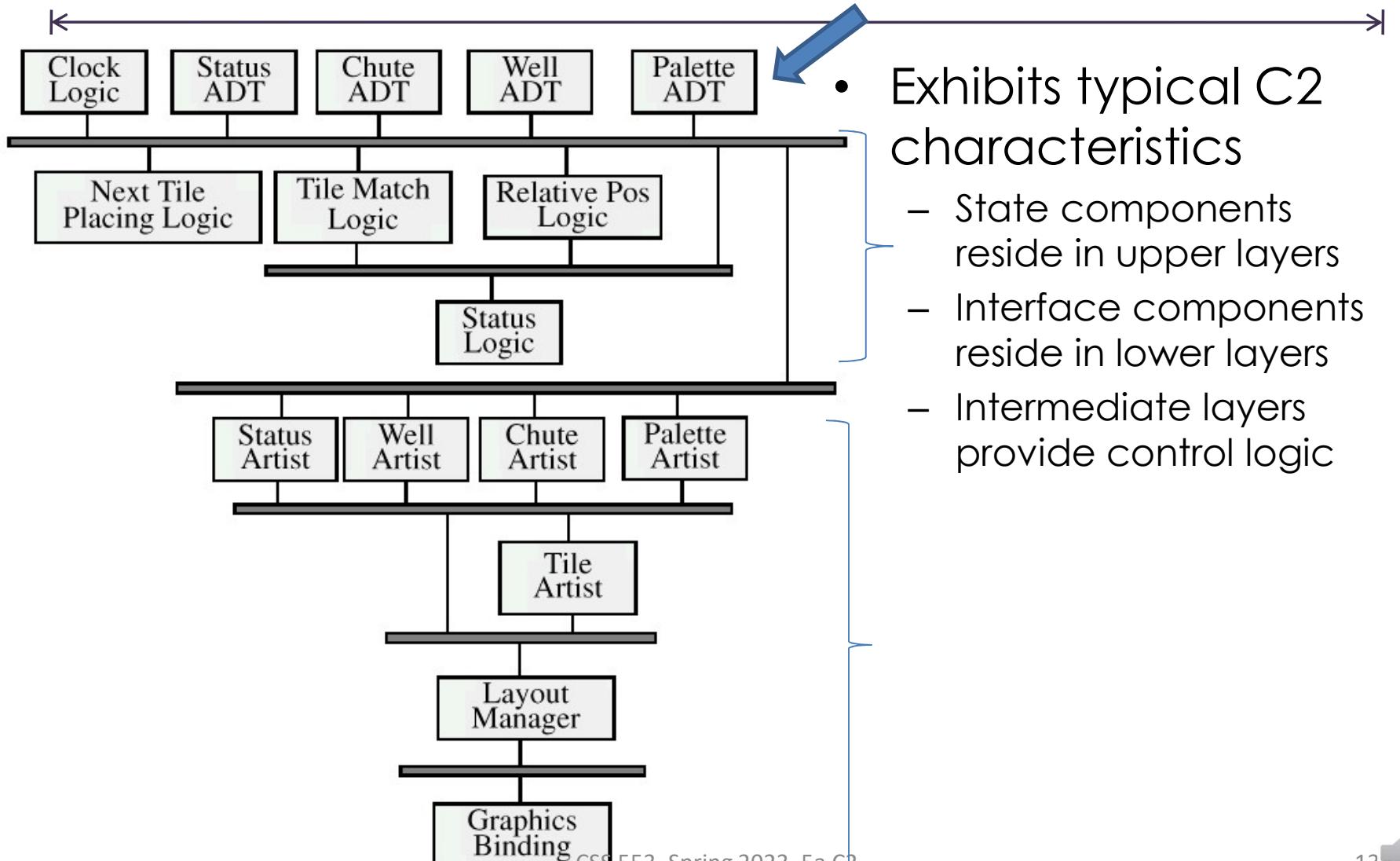


KLAX: Adaptability

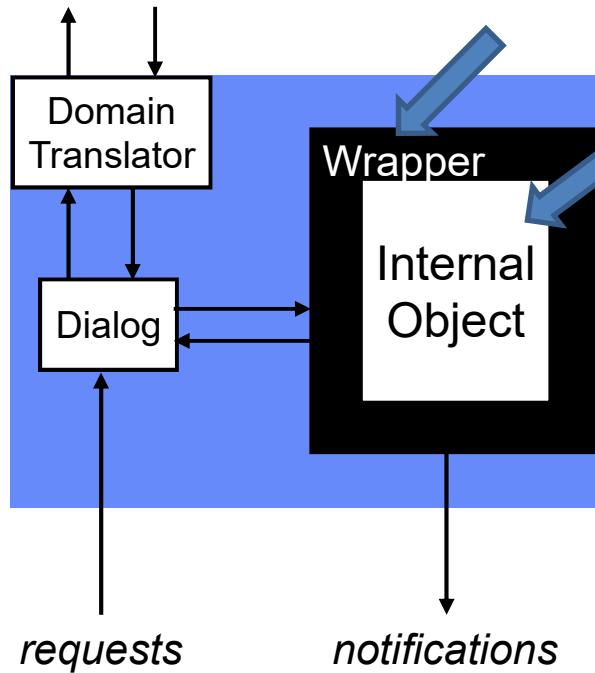
- • Spelling KLAX
 - spell words from falling letters
 - replaces 3 components
- • High score list
 - adds 3 new components
- • Multi-player networked KLAX
 - a match adds a tile to opponents' chute



A C2 Style for KLAX

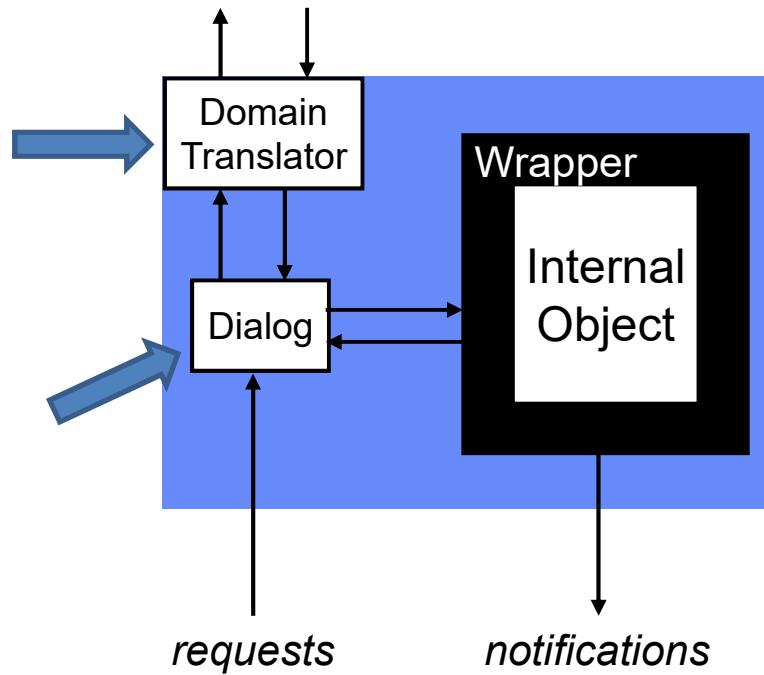


C2 Component Structure (I)



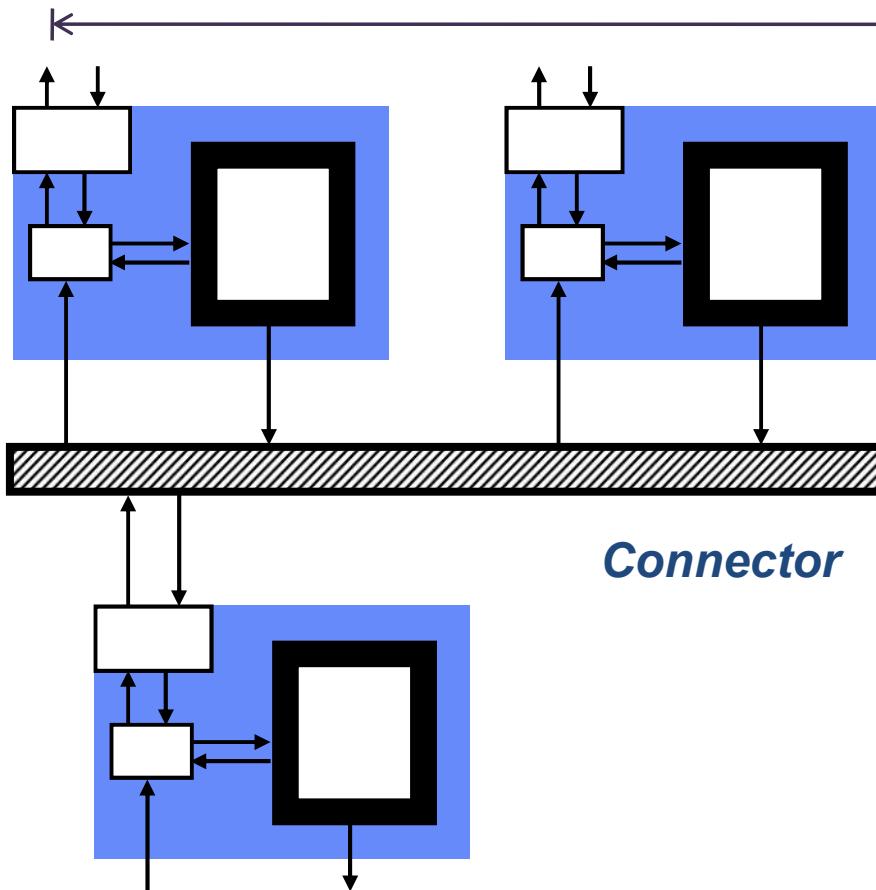
- The *internal object* is the “guts” of the component
- The *wrapper* maps messages to operations of the internal object
 - Requests \Rightarrow Operation Invocations
 - Operation Results \Rightarrow Notifications
 - Wrapped object can be described in terms of method signatures
- This organization allows reuse of independently-developed, “off-the-shelf” components as the internal object

C2 Component Structure (II)



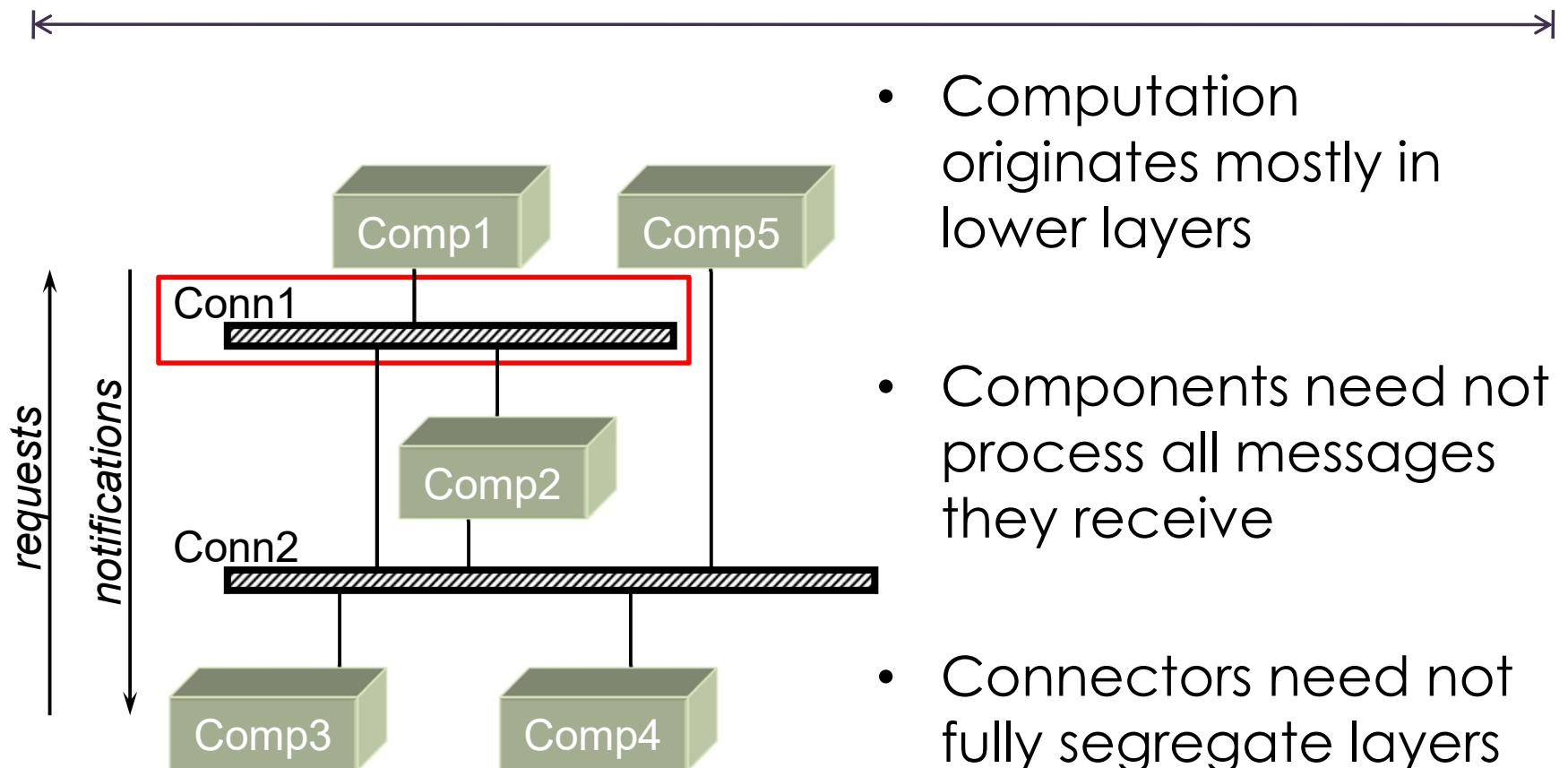
- The *dialog* provides the “control logic”, routing any relevant message to/from the internal object
- The *domain translator* translates messages from this component’s “vocabulary” to that of the layer above
- Note that a component never requests anything of components “beneath” it

C2 Component Connection



- C2 Connectors
 - Broadcast all messages received on the bottom side to the top side, and vice versa
 - Operate across machines
 - Operate across languages
 - Accommodate dynamic attachment and removal of components
 - Can be implemented in a number of ways
 - Procedure calls
 - Java RMI
 - ...

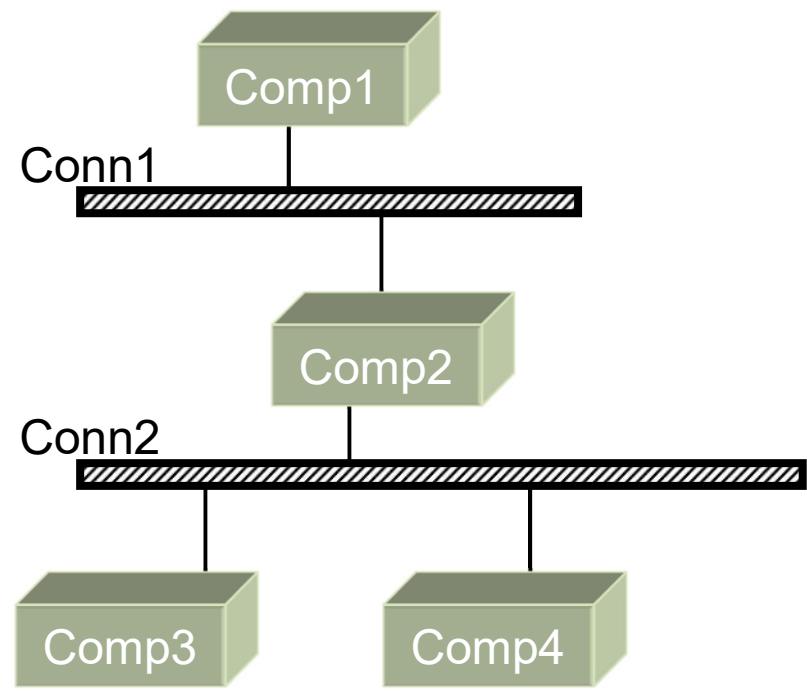
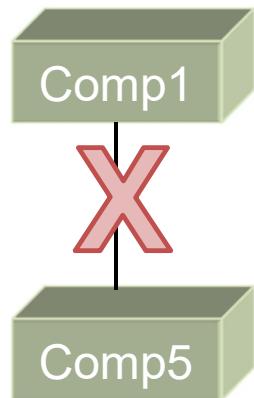
A Typical C2 Configuration



Rules of the C2 Style



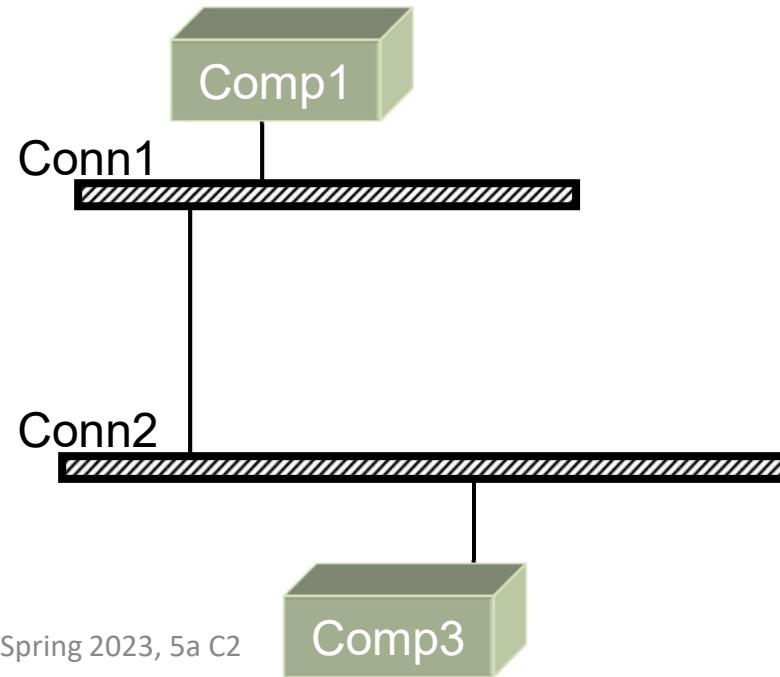
- Components are never attached to other components
 - A component can be attached to at most one connector on its top side and one on its bottom side



Rules of the C2 Style



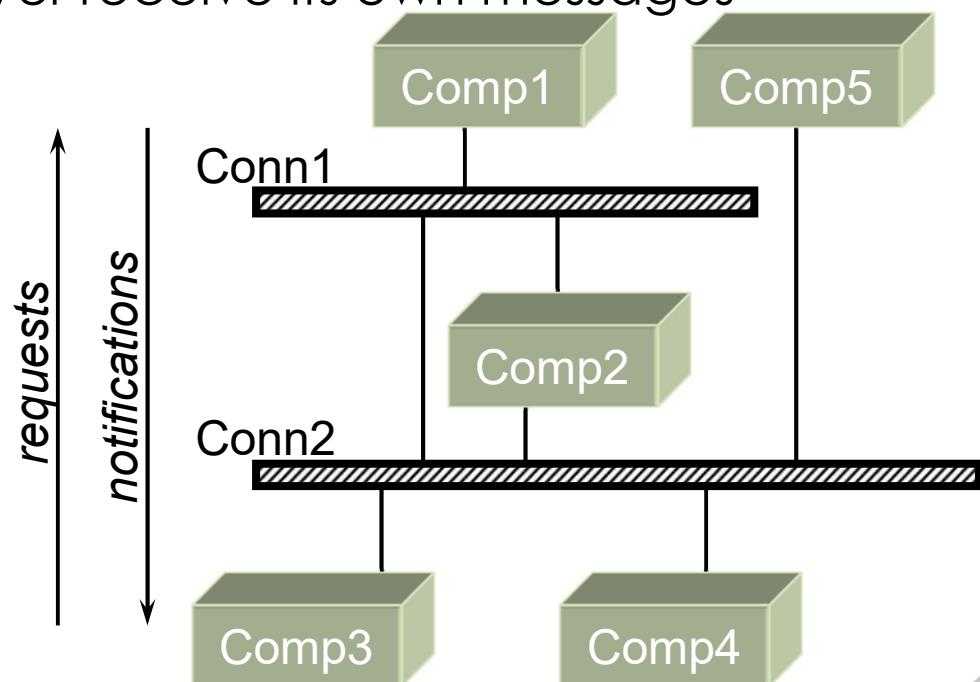
- Connectors can be attached to other connectors
 - A connector must have at least one component or connector attached on its top side and one attached on its bottom side



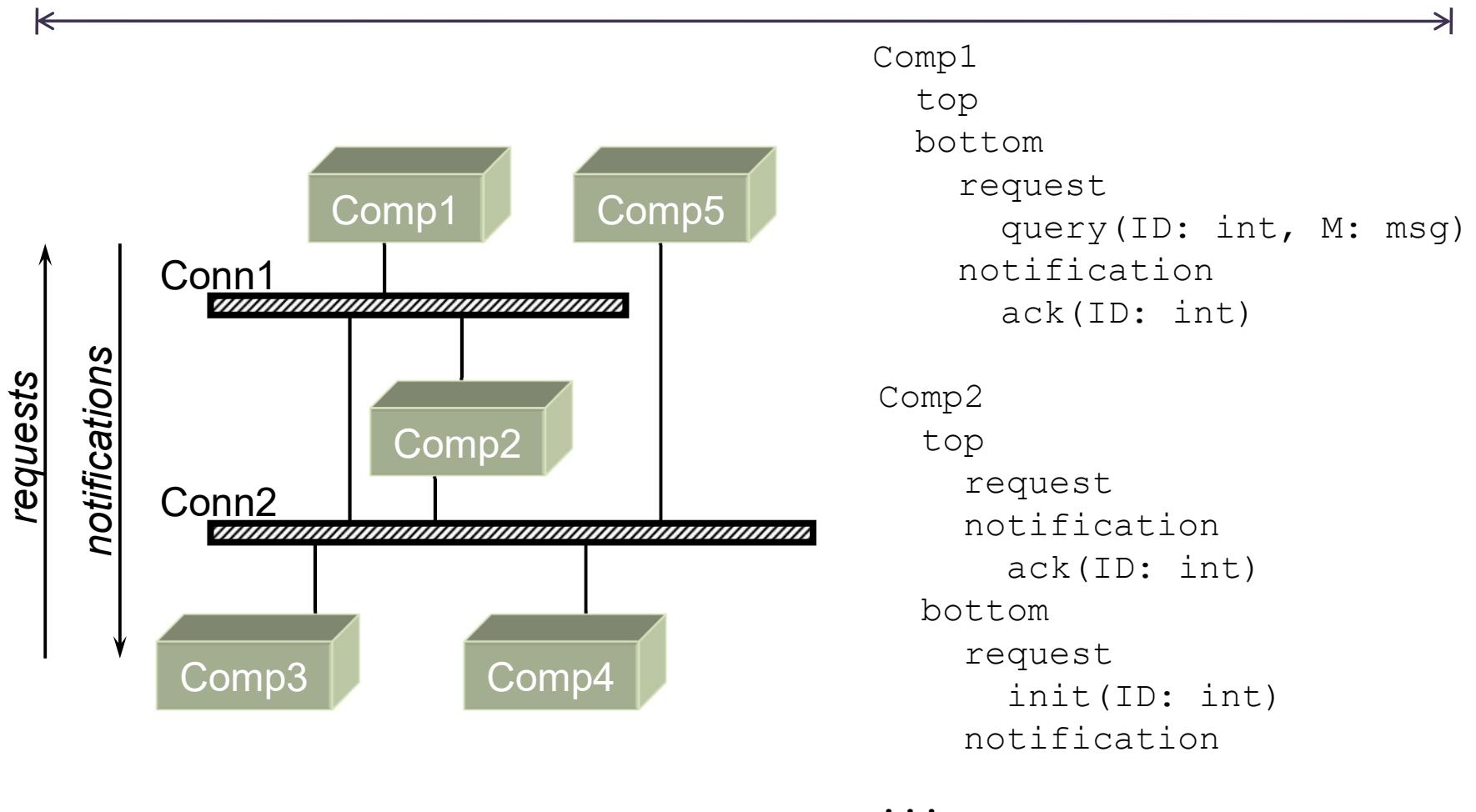
Rules of the C2 Style



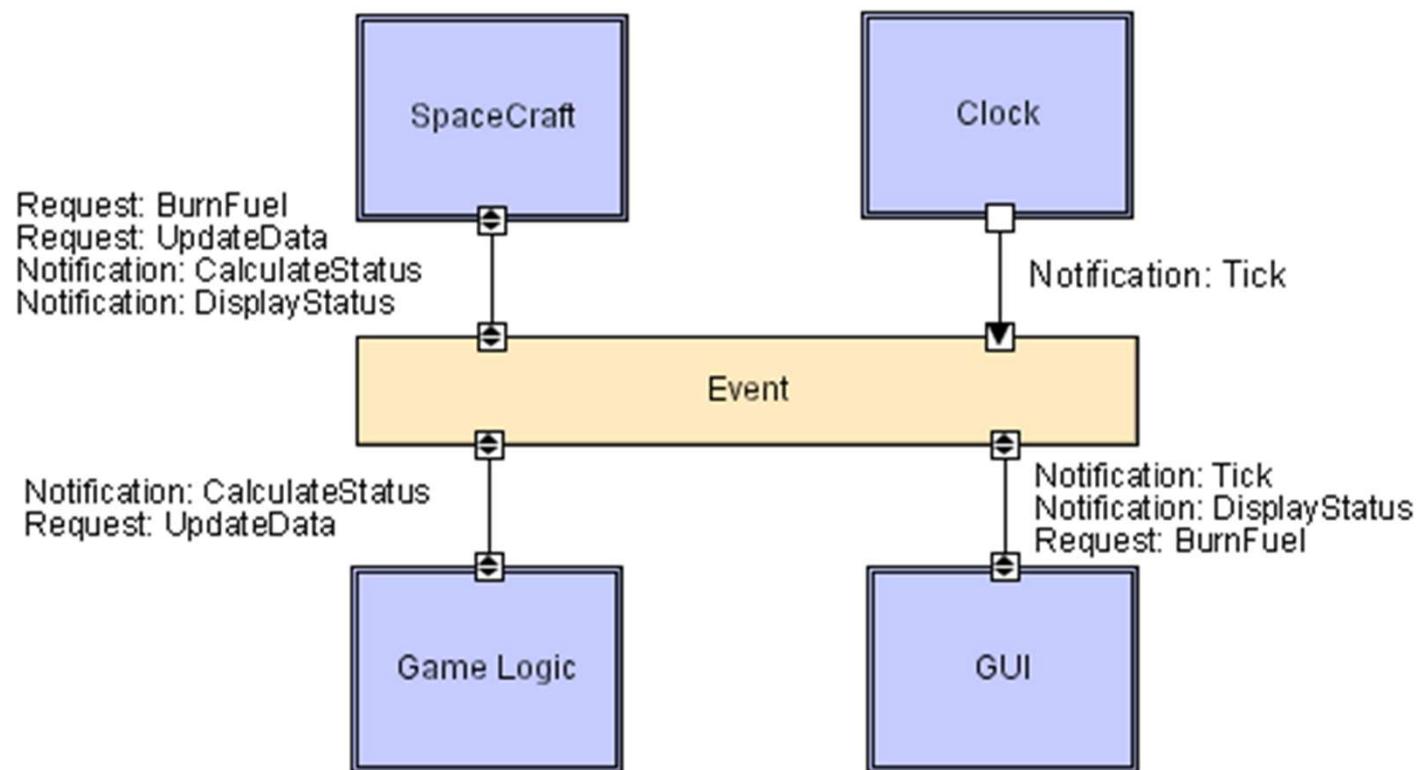
- Requests only go *up* in an architecture
- Notifications only go *down* in an architecture
- There is *no circularity* allowed
 - A component can never receive its own messages



Component Interfaces



Lunar Lander in C2



Sample Implementation



- Java implementation:
 - <https://isr.uci.edu/architecture/JavaFrameworkDocumentation/Package-c2.demo.html>
- Lunar lander in C2:
 - <http://www.softwarearchitecturebook.com/resources/>

Qualities



- Advantages
 - Reusability (discrete components promote reusability)
 - Evolvability/Flexibility: Because event receivers are not explicitly named, it is possible to integrate new components fairly easily
 - Dynamism (can change architecture on the fly)
 - Visibility (explicit & mandatory connectors provide points for observation)
 - Distributability (all events are assumed to be independent, and therefore components can be distributed across hosts)
 - Understandability (particularly promoted via substrate independence)

Qualities



- Disadvantages
 - Familiarity: Programmers often not initially familiar with event-based & asynchronous programming
 - Complexity: Because of asynchrony, concurrency issues are more prevalent and must be handled, event interactions are not always obvious (although tools can help!)



CSS 553, Spring 2023, 5a C2



26

READ THIS FIRST - 20 minutes

Topic: Architecture Styles

This activity is designed to

- Give you practice in designing
- Tie up loose ends regarding upcoming assignment(s)

Roles:

• **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.

• **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.

• **Note Taker** - takes record of the group’s discussion in the Google doc (below).

• **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.

2. Then respond to the prompts, taking notes on your discussion

3. I will assign your group for either Task 2a or 2b

If you have any questions about this activity or the assignments due tonight, type in Red text

Task assignments:

Dawn - 2a

The Boffins - 2b

To be decided - 2a

Team 4 - 2b

Ludus - 2a

Team 9 - 2b

Magratheans - 2a

Open source web services - 2b

2a: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

- ...using a pencil and paper...
- ...be able to explain its meaning...
- ...and money is no object...
- Take a picture of your design (or create it directly on the slide)

2b: •Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

- ...using whatever method you like...
- ...Cost is a concern. Make sure that the cost of implementing your design is reasonable...

Team Name: DAWN

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1: Do we need to utilize the software architecture pattern for this 2a?

2a does not need an arch pattern - just create a design for an award

Ques #2: MS project mean a project in Microsoft? Or a project in Master?

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

2a: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

- ...using a pencil and paper...
- ...be able to explain its meaning...
- ...and money is no object...
- Take a picture of your design (or create it directly on the slide)*

List Roles and Student names

Moderator : Abdul-Muizz Imtiaz

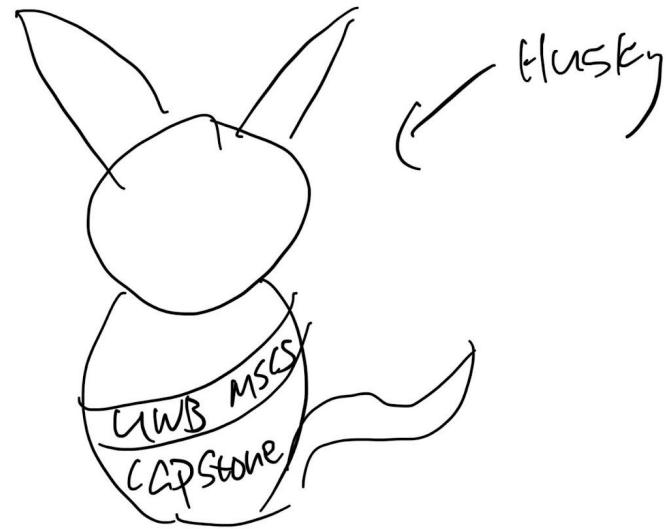
TimeKeeper: Luo Leng

NoteTaker: Barrack

Reporter: Chloe Ma



Aa



This is a true piece of art :)
Definitely NOT a Pikachu

Design Concept: "The Capstone Project Pillar of Excellence"

Description: The award consists of a cristal husky on the front. The husky represents the heights of achievement, determination, and freedom to innovate in the realm of Capstone projects.

At the base of the pillar, there is a circular metal platform with a logo W etched onto it, symbolizing the global impact of outstanding Capstone projects. The platform is made from bronze, representing the richness and diversity of the technologies and ideas that come together in successful projects.

The award recipient's name and the year of the award are inscribed on a small plaque affixed to the front of the base.

Meaning:

1. **Cristal Husky:** Represents clarity of vision, transparency, and unwavering commitment to excellence in project management. Symbolizes the heights of achievement, determination, and freedom to innovate.
2. **W Logo:** Stands for the global impact and reach of excellent capstone projects.
3. **Bronze Platform:** Highlights the richness and diversity of technologies and ideas in the projects.



Team Name: **The Boffins**

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

2b: •Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

•...using whatever method you like...

•...Cost is a concern. Make sure that the cost of implementing your design is reasonable...

Designed using Strategy Pattern to sync traffic controllers with sensors in traffic lights, in a scalable manner

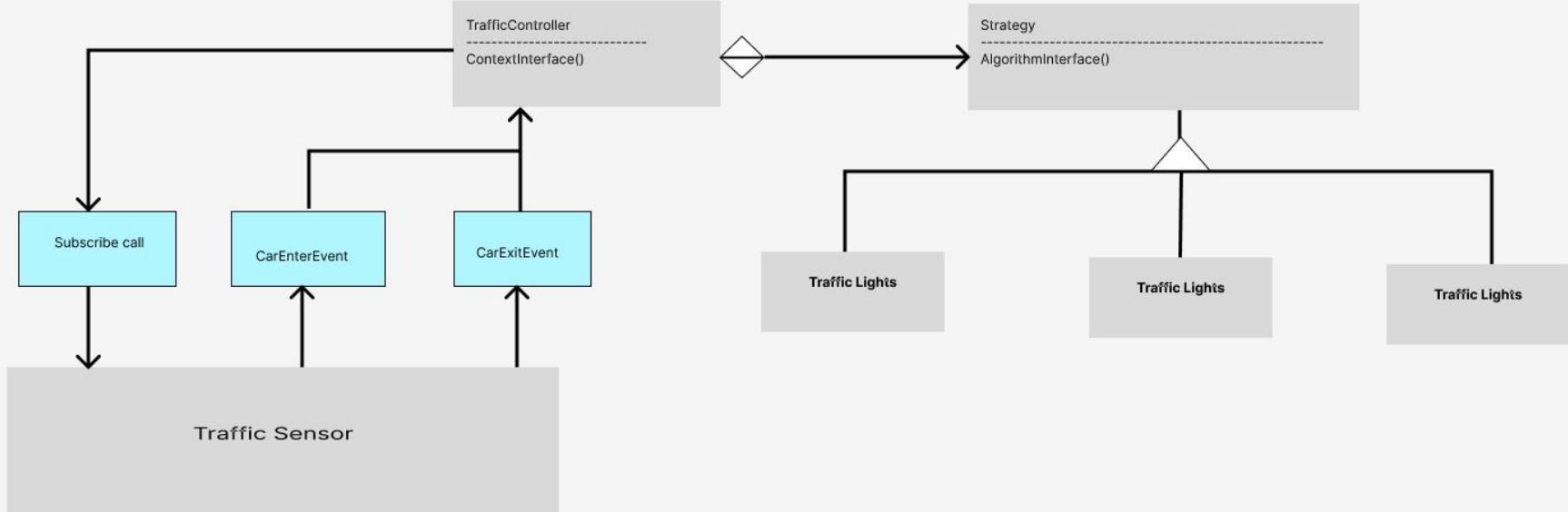
List Roles and Student names

Moderator: Sukeerth Vagaraju

TimeKeeper: Jaron Wang

NoteTaker: Holly East

Reporter: Ioana Preda



Team Name: to be decided

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

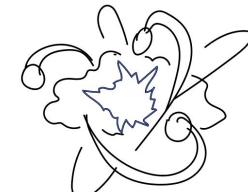
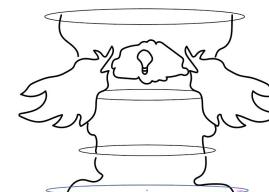
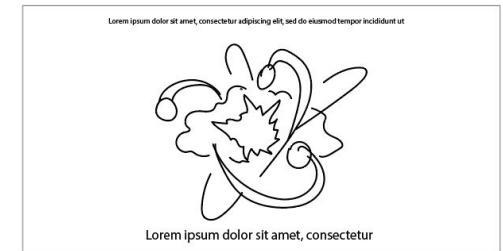
2a: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

- ...using a pencil and paper...
- ...be able to explain its meaning... innovation - breakthroughs in CS/Tech

Web(internet) -> ML -> Quantum comp

•...and money is no object...

•Take a picture of your design (or create it directly on the slide)



List Roles and Student names

Moderator : Sonal Yadav

TimeKeeper: Angelo Williams

NoteTaker: Naima Noor

Reporter: Noura Alroomi

Team Name: Team 4

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

2b: •Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...

•...using whatever method you like...

•...Cost is a concern. Make sure that the cost of implementing your design is reasonable...

Sense-Compute-Control architecture pattern

Sense the amount of traffic, compute the optimized traffic light status strategy, and control the status of the traffic light

Cost reduction: traffic lights take turns to communicate with the server, with consideration of a timely communication against the cost of bandwidth

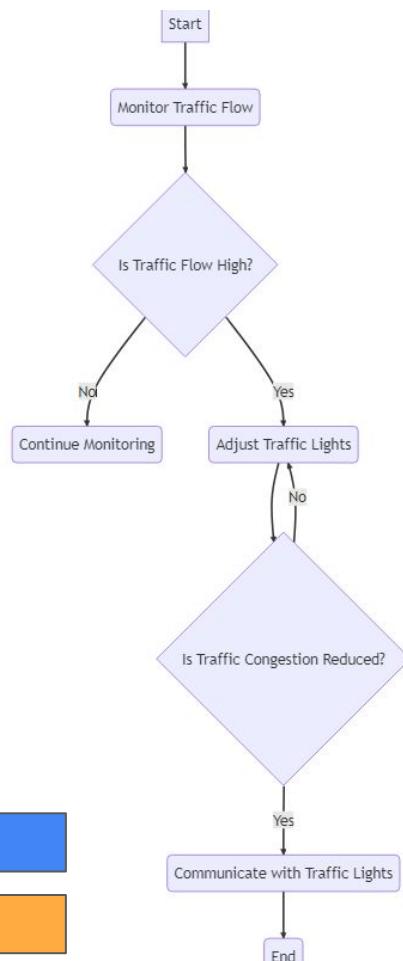
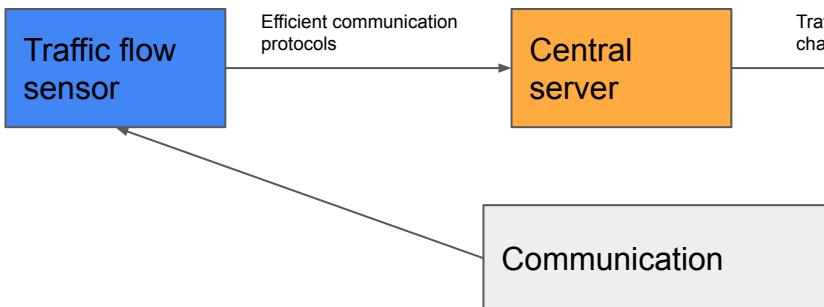
Next page for details

[List Roles and Student names](#)

Moderator : Chengjun Xi

TimeKeeper: Joshua Medvinsky

1. Main system: Use a Client-Server architecture style:
 - a. Cameras act as clients, capturing traffic data and sending it to the central server.
 - b. The central server processes the traffic data, makes decisions on traffic light control, and sends commands to traffic lights.
2. Within the server: rule-based system to make decisions
 - Sense-Compute-Control architecture pattern
 - Sense the amount of traffic, compute the optimized traffic light status strategy, and control the status of the traffic light
3. Cost reduction: traffic lights take turns to communicate with the server, with pondering of a timely communication against the cost of bandwidth



Team Name: Ludus - 2a

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1: **When is this award given?** Will there be an audience? Will this be recorded?

Yes, there may be an audience. It could be recorded (though not guaranteed)

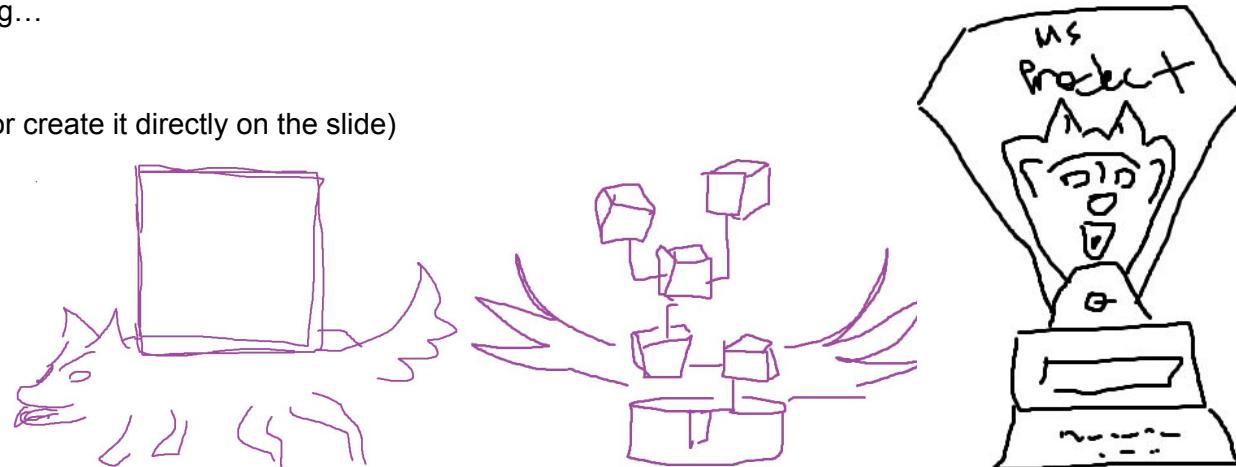
Ques #2: When will the award be presented and in which situation? I.e. during graduation ceremony or as a send via mail.

During graduation (or some other ceremony). This is not just via email

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

2a: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

- ...using a pencil and paper...
- ...be able to explain its meaning...
- ...and money is no object...
- Take a picture of your design (or create it directly on the slide)



List Roles and Student names

Moderator : Tyson

TimeKeeper: Shaun Stangler

NoteTaker: Agustin

Reporter: Austin

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Magratheans

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

2a: Suppose we are to give out an award for excellence in MS project. Create a design for this award...

- ...using a pencil and paper...
- ...be able to explain its meaning...
- ...and money is no object...
- Take a picture of your design (or create it directly on the slide)

Type of material used for the trophy can symbolize different types of merit.

Diamond-plated trophy in shape of the classic Mac and picture of professor Erdly (the head of UW Bothell CS department)

[List Roles and Student names](#)

Moderator : Karan

TimeKeeper: Ryan

NoteTaker: Thomas

Bill Erdly Certificate of Excellence

Awarded to

*for stellar work in the field of computer
science and software engineering*



<recipient name> <year>

Team Name: Team 9

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

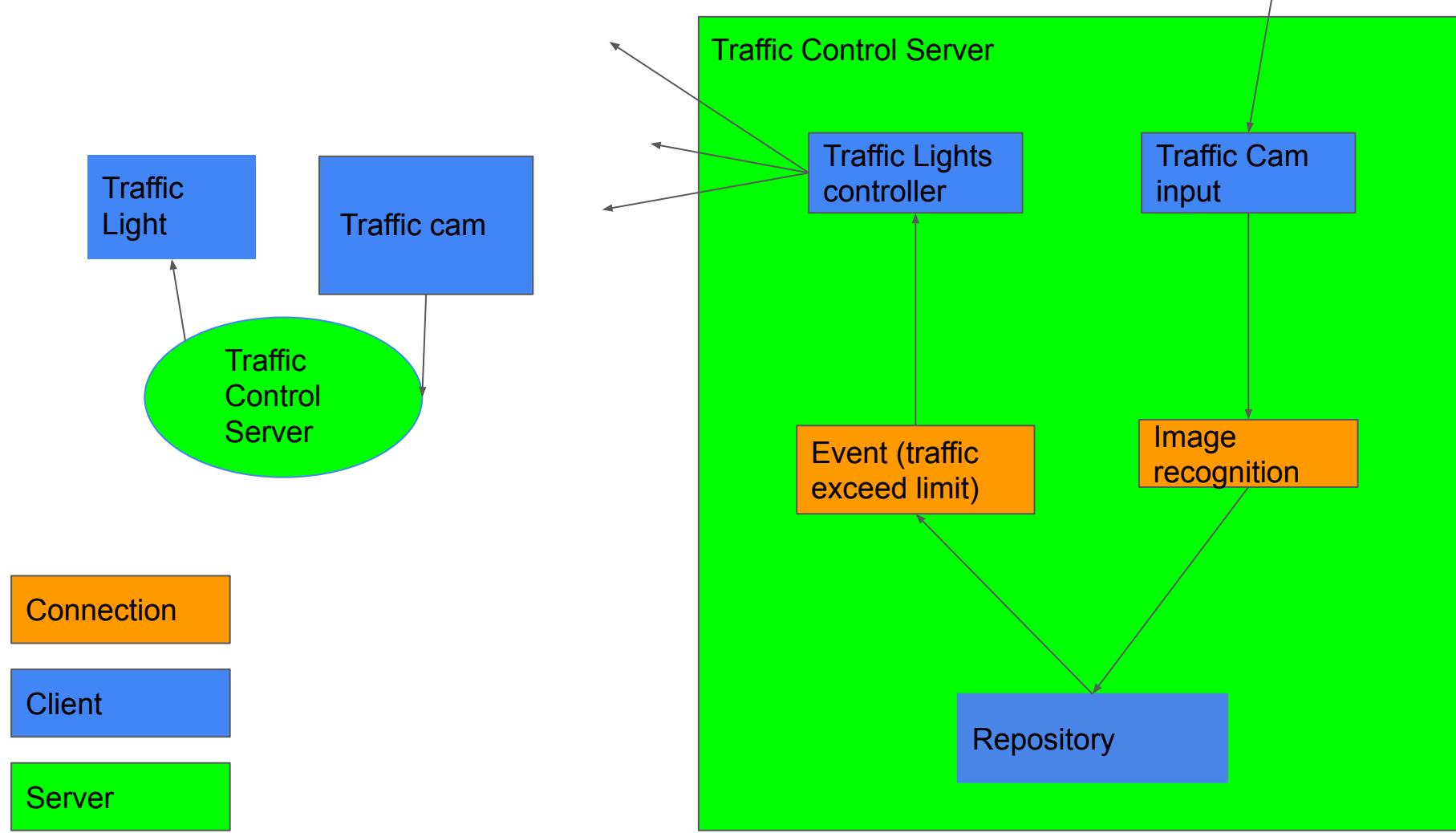
Ques #2:

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator : Pragati

Reporter: William



- Data collection from cam
- Analysis of cam data
- Change duration of signal according to traffic data
- Need robust safety mechanism (if any component fails it should have backup).
- Security: Only authorised person can access and also need to consider how to avoid hackers attack.
- Communication: real time communication

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: open sources web project

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2a or 2b: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

- 2b: •Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...
- ...using whatever method you like...
 - ...Cost is a concern. Make sure that the cost of implementing your design is reasonable...

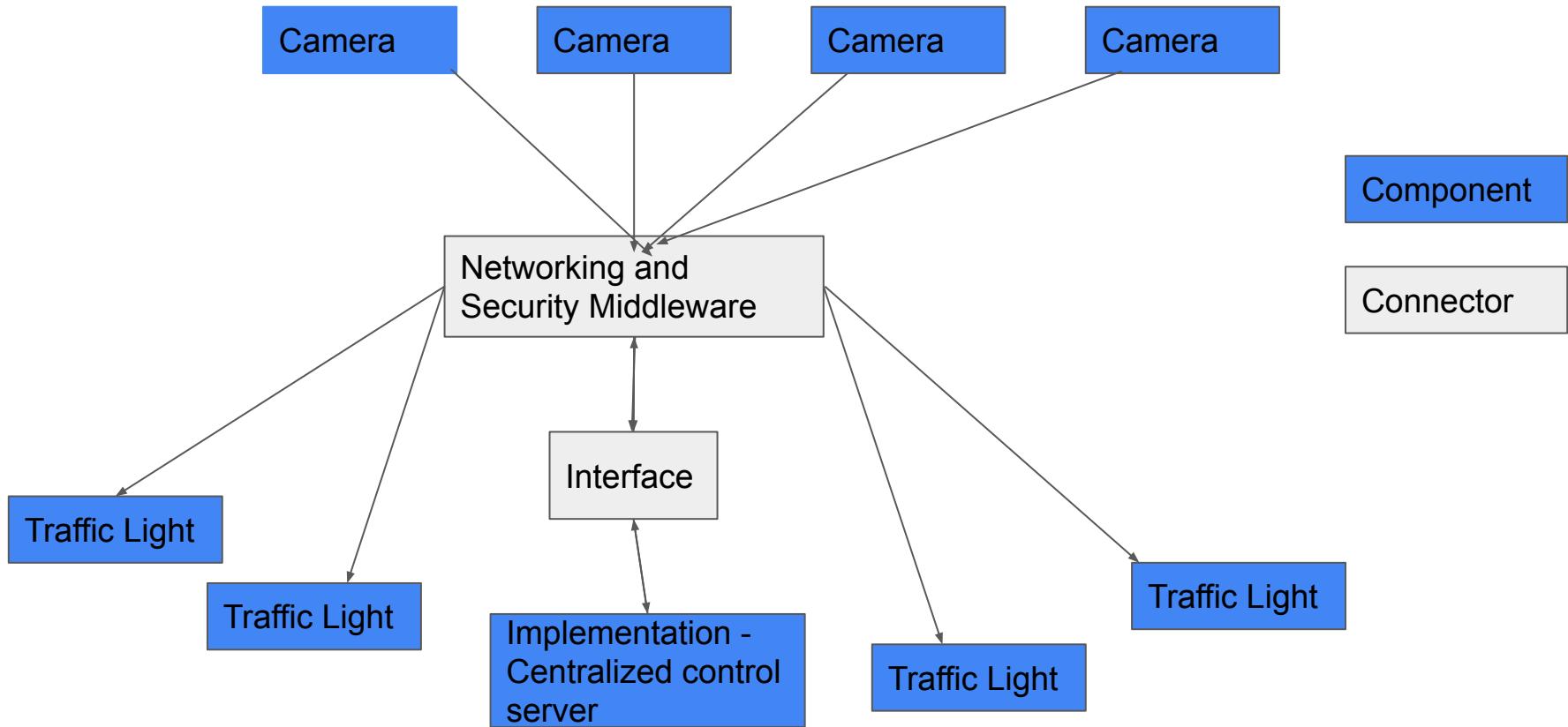
List Roles and Student names

Moderator : Yang Yu

TimeKeeper: All member

NoteTaker: Anthony Bustamante

Reporter: Sunjil Gahatraj





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Agenda



- Review / Questions
- Design Presentation
- Design Exercises – to Prep for midterm
- Mid-course survey

Questions



- Compare C2 with **one** previous architecture styles/patterns covered in class. What are similarities with that style? **Open Source WS Differences? The Boffins**
 - Pub-sub with C2 – event-based, implicit communication, distributed setting
 - pub-sub – less structured (no strict topology), C2 no circularity in messages
 - Pub-sub – list of subscribers, C2 – no specific list of subscribers.
- How can one address the issue of latency with C2? **Dawn**
 - Issue caused by asynchronous: optimize message routing, use more efficient data structure in connectors
 - Caching – same requests
- What types of applications can benefit from using C2?
Magratheans, Team 9
 - **Complex asynchronous communications, modular systems (different vendors): examples: Stock trading, video games**
 - **Cybersecurity - (some detectors, command center), robotics (assembly line)**
- Concepts in a programming language are often different

Questions

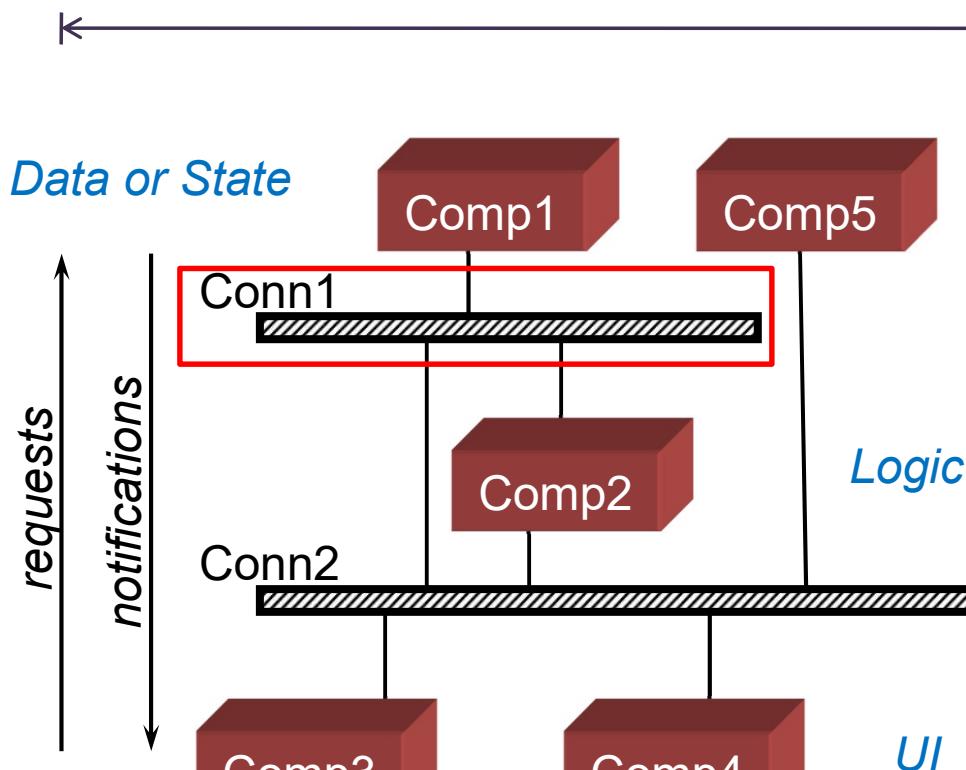


- Concepts in a programming language are often different from architecture concepts. How do you bridge this gap?

Team 4, To be decided

- **Architecture framework – do the translation (bridge) – from architecture to programming language**
 - Mapping is explicit – include arch model in documentation – generative technologies
- Explain how components communicate with each other in C2
- ## **Ludus**
- Requests – up “rise”
 - Notifications – down
 - broadcasts

A Typical C2 Configuration



- Computation originates mostly in lower layers
- Components need not process all messages they receive
- Connectors need not fully segregate layers

Announcements



- Visual Paradigm license
 - Waiting for IT to renew—can hand draw diagrams for G2

DESIGN PRESENTATION

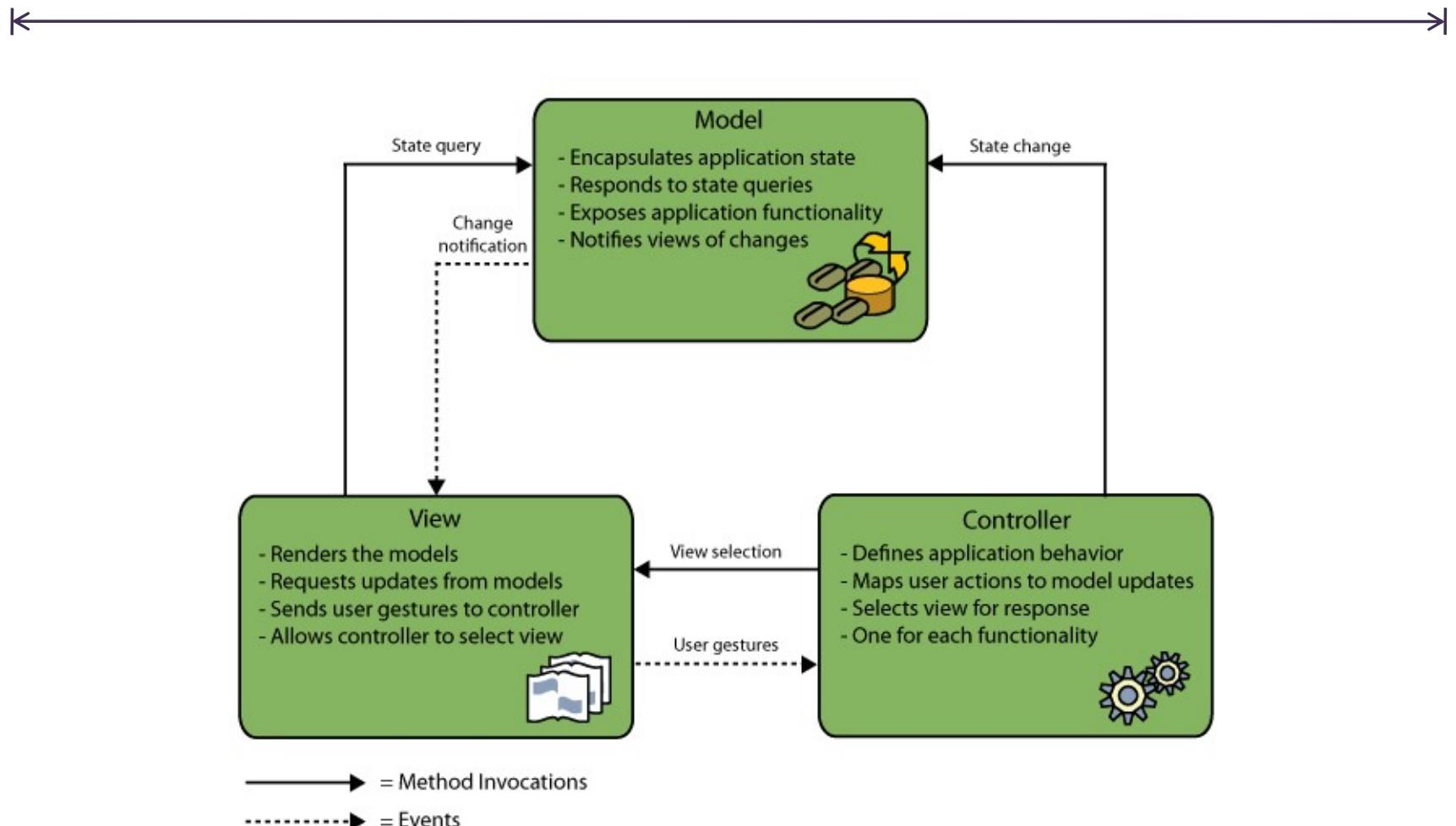
ACTIVITIES

Exercise 2a & 2b

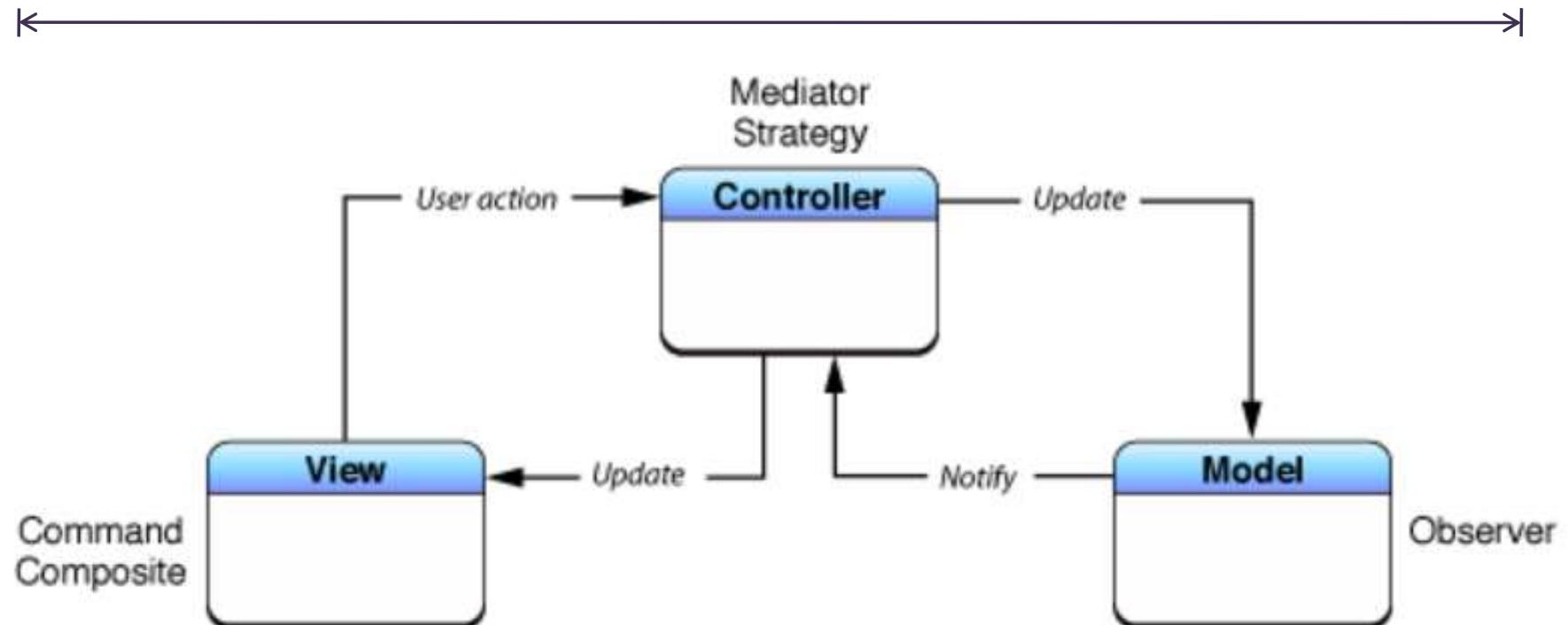


- How do you cast MVC in a C2 style?

Exercise 2a: MVC Variant



Exercise 2b: MVC Variant



Exercise 2c: A Long-Distance Telecommunications System



Draw a C2 architecture for a long-distance telecommunications System

- Telephones
- Local Switches (one per area code)
- Long-Distance Switches (communications backbone)
- Call Records database
- Other embellishments
 - Toll-Free numbers
 - 911 stations
 - Operator stations
 - Bill generators

READ THIS FIRST - 20 minutes

Topic: C2

This activity is designed to help you practice concepts about the C2 style

Roles:

- **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.
- **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.
- **Note Taker** - takes record of the group’s discussion in the Google doc (below).
- **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion. Reporter will defend the group’s answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity, type it in Red text

Task assignments:

Dawn - 2a

The Boffins - 2b

To be decided - 2c

Team 4 - 2b

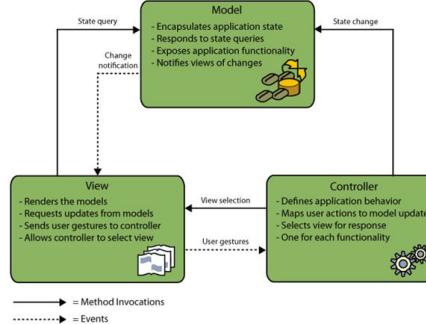
Ludus - 2a

Team 9 - 2c

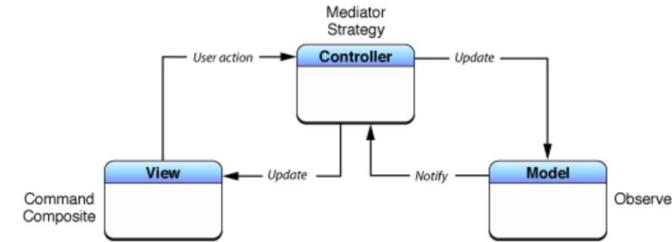
Magratheans - 2b

Open source web services - 2c

2a



2b



2a, 2b: How do you cast the following MVC in a C2 style?

- Take a picture of your design (or create it directly on the slide)

2c: Draw a C2 architecture for a long-distance telecommunications System

Telephones

Local Switches (one per area code)

Long-Distance Switches (communications backbone)

Call Records database

Other embellishments

- Toll-Free numbers
- 911 stations
- Operator stations
- Bill generators

Team Name : Dawn Specify one: 2a

Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

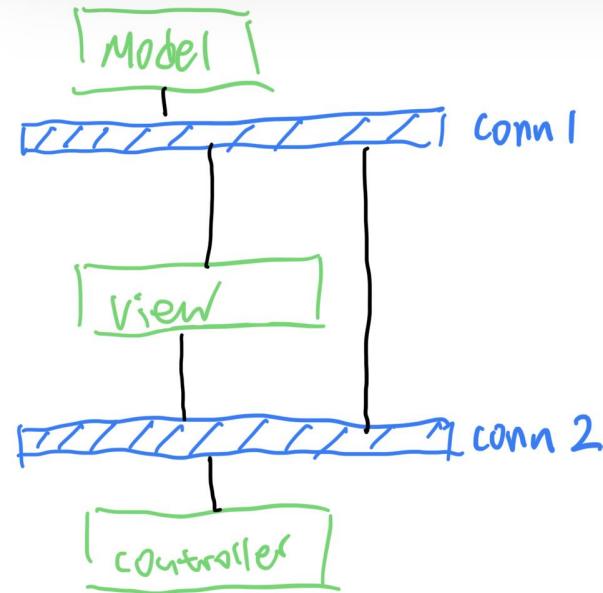
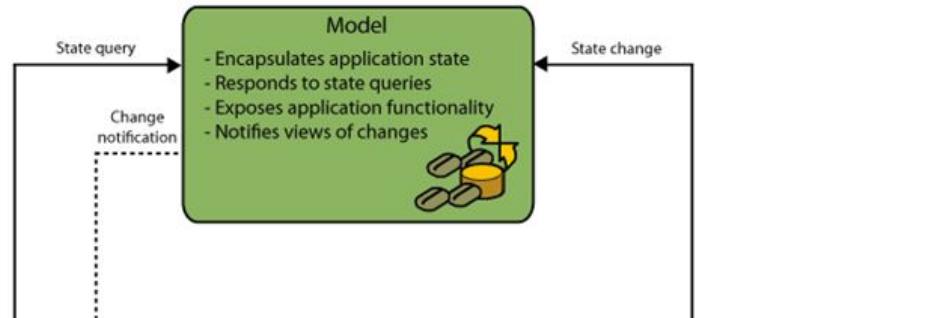
Ques #1:

Ques #2:

Task 2: create your assigned diagram here (Time: 17 minutes)

2a, 2b: How do you cast the following MVC in a C2 style?

- Take a picture of your design (or create it directly on the slide)



List Roles and Student names

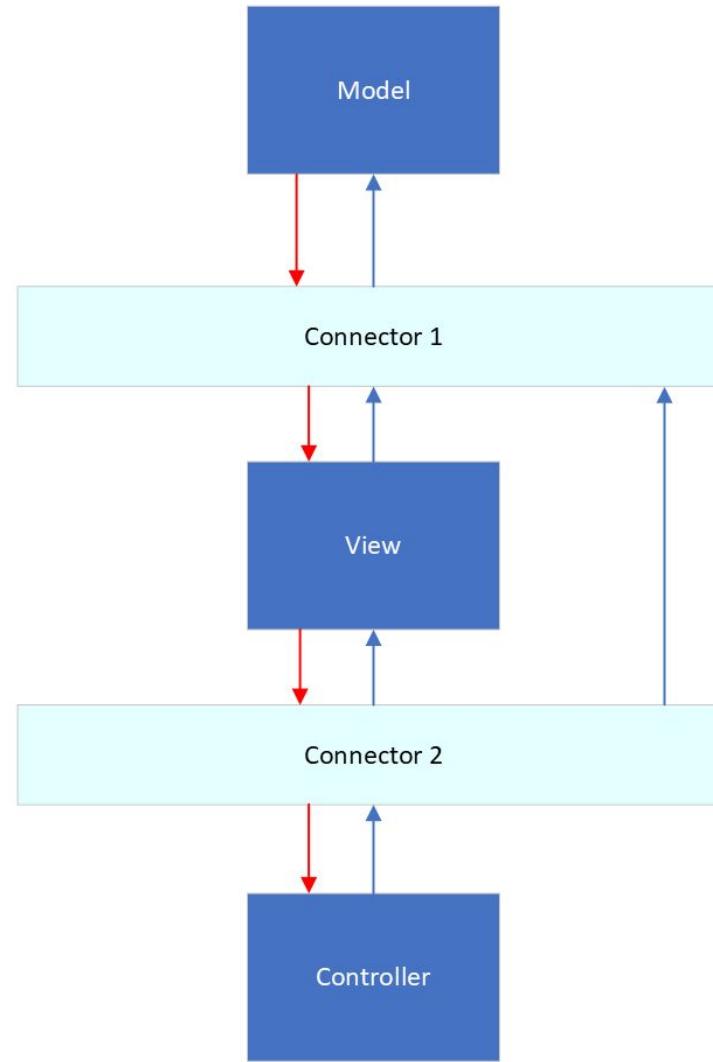
Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Dawn - 2a



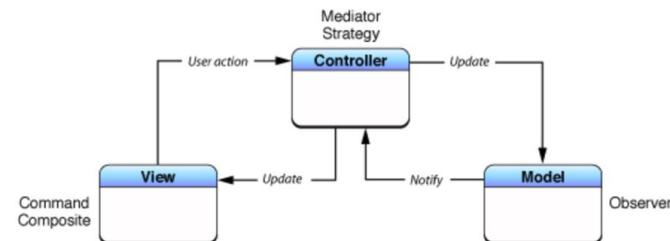
Team Name : The Boffins 2b

Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

Ques #1:

Ques #2:

Task 2: create your assigned diagram here (Time: 17 minutes)



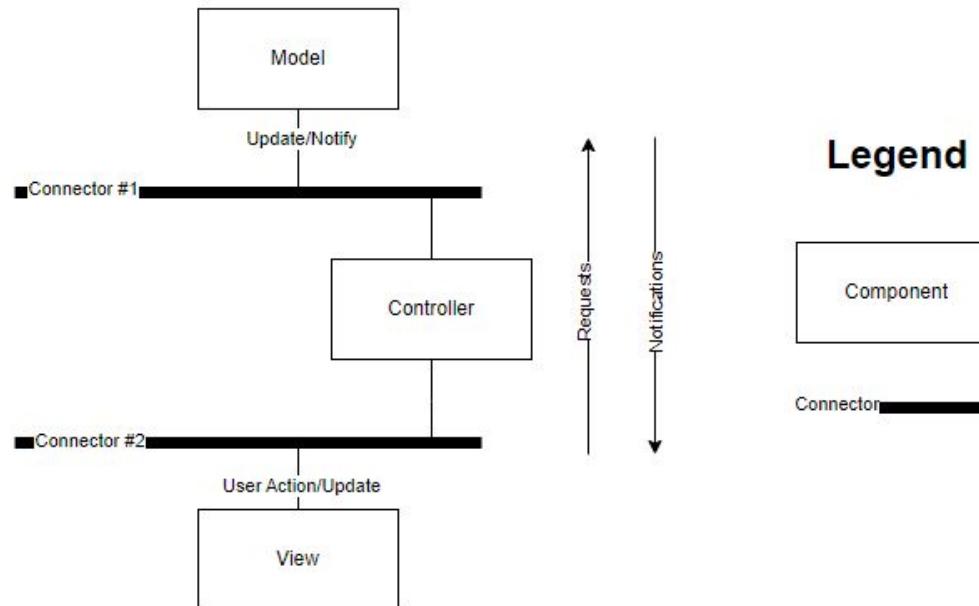
List Roles and Student names

Moderator : Holly East

TimeKeeper: Jaron Wang

NoteTaker: Sukeerth Vegaraju

Reporter: Ioana Preda



Team Name : To be decided Specify one: 2c

2c: Draw a C2 architecture for a long-distance telecommunications System

Telephones

Data / State

Local Switches (one per area code)

Long-Distance Switches (communications backbone)

Call Records database

Other embellishments

- Toll-Free numbers
- 911 stations*
- Operator stations
- Bill generators

Logic

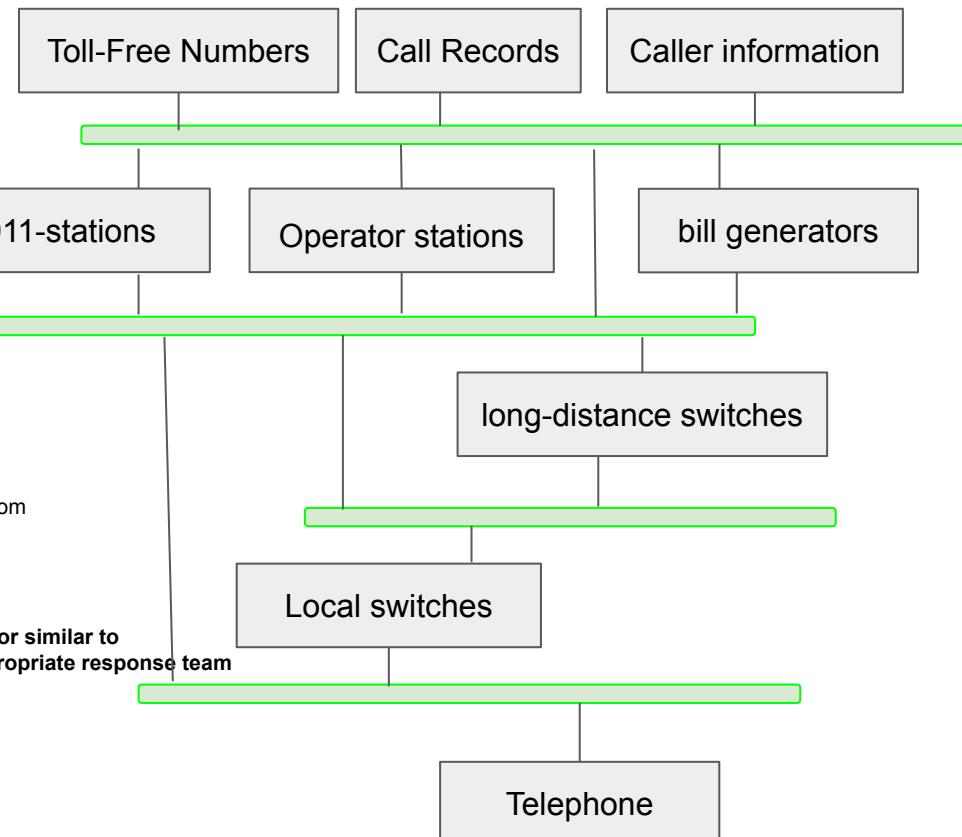
State (top), Logic (middle), UI (bottom)

Call records hold data information and would be at the very top (just a state).

Telephones are the interface for the users (callers), ie the GUI at the very bottom

Operator stations and bill generator are computing data such as routing calls and calculating bills.

*911 stations could either mean the stations the receive emergency calls or similar to operator stations where they are responsible for routing calls to the appropriate response team



Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

Ques #1:

Ques #2:

Caller Interface

Task 2: create your assigned diagram here (Time: 17 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

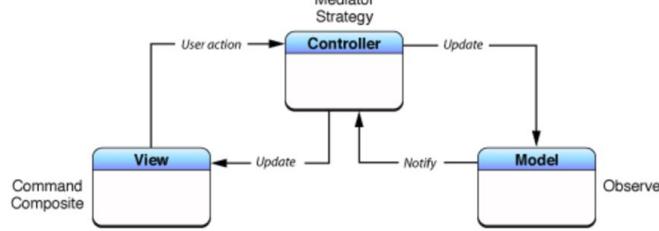
Refresher:

Team Name : Team 4 Specify one: 2b

Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

Ques #1:

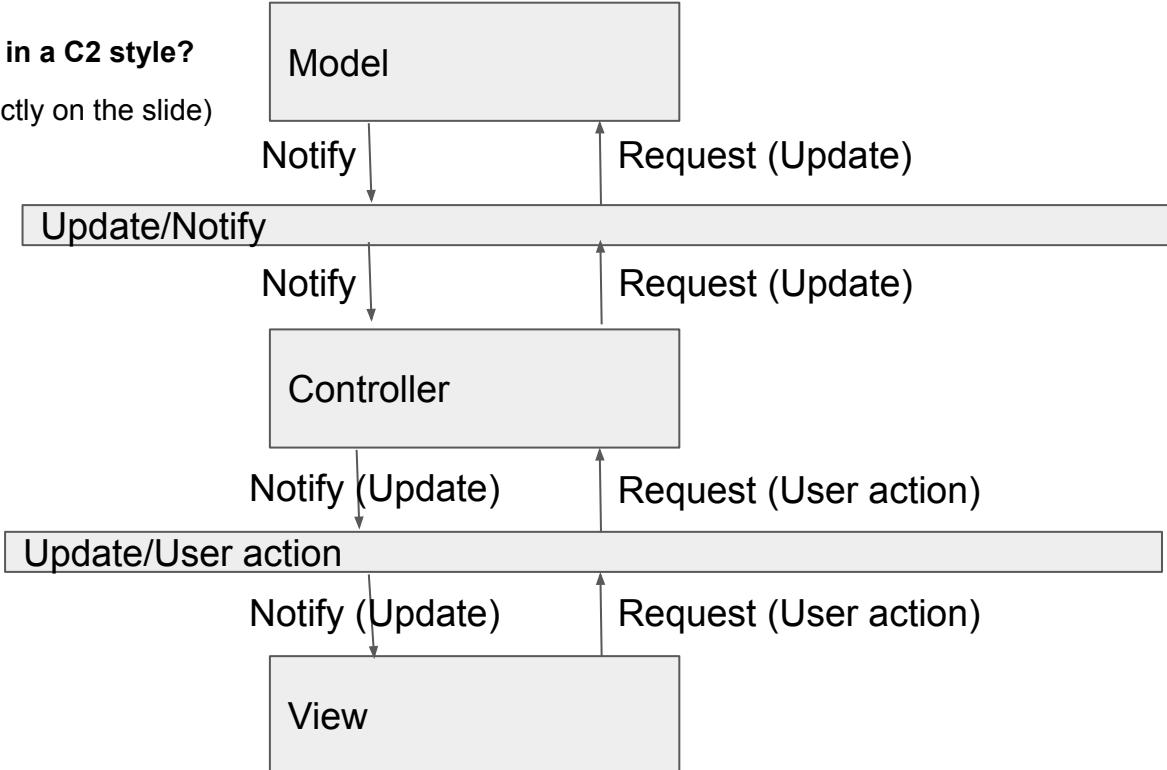
Ques #2:



Task 2: create your assigned diagram here (Time: 17 minutes)

2a, 2b: How do you cast the following MVC in a C2 style?

- Take a picture of your design (or create it directly on the slide)



List Roles and Student names

Moderator :

TimeKeeper: Joshua Medvinsky

NoteTaker:

Reporter:

Team Name :Ludus 2a

List Roles and Student names

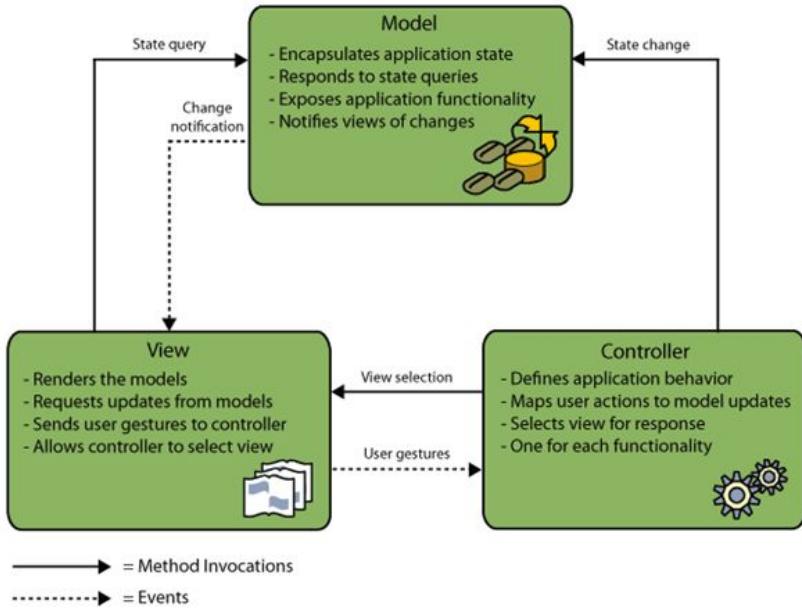
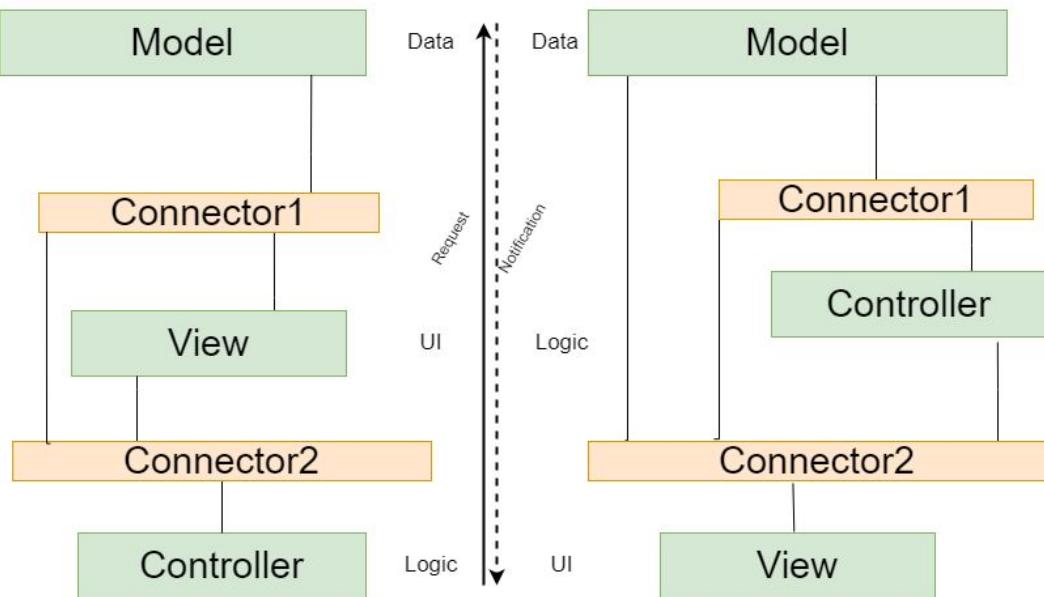
Moderator : Tyson Heo

TimeKeeper: Shaun Stangler

NoteTaker: Agustin Castillo

Reporter: Austin Yao

Diagram Translation vs Actual Implementation



Team Name : <<to add>> Specify one: 2a, 2b or 2c

Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

Ques #1:

Ques #2:

Task 2: create your assigned diagram here (Time: 17 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name : Magratheans 2b

Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

Ques #1:

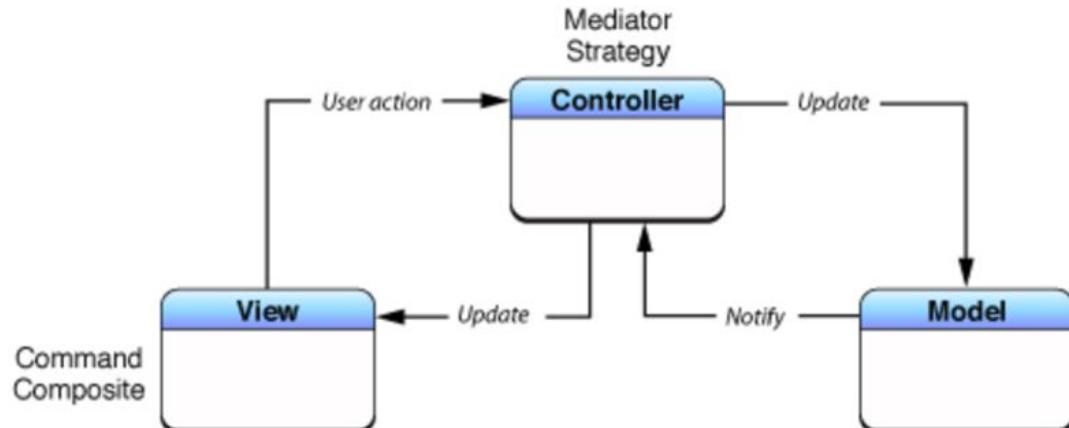
Ques #2:

2b: How do you cast the following MVC in a C2 style?

- Take a picture of your design (or create it directly on the slide)

Task 2: create your assigned diagram here (Time: 17 minutes)

See next slide



List Roles and Student names

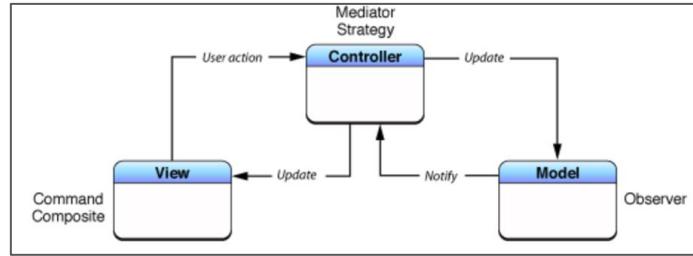
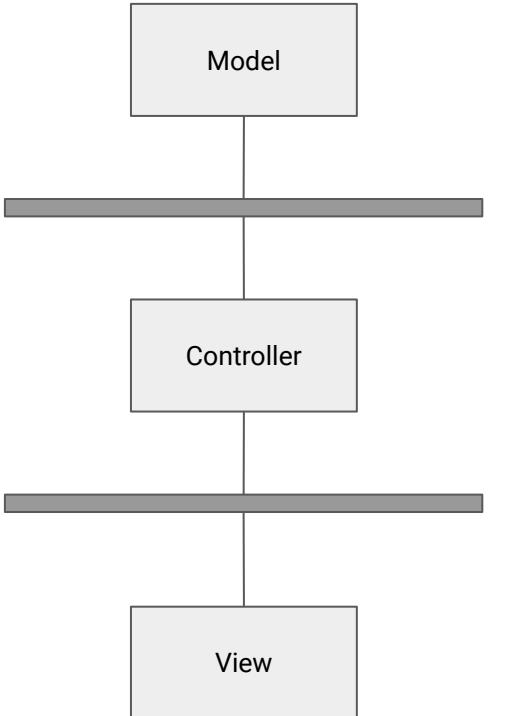
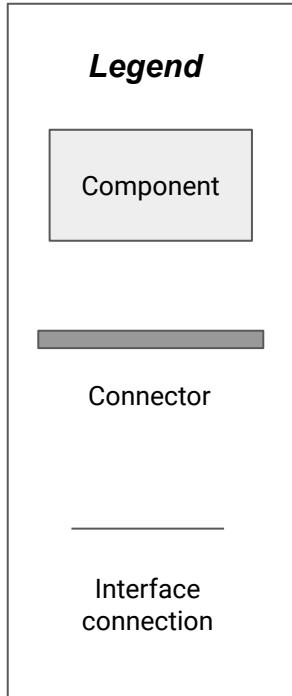
Moderator : Ryan

TimeKeeper: Jason

NoteTaker: Karan

Reporter: Thomas

Magratheans: 2b



"Update" => request
"Notify" => notification

"User action" => request
"Update" => notification

Team Name : Team 9 Specify one: 2c

Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

Ques #1:

Ques #2:

2c: Draw a C2 architecture for a long-distance telecommunications System

Telephones

Local Switches (one per area code)

Long-Distance Switches (communications backbone)

Call Records database

Other embellishments

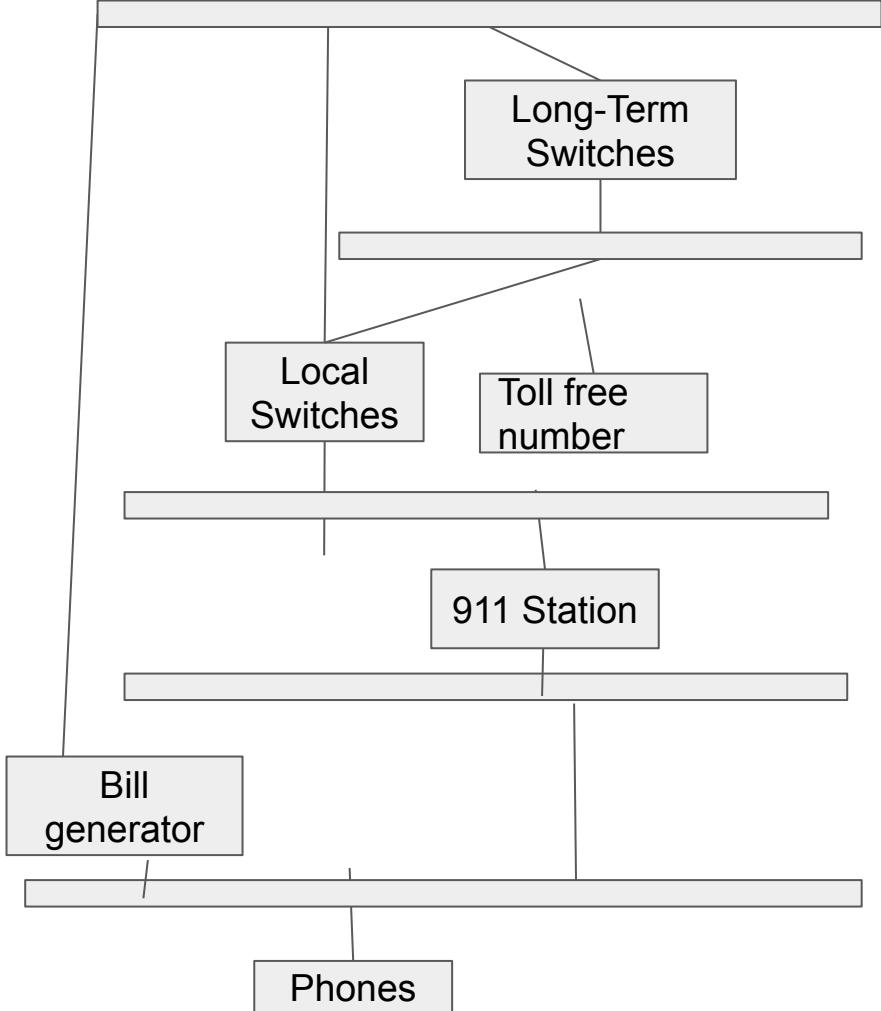
- Toll-Free numbers
- 911 stations
- Operator stations
- Bill generators

List Roles and Student names

Moderator : William

Reporter: Pragati

↑
Request
↓
Notification



data/state

Logic
2

Logic
1

UI

Team Name : <<to add>> Specify one: 2a, 2b or 2c

Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

Ques #1:

Ques #2:

Task 2: create your assigned diagram here (Time: 17 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name : Open Source Web Services Specify one: 2c

Task 1: List questions you have for the Professor about this activity (Time: 3 mins)

Ques #1:

Ques #2:

Task 2: create your assigned diagram here (Time: 17 minutes)

List Roles and Student names

Moderator : yang

TimeKeeper: Sunjil Gahatraj

NoteTaker: Nur Dincer

Reporter:

2c: Draw a C2 architecture for a long-distance telecommunications System

Telephones

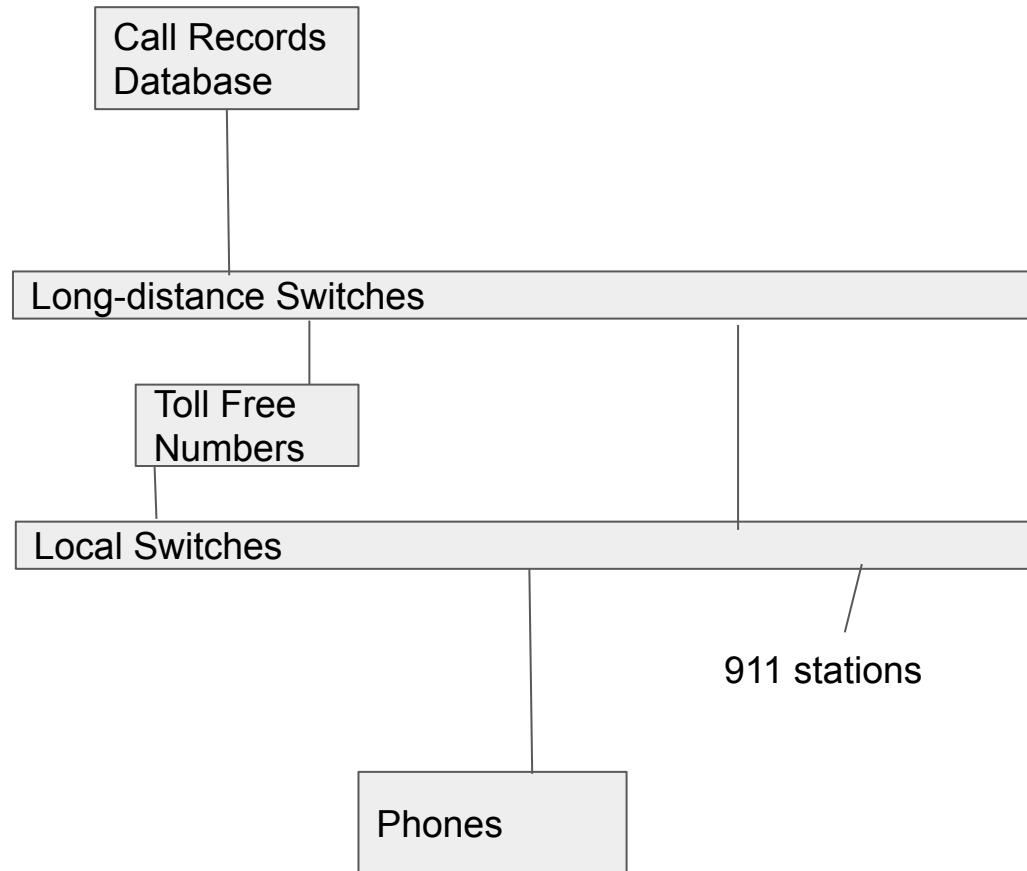
Local Switches (one per area code)

Long-Distance Switches (communications backbone)

Call Records database

Other embellishments

- Toll-Free numbers
- 911 stations
- Operator stations
- Bill generators





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



DISTRIBUTED OBJECTS STYLE

Portions taken and adapted from Richard N. Taylor and Eric Dashofy (UCI)

Heterogeneous Styles



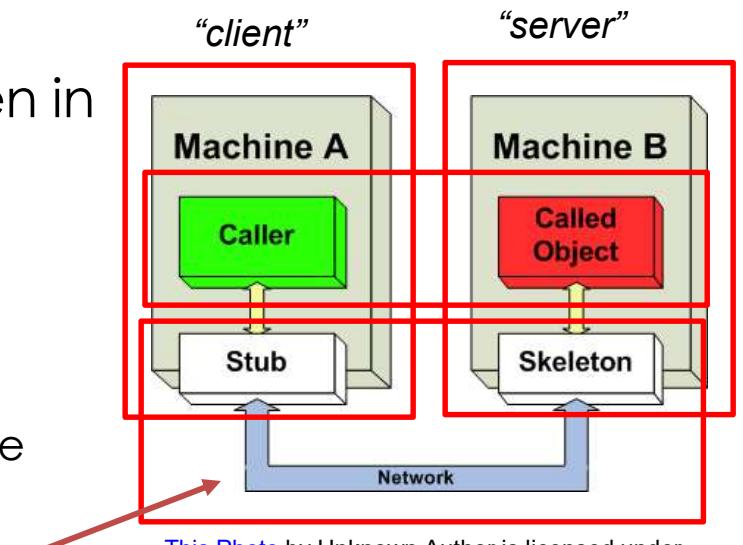
- More complex styles created through composition of simpler styles
- REST
 - Complex history presented later in course
- C2
 - Implicit invocation + Layering + other constraints
- Distributed objects
 - OO + client-server network style *middleware*
 - E.g., Java RMI, CORBA, Microsoft DCOM/ActiveX/.Net



Distributed Objects



- Objects
 - Coarse- or fine-grained
 - Run on heterogeneous hosts, written in heterogeneous languages
 - Components
 - Objects
 - Connectors
 - Remote method invocation (aka remote procedure call)
 - Data elements
 - Parameters of methods, return values, and exceptions
 - Topology
 - Graph of objects from callers to callees



[This Photo](#) by Unknown Author is licensed under
[CC BY-SA](#)

Distributed objects



- Typical uses
 - Distributed systems composed of components running on different hosts
- Caution
 - Mostly synchronous interactions
 - Applications that use middleware have to abide by distributed objects style
 - Difficulty with streams and high volume data flow
- Qualities
 - Strict separation of implementation from interfaces
 - Mostly transparent interoperability across location, platform, and language boundaries

Distributed objects: CORBA



Common Object Request Broker Architecture (CORBA)

- Standard for implementing middleware
- Popular in 1990s
- Provides interfaces in terms of programming language and IDL (interface definition language)
- IDL
 - independent of programming language
 - maps to popular languages (e.g., C, C++, Java, Ruby, Python, etc)

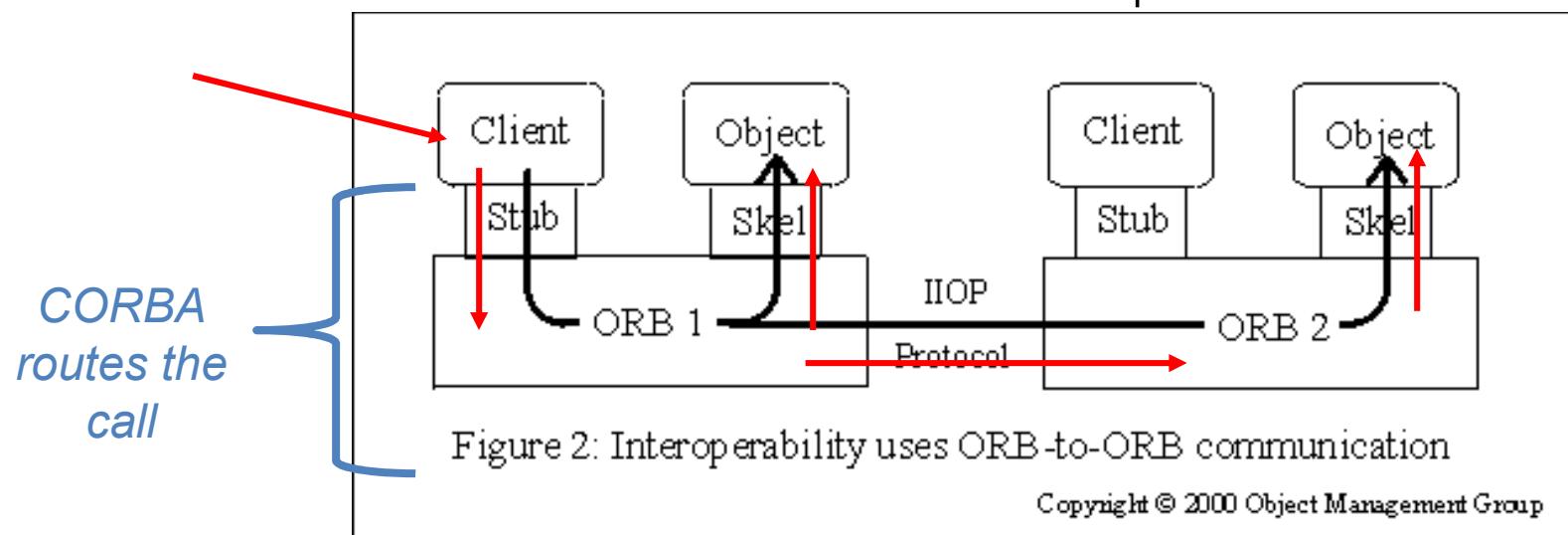
<https://www.omg.org/spec/CORBA/>

<https://www.corba.org/>

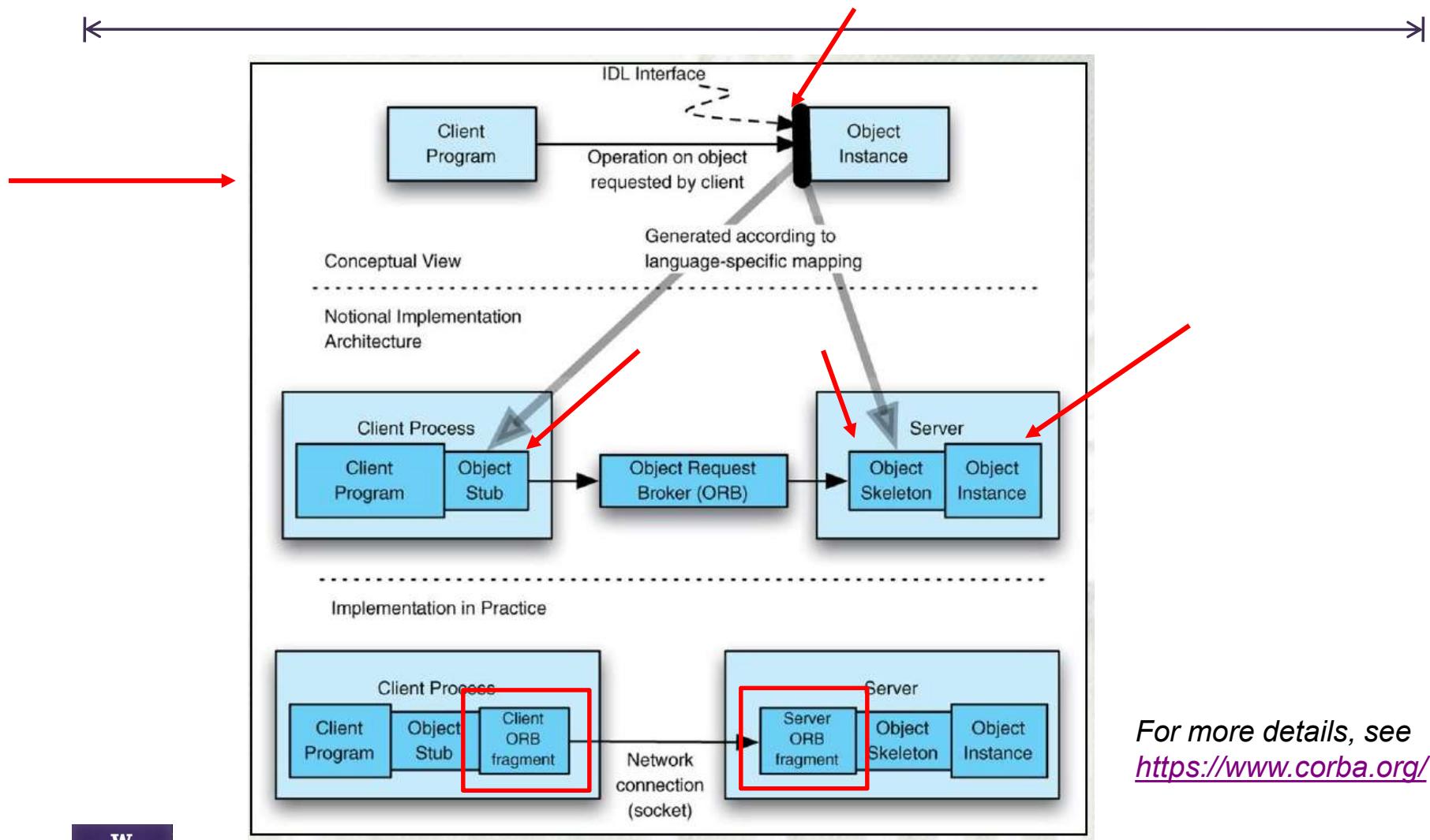
Distributed Objects: CORBA



- Basic idea
 - Client makes a procedure call
 - CORBA middleware routes the call to the appropriate component, or object, that provides that service
 - Client may invoke a procedure call on a remote object, but the middleware makes the call transparent



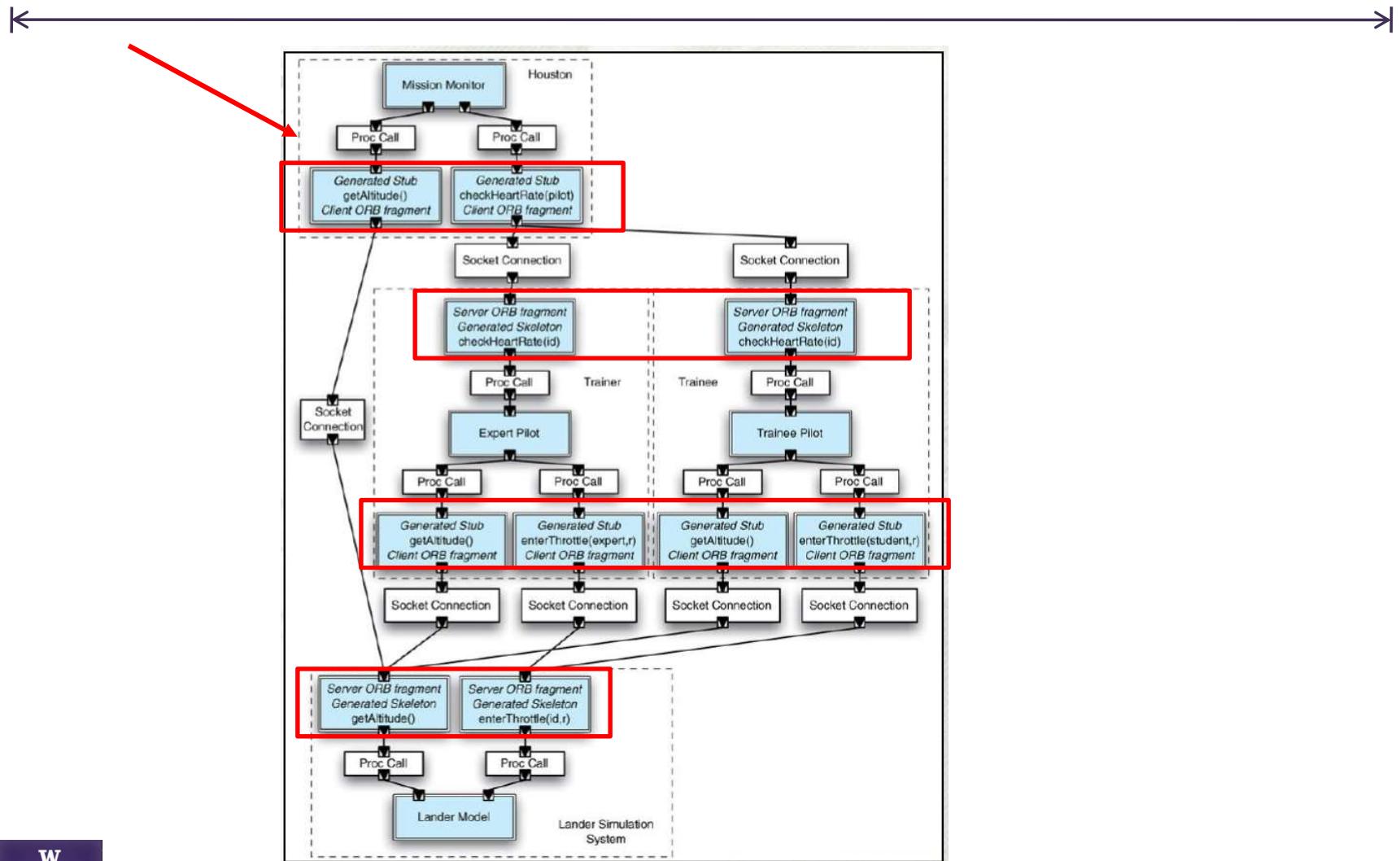
CORBA Concept and Implementation



For more details, see
<https://www.corba.org/>



CORBA Lunar Lander



CSS 553, Spring 2023, 7a Distributed Objects

9



Questions



- Compare/contrast these heterogeneous styles. How are they similar? Different?
- What benefits do we get from using patterns or styles?
- Section 4.3.6 in the book mentions “freedom of constraints”. What does this mean?
- Which of these heterogeneous styles use implicit invocation? Explicit invocation?
- Pick one REST principle and explain what it means
- Pick one of these heterogeneous style and specify **one** drawback
- Once we created an architecture diagram, are we done with designing the system? Explain.



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



HETEROGENEOUS STYLES: INTRO TO REST

Fallacies of Distributed Systems



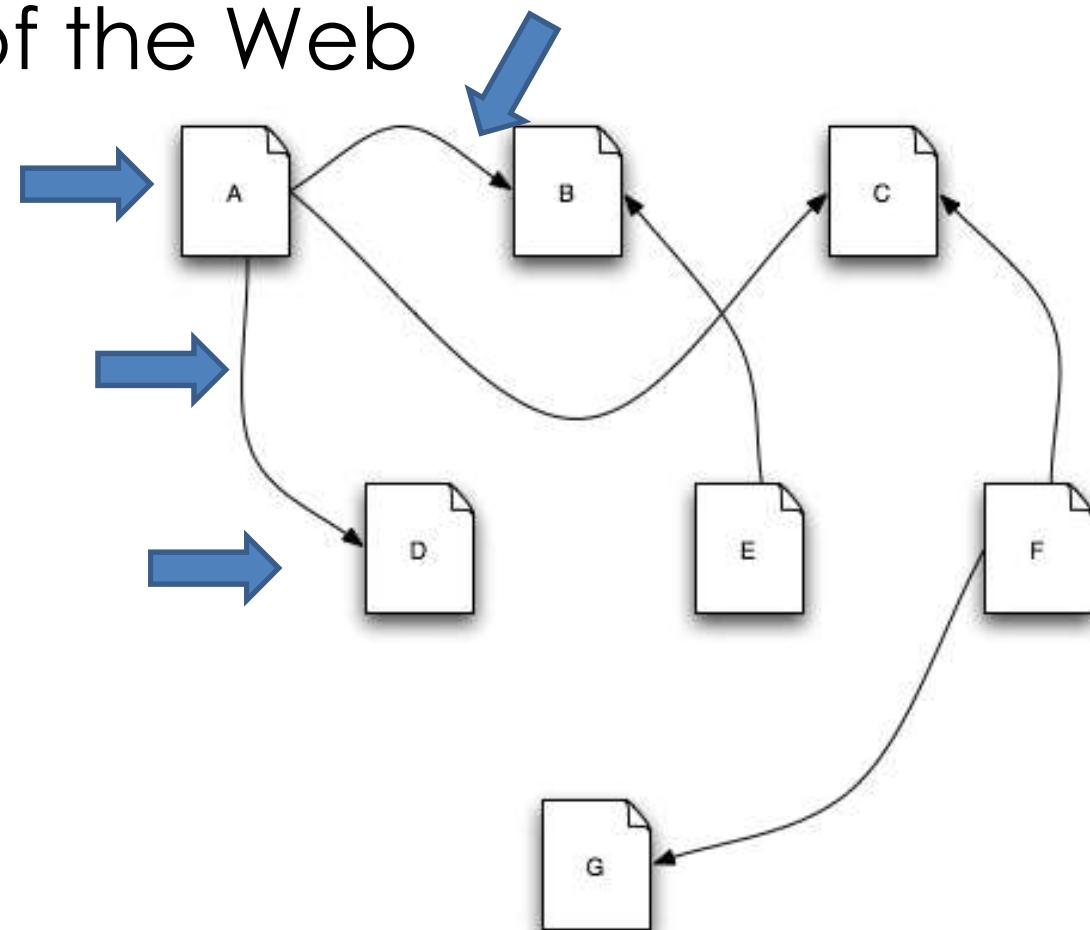
- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

-- Deutsch & Gosling

Architecture in Action: WWW

← →

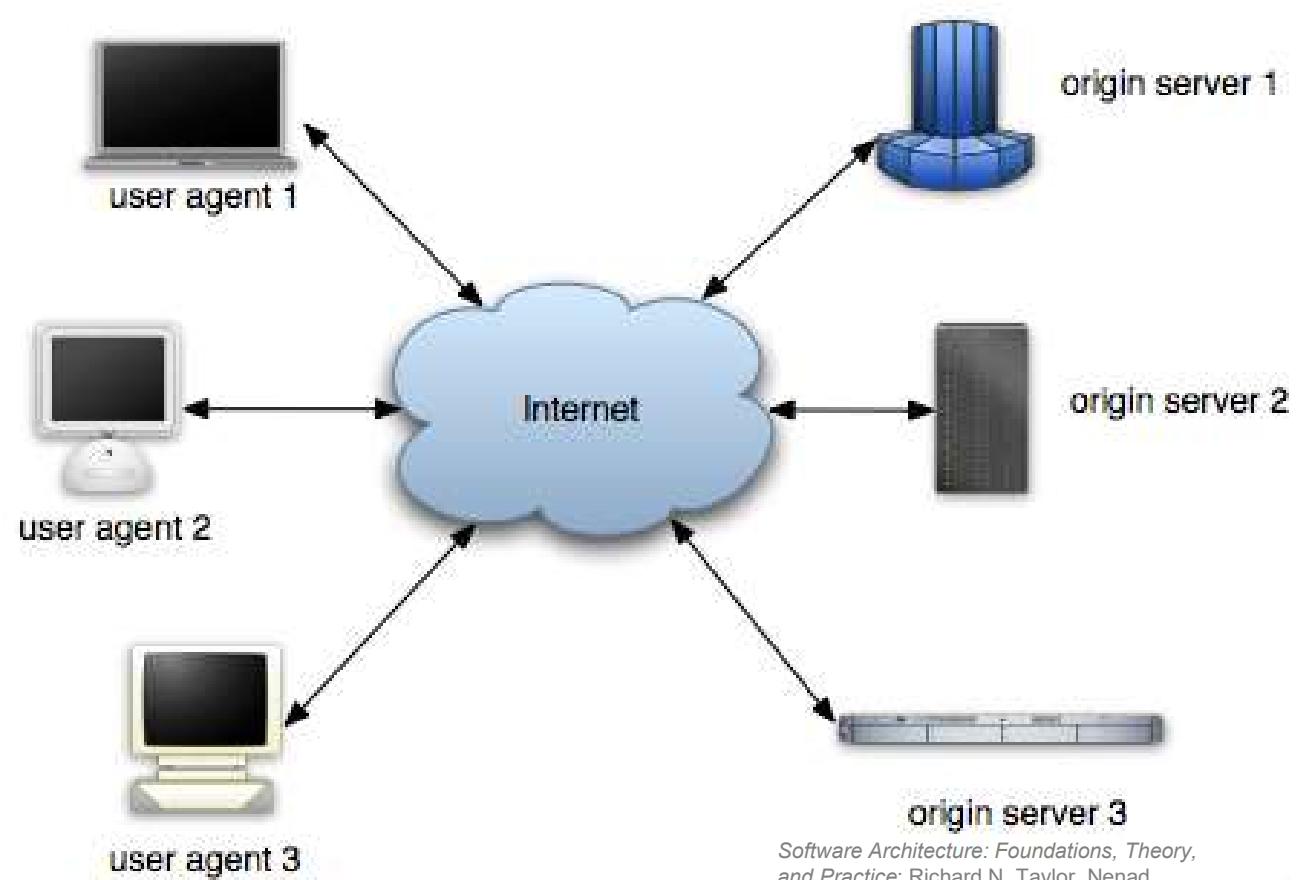
- One view of the Web



Architecture in Action: WWW (cont'd)



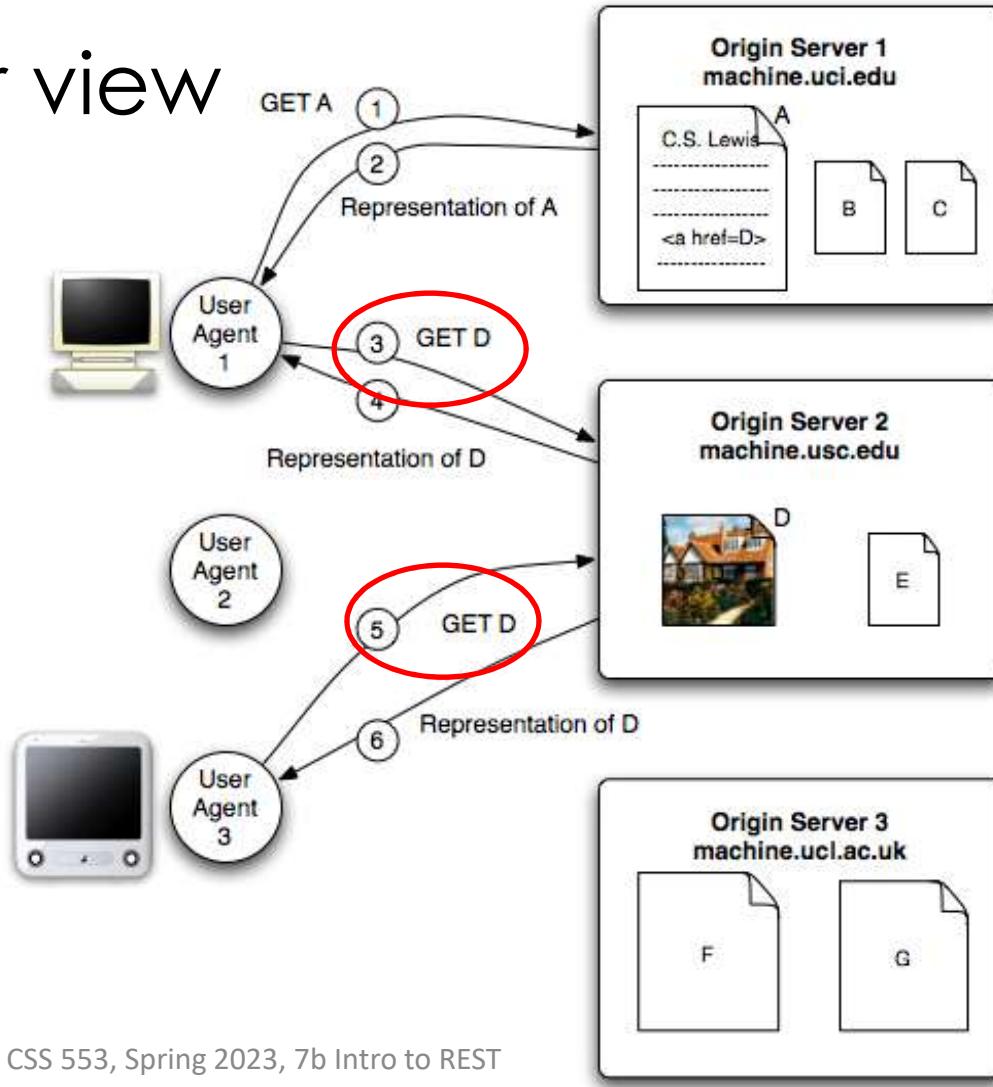
- Another view



Architecture in Action: WWW

← →

- And another view



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

WWW Requirements



- “Web’s major goal was to be a shared information space through which people and machines could communicate” [Berners-Lee 1996]
- Low entry-barrier
- Extensibility
- Internet-Scale
 - Anarchic scalability
 - Independent deployment
- Distributed Hypermedia
- Evolving Requirements

What Basic Arch Styles are in REST?



What Basic Arch Styles are in REST?



- Client-server
 - Separation of concerns
 - NFPs: Portability, scalability
 - “Context-Free”
 - NFPs: Visibility, reliability, scalability
 - Tradeoff?
 - Decreased network performance, less server control

What Basic Arch Styles are in REST?



- Layered
 - Substrate independence – knowledge at a single layer
 - NFP? Scalability
 - Tradeoff? Add overhead and latency

What Basic Arch Styles are in REST?



- Code-on-demand (variant of Mobile Code)
 - Extend client functionality – download code / scripts
 - NFP: extensibility
 - Tradeoff? Reduce visibility

Other constraints:

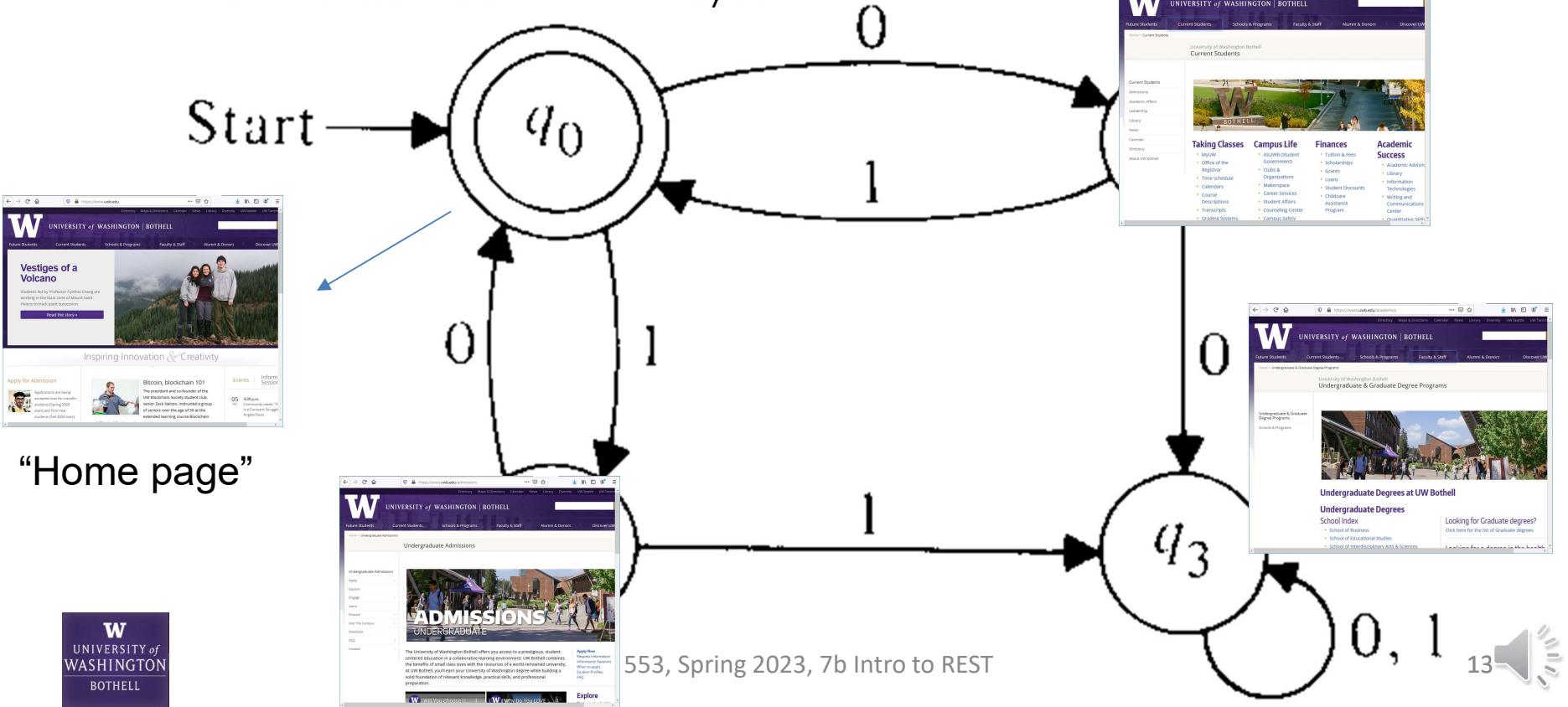


- Cache
 - NFPs: efficiency, scalability, user-perceived performance
 - Tradeoff?
 - Decrease reliability
- Uniform interface – generality
 - NFPs? Evolvability
 - Tradeoff?
 - Degrade efficiency

REST



- REpresentational State Transfer
- Style of modern web architecture
 - One instance of REST style



WWW's Architecture



- **The application is distributed (actually, decentralized) hypermedia**
- Architecture of the Web is wholly separate from the code
- There is no single piece of code that implements the architecture.
- There are multiple pieces of code that implement the various components of the architecture.
 - E.g., different Web browsers
- Stylistic constraints of the Web's architectural style are not apparent in the code
 - The effects of the constraints are evident in the Web
- One of the world's most successful application is only understood adequately from an architectural vantage point.





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



REST

Identify the elements



- Data?
- Components?
- Connectors?

REST — Data Elements



- Data is a key aspect of REST
- Three options – distributed hypermedia:
 - Render the data where it is located and send a fixed-format image to the recipient;
 - Encapsulate the data with a rendering engine and send both to the recipient;
 - Send the raw data to the recipient along with metadata that describes the data type, so that the recipient can choose their own rendering engine.
- REST – hybrid of all three

REST — Data Elements



- Resource
 - Key information abstraction
- Resource ID: URL, URN
- Representation HTML doc, JPG image
 - Data plus metadata
- Representation metadata: media type, last modified time
- Resource metadata: source link, alternates
- Control data cache-control
 - e.g., specifies action as result of message

REST — Components



- User agent
 - e.g., browser
- Origin server
 - e.g., Apache Server, Microsoft IIS
- Proxy
 - Selected by client
- Gateway
 - Squid, CGI, Reverse proxy
 - Controlled by server

REST — Connectors



- client libwww, libwww-perl
- server libwww, Apache API, NSAPI
- cache browser cache, Akamai cache network
- resolver bind (DNS lookup library)
- tunnel SOCKS, SSL after HTTP CONNECT

REST Principles



- [RP1] The key abstraction of information is a resource, named by an URL. Any information that can be named can be a resource.
- [RP2] The representation of a resource is a sequence of bytes, plus representation metadata to describe those bytes. The particular form of the representation can be negotiated between REST components.
- [RP3] All interactions are context-free: each interaction contains all of the information necessary to understand the request, independent of any requests that may have preceded it.

REST Principles (cont'd)



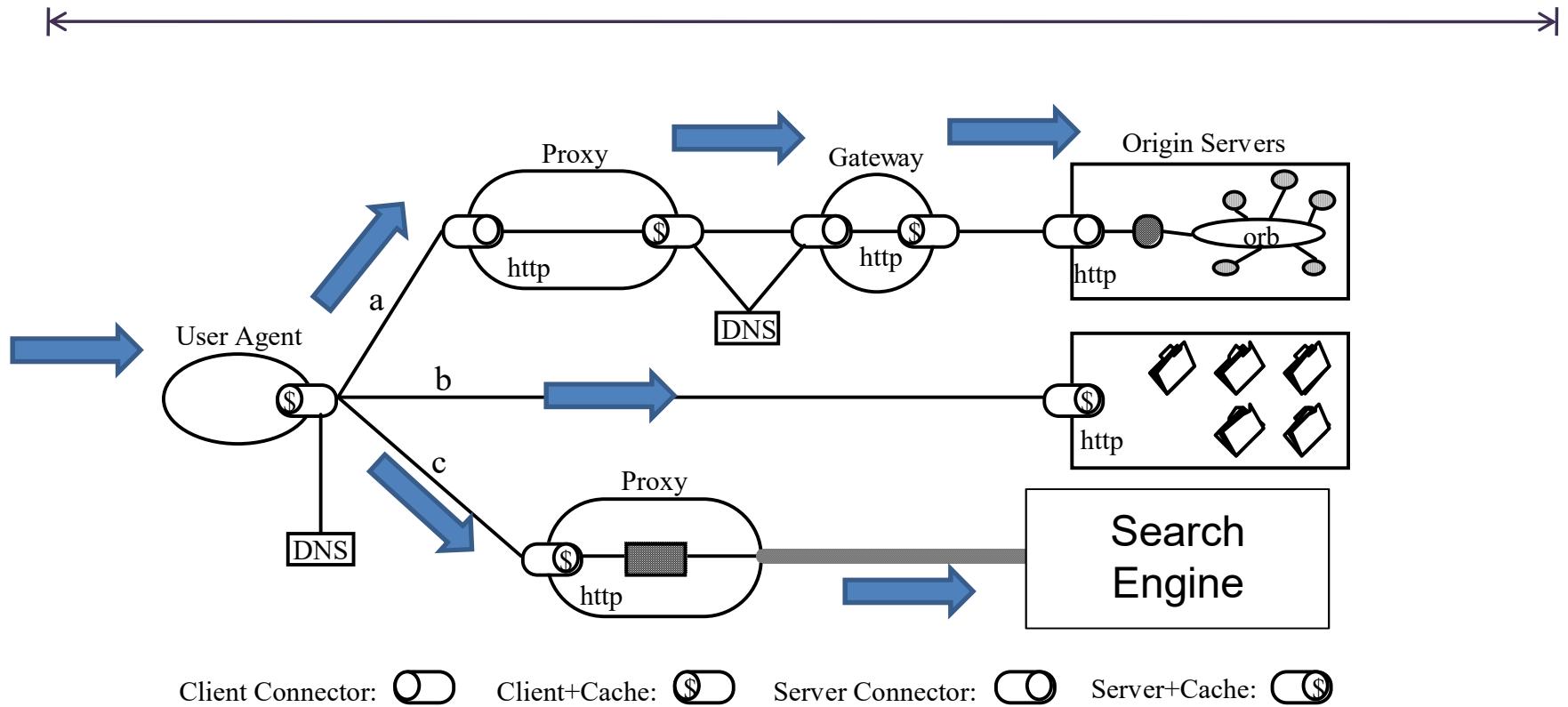
- [RP4] Components perform only a small set of well-defined methods on a resource producing a representation to capture the current or intended state of that resource and transfer that representation between components. These methods are global to the specific architectural instantiation of REST; for instance, all resources exposed via HTTP are expected to support each operation identically.

REST Principles (cont'd)



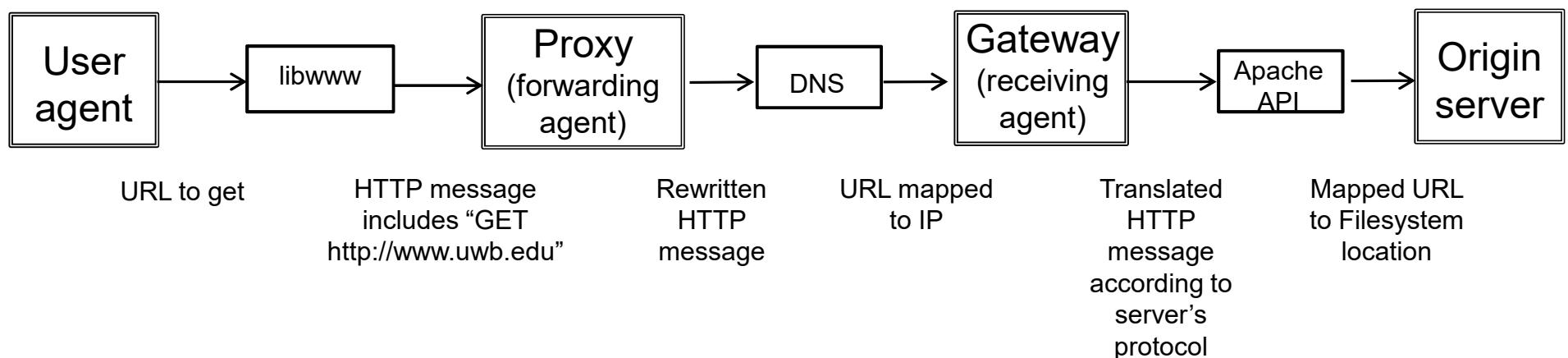
- [RP5] Idempotent operations and representation metadata are encouraged in support of caching and representation reuse.
- [RP6] The presence of intermediaries is promoted. Filtering or redirection intermediaries may also use both the metadata and the representations within requests or responses to augment, restrict, or modify requests and responses in a manner that is transparent to both the user agent and the origin server.

An Instance of REST



REST (underlying HTTP 1.1)

- Communication takes place over TCP/IP connections
 - TCP 80 (default)
- See REST-Style Architecture handout on course page
- Data flow view:



View HTTP Request and Response: <http://testuri.org/>

WWW: Architectural degradation?



Examples of Architectural Degradation

- 
- URI
 - Embed user ids in the URI – track user interactions on the server
 - Shared caching becomes ineffective
 - Reduce server scalability
 - Undesirable effects when user shares URI with another user
 - HTTP
 - Cookies – data assigned by the origin server to the user agent; to be included by the user agent in future requests
 - Back button will not match the stored state in the cookie
 - Security and privacy

Q&A



- Does REST use URLs?
 - Yes, “resource” concept in the URI standard was taken from REST
 - Table - only shows specific examples of URI
- Can REST run over UDP?
 - The implementation of REST (HTTP/1.1) can run on any protocol as long as it is reliable (see <http://www.ietf.org/rfc/rfc2616.txt>)
 - UDP (Universal/User Datagram protocol) is fast, but unreliable
 - TCP is slow but reliable. TCP/IP usually used by HTTP
 - Reliable UDP may be used

Q&A



- What is the purpose of the client and server side connectors?
 - See slide 7 and the reading

Q&A



- Is gateway located on the same machine as origin server?
 - It could be, but not necessarily
 - Gateway is like a reverse proxy which redirects requests
 - Gateway hosted on different machine, especially if server is behind a firewall (i.e., not accessible to the Internet)

READ THIS FIRST - 20 minutes

Topic: Architecture Styles

This activity is designed to

- Give you practice in designing
- Tie up loose ends regarding upcoming assignment(s)

Roles:

• **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.

• **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.

• **Note Taker** - takes record of the group’s discussion in the Google doc (below).

• **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity or the assignments due tonight, type in Red text

All teams will do the following task:

Previous design exercise

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

- ...using whatever method you like...
- ...using whatever architectural style / pattern we covered thus far in class...
- Cost is a concern...and this is the final design that you will hand off to your development team

Team Name: Dawn

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: To be decided

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Previous design exercise

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches Traffic light - Camera - Traffic lights control system - load balancing server
- Functional for at least 5 years

Create a design for this software...

- ...using whatever method you like...
- ...using whatever architectural style / pattern we covered thus far in class...
- ...Cost is a concern...and this is the final design that you will hand off to your development team

List Roles and Student names

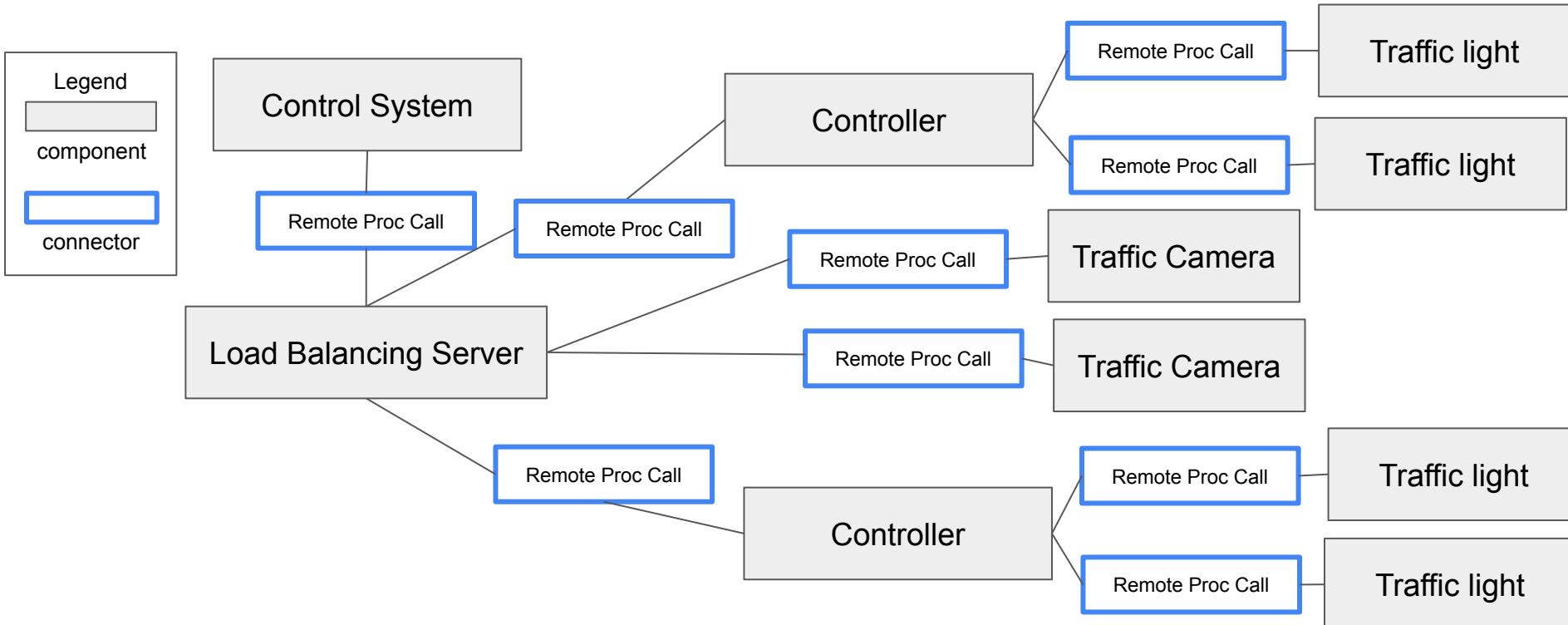
REST style; its standardized and general interfaces allows for extendibility (something that would remain in tact for over 5 years).

We want to implement load balancing for the sake of facilitating the higher-order more complex interaction in terms of the flow of control

- prevents the control system from being overwhelmed by a specific stream of data

Remote Procedure Calls coordinate, communicate, and facilitate the data

- each set of traffic lights have a centralized controller



Team Name: Team 4

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

...using whatever method you like...

...using whatever architectural style / pattern we covered thus far in class...

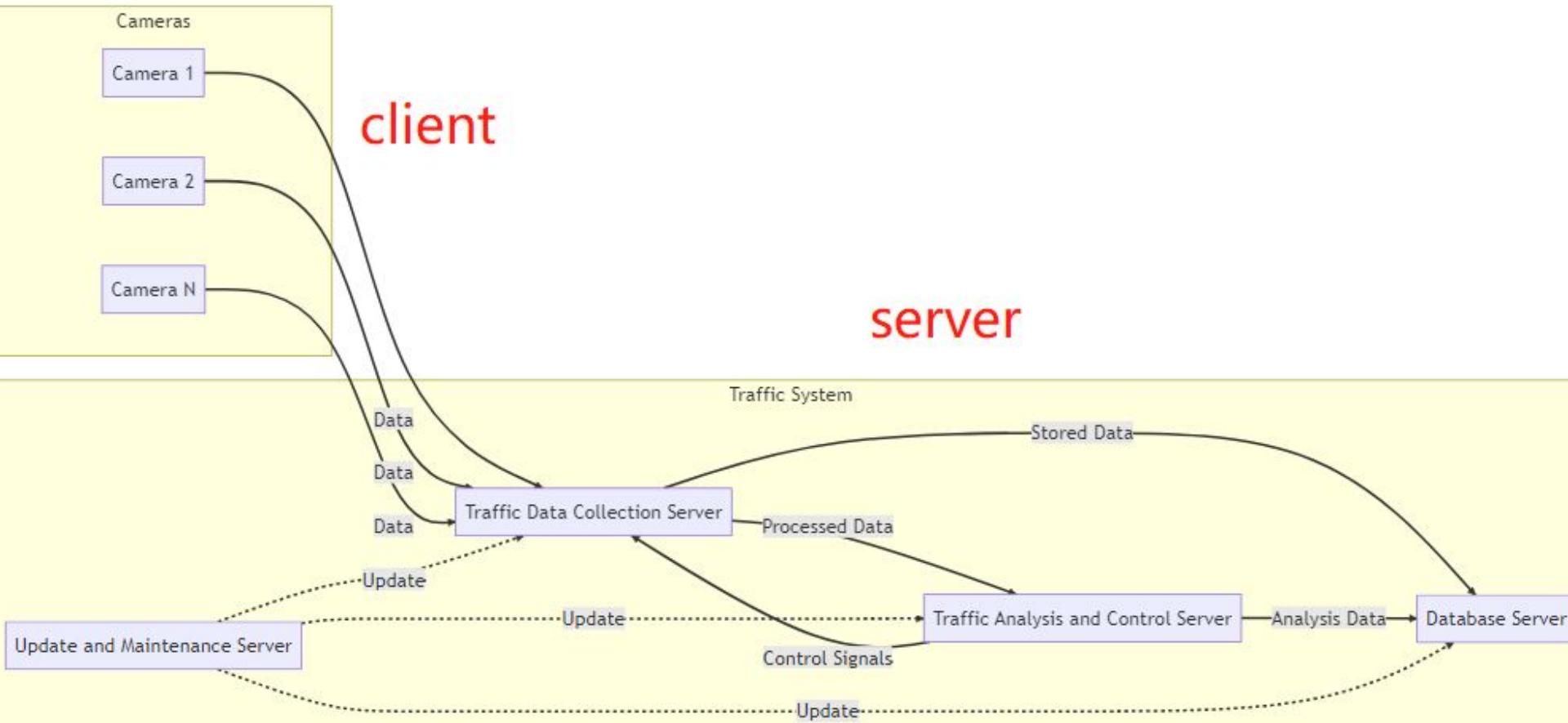
...Cost is a concern...and this is the final design that you will hand off to your development team

Functional for 5 years

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Sense-Compute-Control for real-time part

Robust



Team Name: Ludus

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Not C2

Module based

Event - Sensor

Overall Control

P2P

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Magratheans

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

- ...using whatever method you like...
- ...using whatever architectural style / pattern we covered thus far in class...
- ...Cost is a concern...and this is the final design that you will hand off to your development team

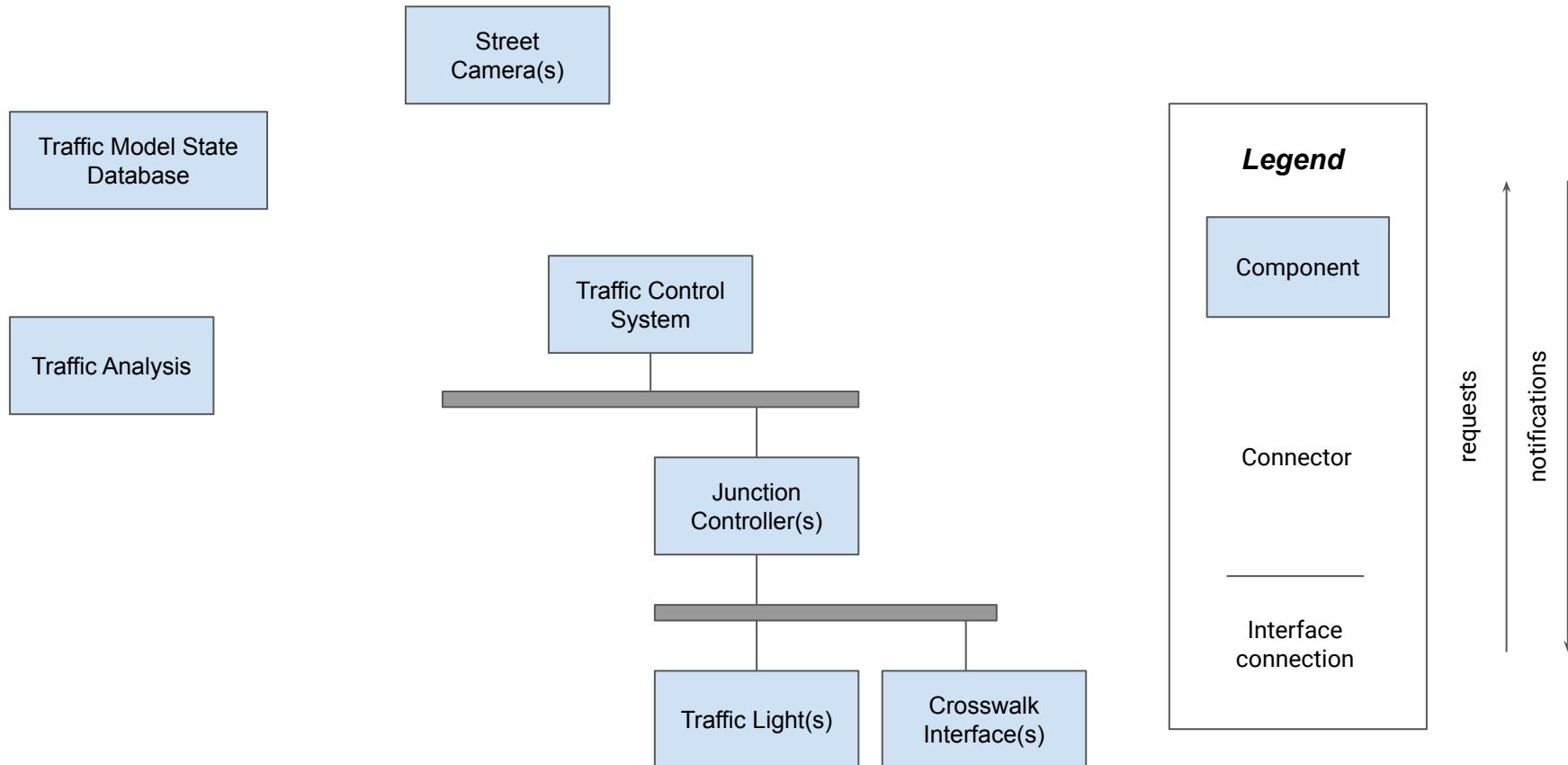
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:



Team Name: Team 9

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Using C2 Architecture

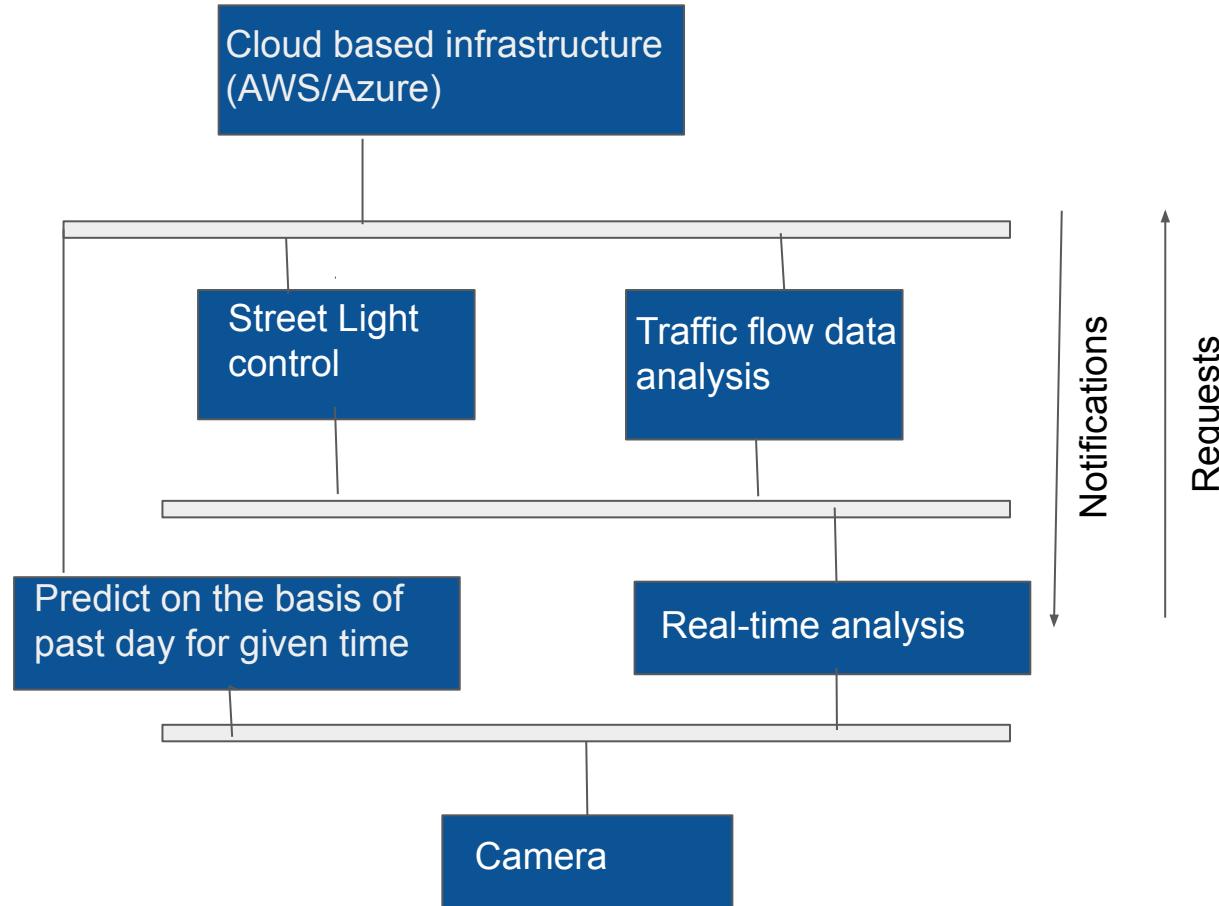
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Open Source Web Services

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

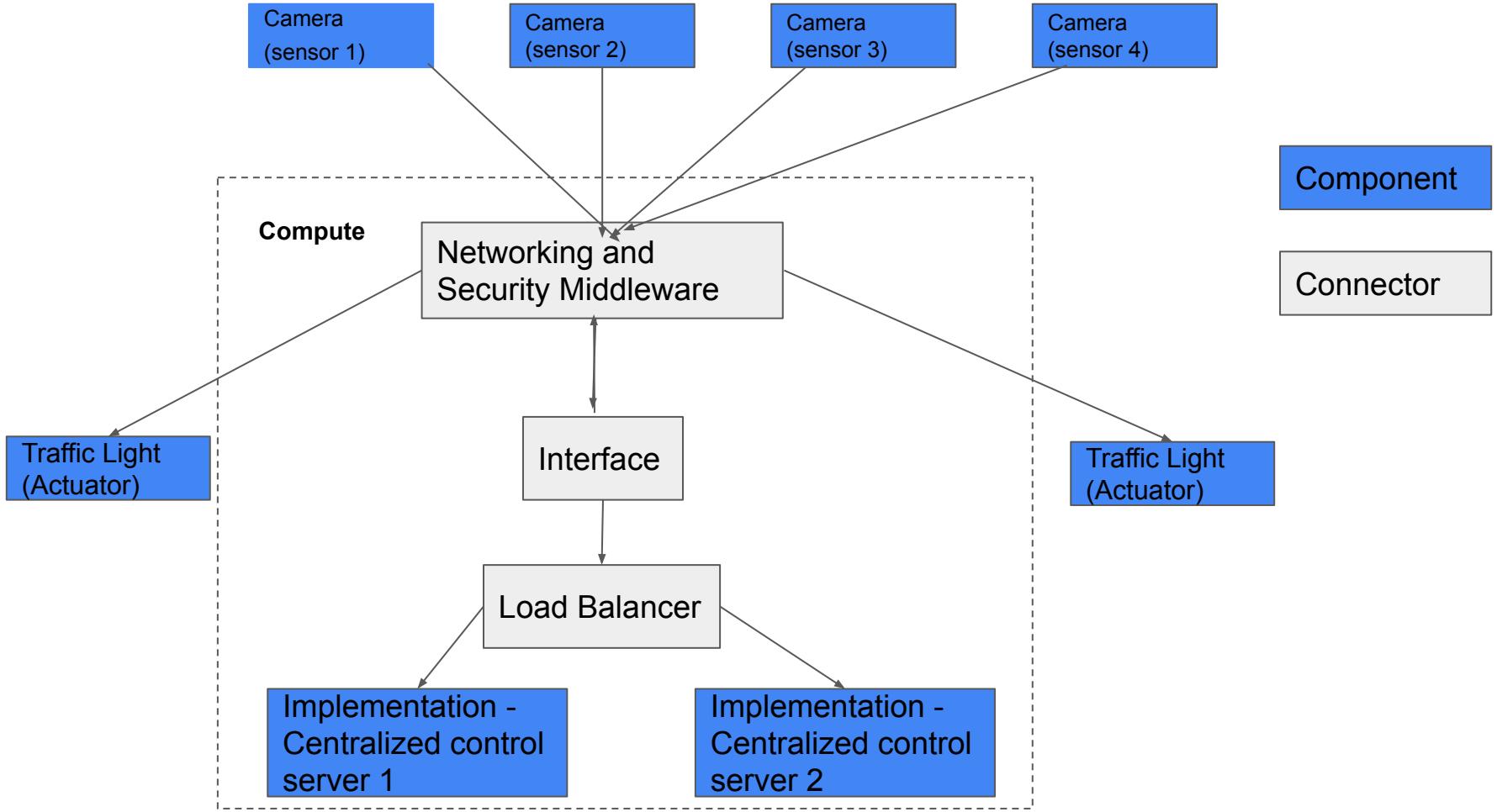
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Agenda



- Overview
- Q&A
- REST derivation
- G3 Q&A
- -----
- Design Presentation
- Previous activity

Overview



- Distributed Objects
- REST

Questions



- Compare/contrast these heterogeneous styles.
 - C2
 - REST
 - Distributed Objects (OO with Middleware)
 - How are they similar? **Dawn** 1) can be used to design distributed systems, 2) C2 & Distributed Objects – Modularity and separation of concerns, 3) C2 & REST – stateless – requests need to contain all info
 - Different? **The Boffins** 1) Distributed objects – synchronous, REST asynchronous, C2 strict request/notification pattern, 2) distributed objects – multiple hosts, REST – client-server relationship, but could involve multiple machines in between, 3) REST – strictly defined & limited interfaces (standard interfaces), C2 – can define own messages, Distributed objects – can define own interfaces (can extend interfaces), interfaces are tightly coupled
- What benefits do we get from using patterns or styles? **To be decided**
 - Captures wisdom of architects – embodies best practices for any project
 - Promotes modular design
 - Component & code reusability
 - Better communication among development team – well defined patterns
 - Defined constraints – limit solution – allow experiment within the style

Questions



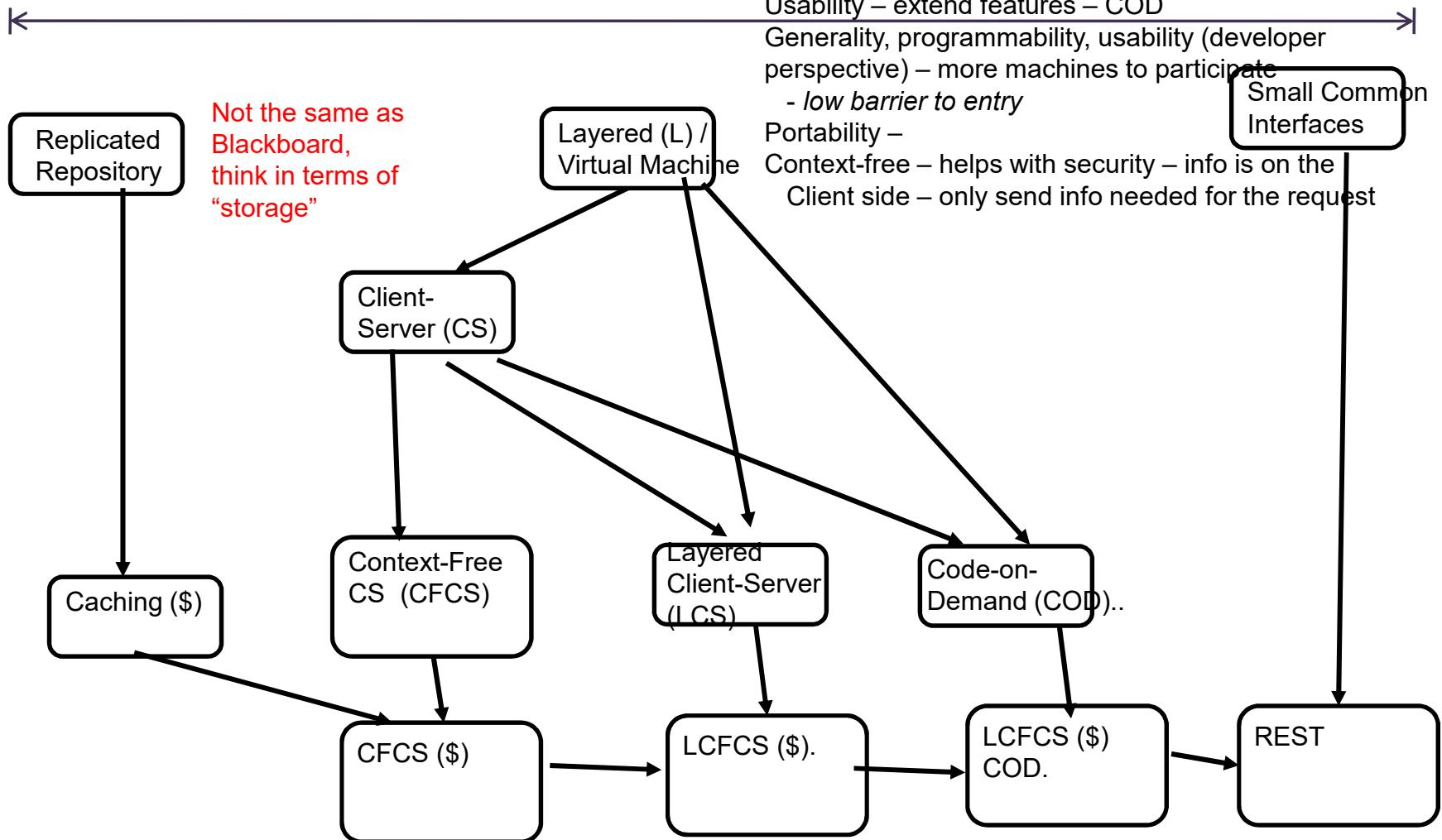
- Section 4.3.6 in the book mentions “freedom of constraints”. What does this mean? **Team 4**
 - Place constraints on the architecture – **freedom on what?**
 - Provide more certainty – interaction between components
- ~~Which of these heterogeneous styles use implicit invocation? Explicit invocation?~~
- Pick one REST principle and explain what it means **Ludus**
 - All interactions are context free – all requests contain all info needed – more flexible (any machine, any hardware/software combination); reliability – something goes down, spin up another server and pick up anything that goes down; caching is not needed – don’t need to store
- Pick a basic arch style used by REST **Team 9**
 - Client-Server – clients request, servers reply
 - Layered architecture – API, internal logic – return back to user – multiple machines involved in a request

Questions



- Pick one of these heterogeneous style and specify **one** drawback **The Magratheans**
 - REST – overhead trying to adhere to the standard interfaces – example a networking application – if use a specialized connection, can better use bandwidth - how data is packaged and size of data
- Once we created an architecture diagram, are we done with designing the system? Explain. **Open Source Web services**
 - No, arch design is not a single step, from implementation to deployment
 - As-implemented can be different from as-intended, depending on needs and problems that can come back

Derivation of REST



Derivation of REST

Requirements

Share info

Low barrier to entry

Extensibility

Internet scale

- anarchic scalability

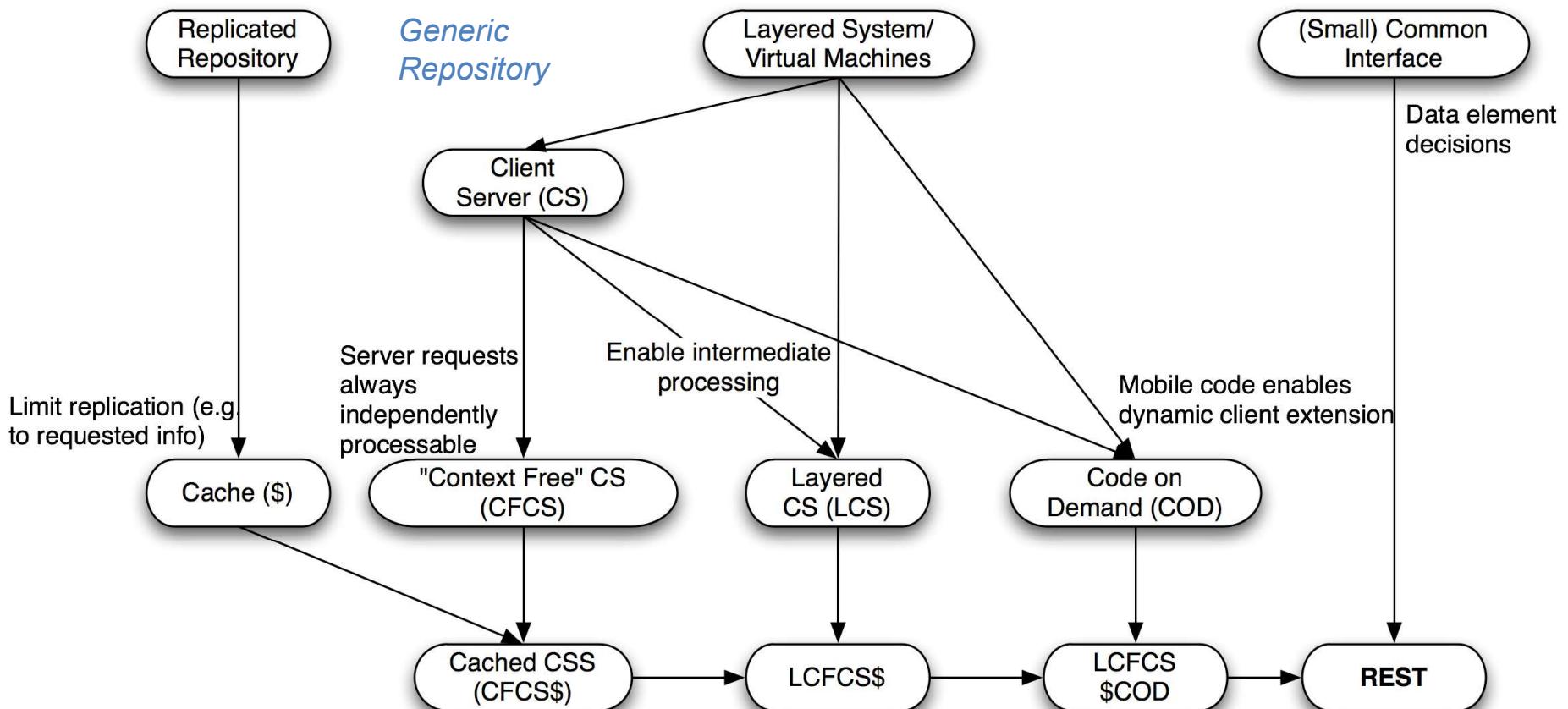
- independent deployment

Distributed hypermedia

Evolving requirements

Small number of common interfaces demanded of all participants

Multiple equivalent sources promote efficiency and robustness



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

ACTIVITIES

Design Exercise #2



- Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion. Create a design for this software...
- ...using whatever method you like...
- ...Cost is a concern. Make sure that the cost of implementing your design is reasonable...
- Provide your answer in the Google Slide

Reflection Questions



- How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?
- Who did you keep in mind when making your design?
- What was your goal with your design?
- Did you have more than one goal?
- Did you reach the goal(s) with your design?

Design Exercise #3



- Improve your software design with the following new requirements. It must be robust (i.e. handle hardware malfunctions). It must also require minimal downtime for software fixes / patches. It must be functional for at least 5 years.
- ...using whatever method you like...
- ...using whatever architectural style / pattern ...
- ...Cost is a concern, and this will be your final design that you will hand off to your development team...
- ...do this exercise with your group...



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



INTRO TO CONNECTORS

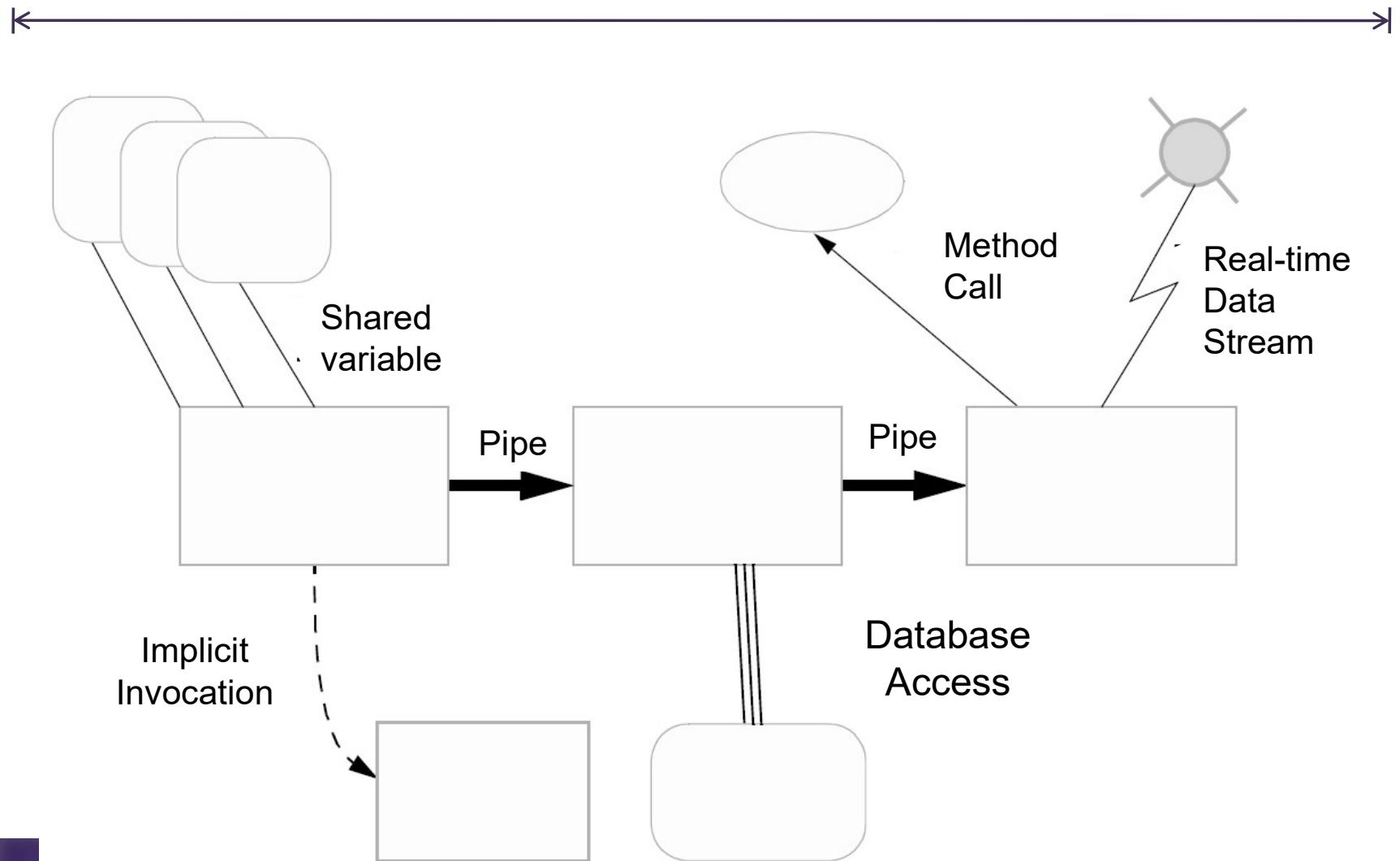


What is a Software Connector?



- Architectural element that models
 - Interactions among components
 - Rules that govern those interactions
- Simple interactions
 - Procedure calls
 - Shared variable access
- Complex & semantically rich interactions
 - Client-server protocols
 - Database access protocols
 - Asynchronous event multicast
- Each connector provides
 - Interaction duct(s)
 - Transfer of control and/or data

Where are Connectors in Software Systems?



Off-the-shelf Connectors



- Commercial & Research
 - CORBA
 - COM/DCOM
 - ActiveX
 - Java RMI
 - HP SoftBench
 - Q
 - Tooltalk
 - etc.

For more info, see Medvidovic, N.,, Dashofy, E.M., Taylor, R.N., Employing Off-the-Shelf Connector Technologies in C2-Style Architectures, In Proc of the California Software Symposium (CSS'98), pages 21-30, Irvine, CA, October 23, 1998.



Implemented vs. Conceptual Connectors



- Connectors in software system implementations
 - Frequently no dedicated code
 - Frequently no identity
 - Typically do not correspond to compilation units
 - Distributed implementation
 - Across multiple modules
 - Across interaction mechanisms

Implemented vs. Conceptual Connectors (cont'd)



- Connectors in software architectures
 - First-class entities
 - Have identity
 - Describe all system interaction
 - Entitled to their own specifications & abstractions

Reasons for Treating Connectors Independently



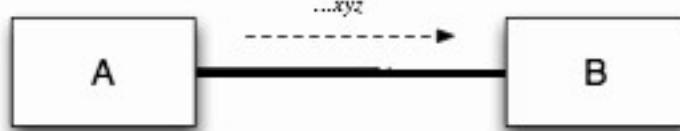
- Connector ≠ Component
 - Components provide application-specific functionality
 - Connectors provide application-independent interaction mechanisms
- Interaction abstraction and/or parameterization
- Can facilitate system properties
 - Performance, resource utilization, global rates of flow, scalability, reliability, etc.

Benefits of First-Class Connectors



- Separate computation from interaction
- Minimize component interdependencies
- Support software evolution
 - At component-, connector-, & system-level
- Potential for supporting dynamism
- Facilitate heterogeneity
- Become points of distribution
- Aid system analysis & testing

An Example of Explicit Connectors

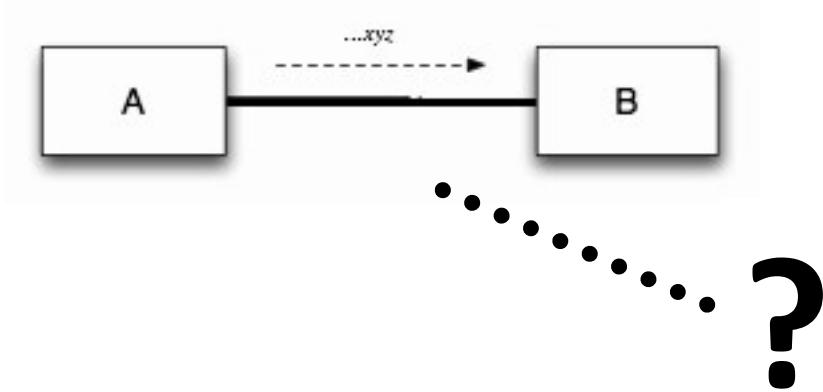


Change interaction

- discrete, typed packets
- bidirectional transfer
- get acknowledgements for each transfer.
- add another component

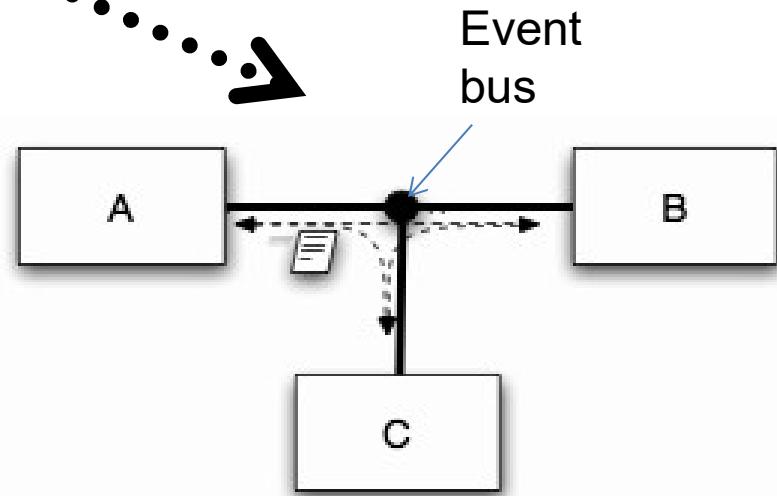


An Example of Explicit Connectors (cont'd)



Change interaction

- discrete, typed packets
- bidirectional transfer
- get acknowledgements for each transfer.
- add another component





CSS 553, Spring 2023, 8a IntroConnectors





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



CONNECTORS



Software Connector Roles (Services)



- Roles
 - Communication
 - Transmission of data
 - Coordination
 - Exchange of control
 - Conversion
 - Transform interactions to remedy heterogeneity
 - Facilitation
 - Streamline interactions for interoperation

Connectors as Communicators



- Main role associated with connectors
- Transfer data
- Supports
 - Different communication mechanisms
 - e.g. procedure call, RPC, shared data access, message passing
 - Constraints on communication structure/direction
 - e.g. pipes
 - Constraints on quality of service
 - e.g. persistence

Connectors as Coordinators



- Transfer of control
- Orthogonal to communication, conversion, and facilitation

Connectors as Converters



- Enable interaction of independently developed, mismatched components
- Examples of converters
 - Adaptors
 - Wrappers

Connectors as Facilitators



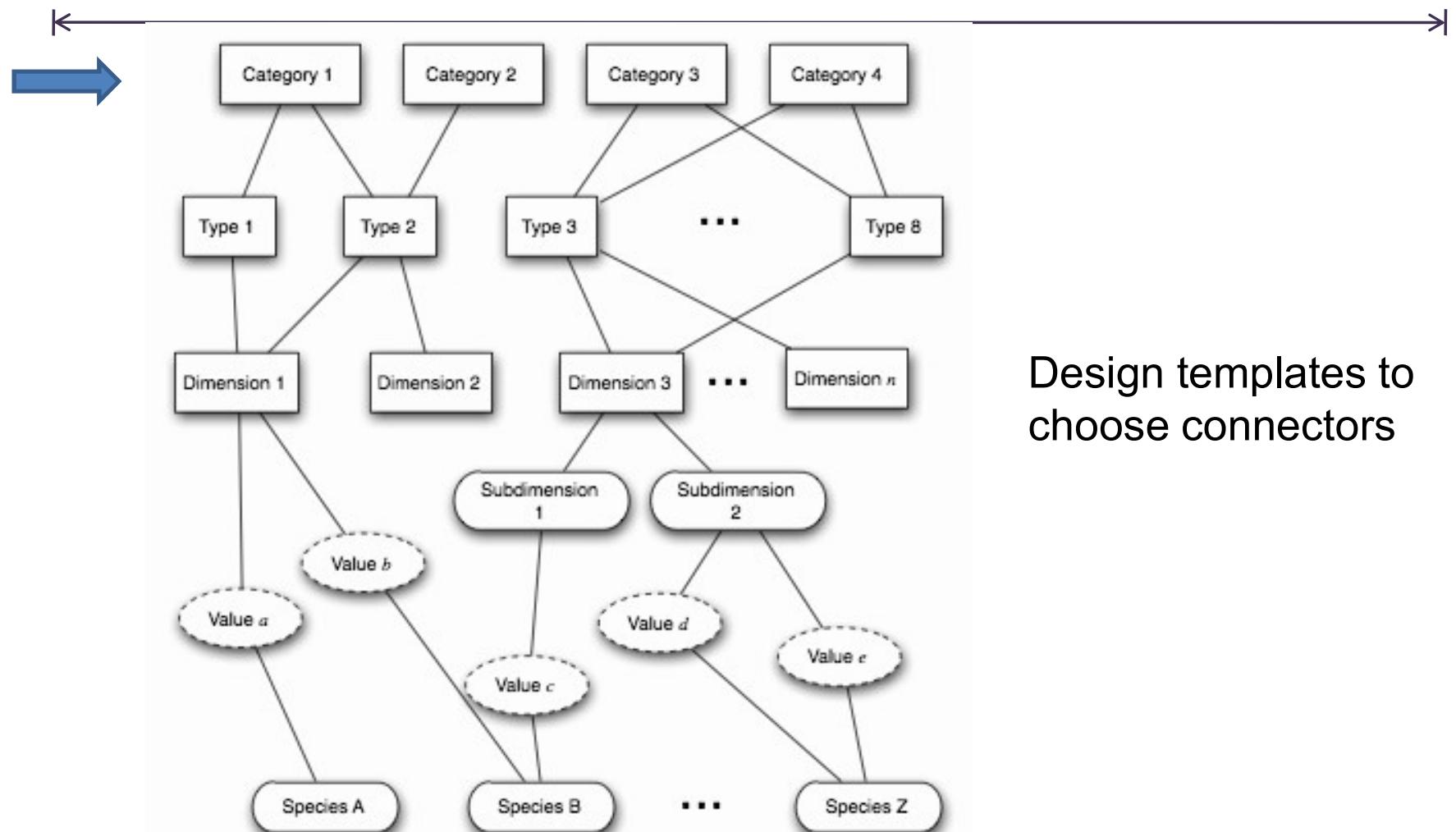
- Enable interaction of components intended to interoperate
 - Mediate and streamline interaction
- Govern access to shared information
- Ensure proper performance profiles
 - e.g., load balancing
- Provide synchronization mechanisms
 - Critical sections
 - Monitors

Connector Types



- Procedure call
- Event
- Data access
- Linkage
- Stream
- Arbitrator
- Adaptor
- Distributor

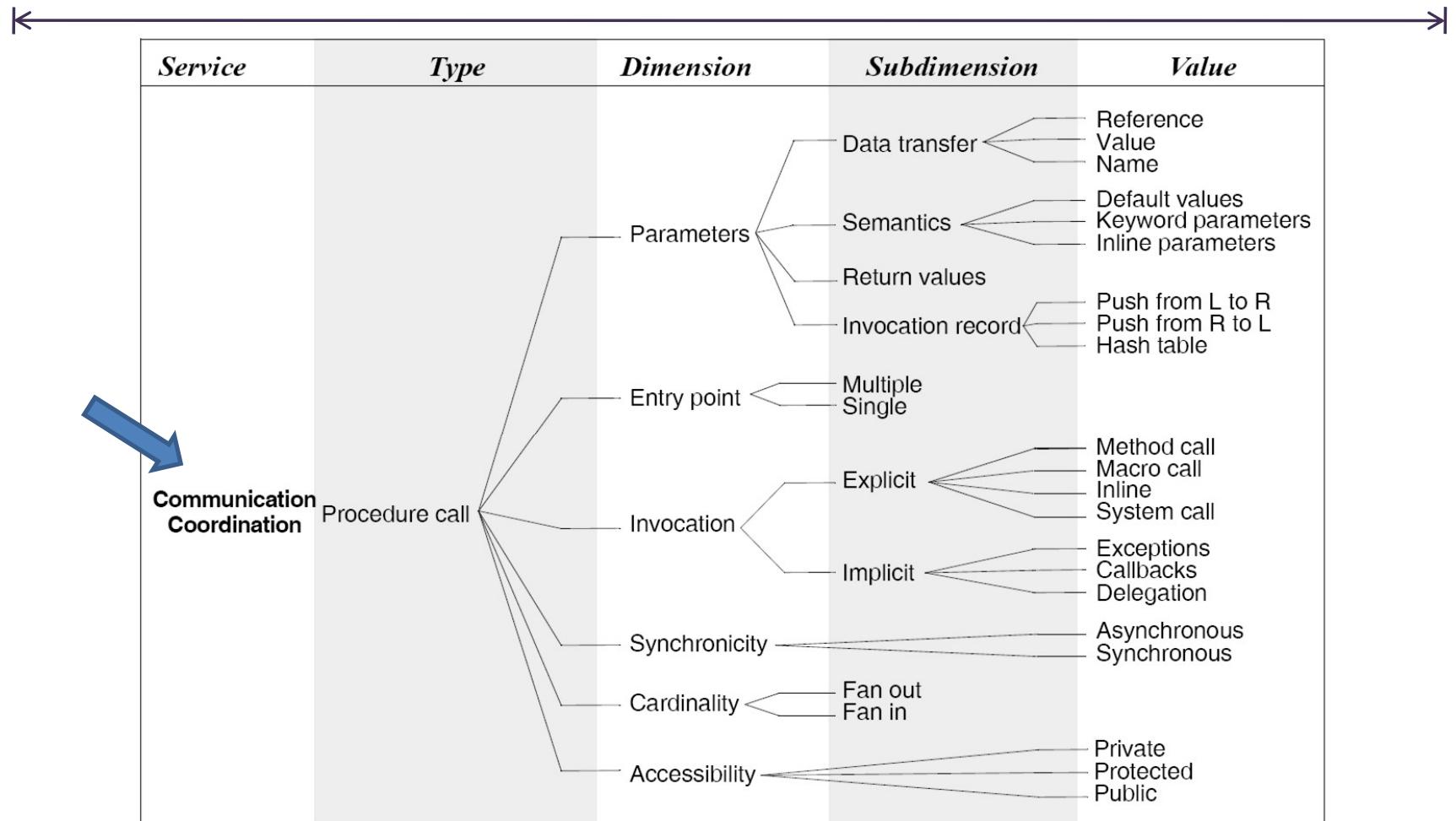
A Framework for Classifying Connectors



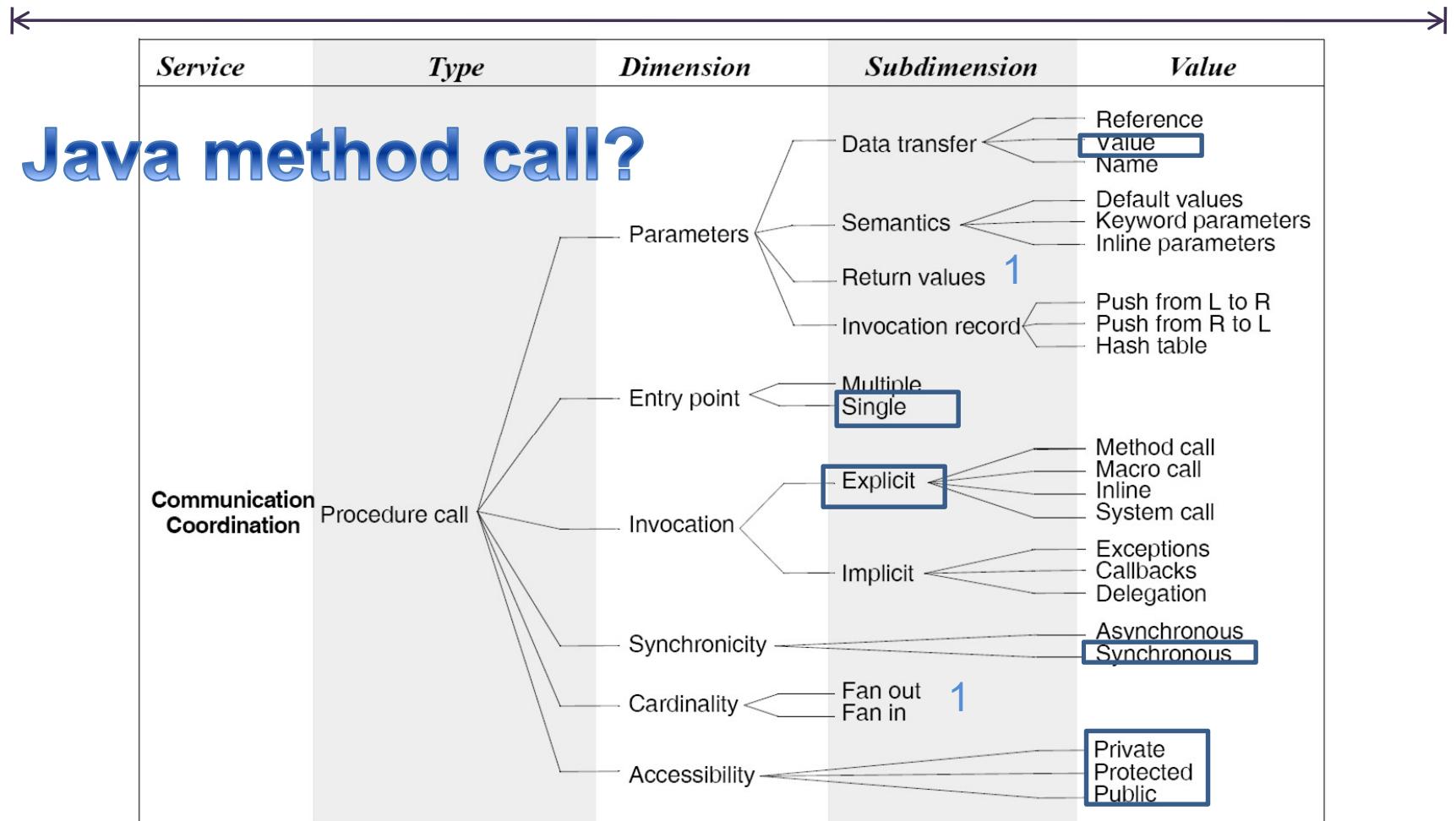
Design templates to choose connectors



Procedure Call Connectors



Procedure Call Connectors



Event Connectors



Service	Type	Dimension	Subdimension	Value
Communication Coordination	Event	Cardinality	Producers Observers Event patterns	Best effort Exactly once At most once At least once

A blue arrow points from the text "Communication Coordination" to the "Service" column of the table.

```
graph TD; Event --> Cardinality[Cardinality]; Event --> Delivery[Delivery]; Event --> Priority[Priority]; Event --> Synchronicity[Synchronicity]; Event --> Notification[Notification]; Event --> Causality[Causality]; Event --> Mode[Mode]; Cardinality --> Producers[Producers]; Cardinality --> Observers[Observers]; Cardinality --> EventPatterns[Event patterns]; Producers --> BE[Best effort]; Observers --> EO[Exactly once]; EventPatterns --> AOMO[At most once]; EventPatterns --> AOL[At least once]; Delivery --> Outgoing[Outgoing]; Delivery --> Incoming[Incoming]; Priority --> Outgoing; Priority --> Incoming; Synchronicity --> Synchronous[Synchronous]; Synchronicity --> ASynchronous[Asynchronous]; Synchronicity --> TSynchronous[Time out synchronous]; Notification --> Polled[Polled]; Notification --> PS[Publish/subscribe]; Notification --> CU[Central update]; Notification --> QD[Queued dispatch]; Causality --> Absolute[Absolute]; Causality --> Relative[Relative]; Mode --> Hardware[Hardware]; Mode --> Software[Software]; Hardware --> PF[Page faults]; Hardware --> I[Interrupts]; Hardware --> T[Traps]; Software --> Signals[Signals]; Software --> GIO[GUI input/output]; Software --> Triggers[Triggers]
```

Event Connectors



C2 event connector?

Service	Type	Dimension	Subdimension	Value
Communication Coordination	Event	Cardinality	Producers Observers Event patterns	1 n
		Delivery	Best effort Exactly once At most once At least once	
		Priority	Outgoing Incoming	
		Synchronicity	Synchronous Asynchronous Time out synchronous	
		Notification	Polled Publish/subscribe Central update Queued dispatch	
		Causality	Absolute Relative	
		Mode	Hardware Software	Page faults Interrupts Traps Signals GUI input/output Triggers

Data Access Connectors



<i>Service</i>	<i>Type</i>	<i>Dimension</i>	<i>Subdimension</i>	<i>Value</i>
Communication Conversion	Data Access	Locality		Thread specific Process specific Global
		Access		Accessor Mutator
		Availability	Transient	Register Cache DMA Heap Stack
			Persistent	Repository access File I/O Dynamic data exchange Database Access
		Accessibility		Private Protected Public
		Lifecycle		Initialization Termination
		Cardinality	Defines Uses	

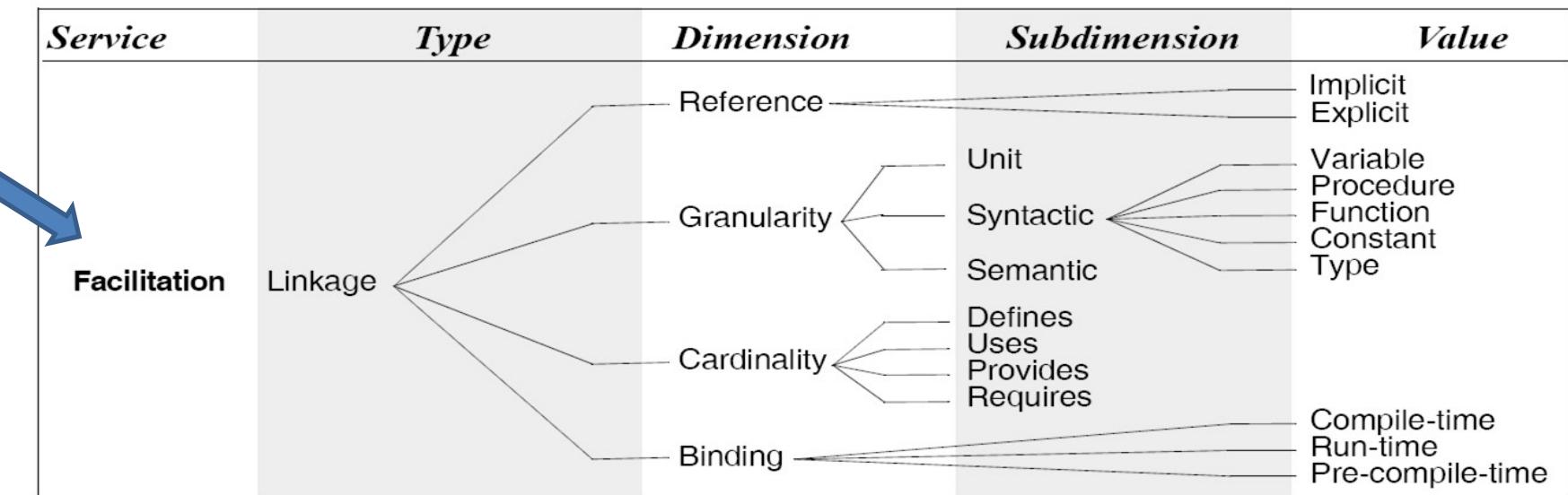
Data Access Connectors



SQL access?

<i>Service</i>	<i>Type</i>	<i>Dimension</i>	<i>Subdimension</i>	<i>Value</i>
Communication Conversion	Data Access	Locality Access Availability Accessibility Lifecycle Cardinality	Transient Persistent	Thread specific Process specific Global Accessor Mutator Register Cache DMA Heap Stack Repository access File I/O Dynamic data exchange Database Access Private Protected Public Initialization Termination Defines Uses 1 <i>n</i>

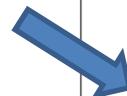
Linkage Connectors



Stream Connectors



Service	Type	Dimension	Subdimension	Value
Communication Stream		Delivery		Best effort Exactly once At most once At least once
		Bounds		Bounded Unbounded
		Buffering		Buffered Unbuffered
		Throughput		Atomic units Higher-order units
		State		Stateless Stateful
		Identity		Named Unnamed
		Locality		Local Remote
		Synchronicity		Synchronous Asynchronous Time out synchronous
		Format		Raw Structured
		Cardinality	Binary	Multi sender Multi receiver
		Cardinality	N-ary	Multi sender/receiver



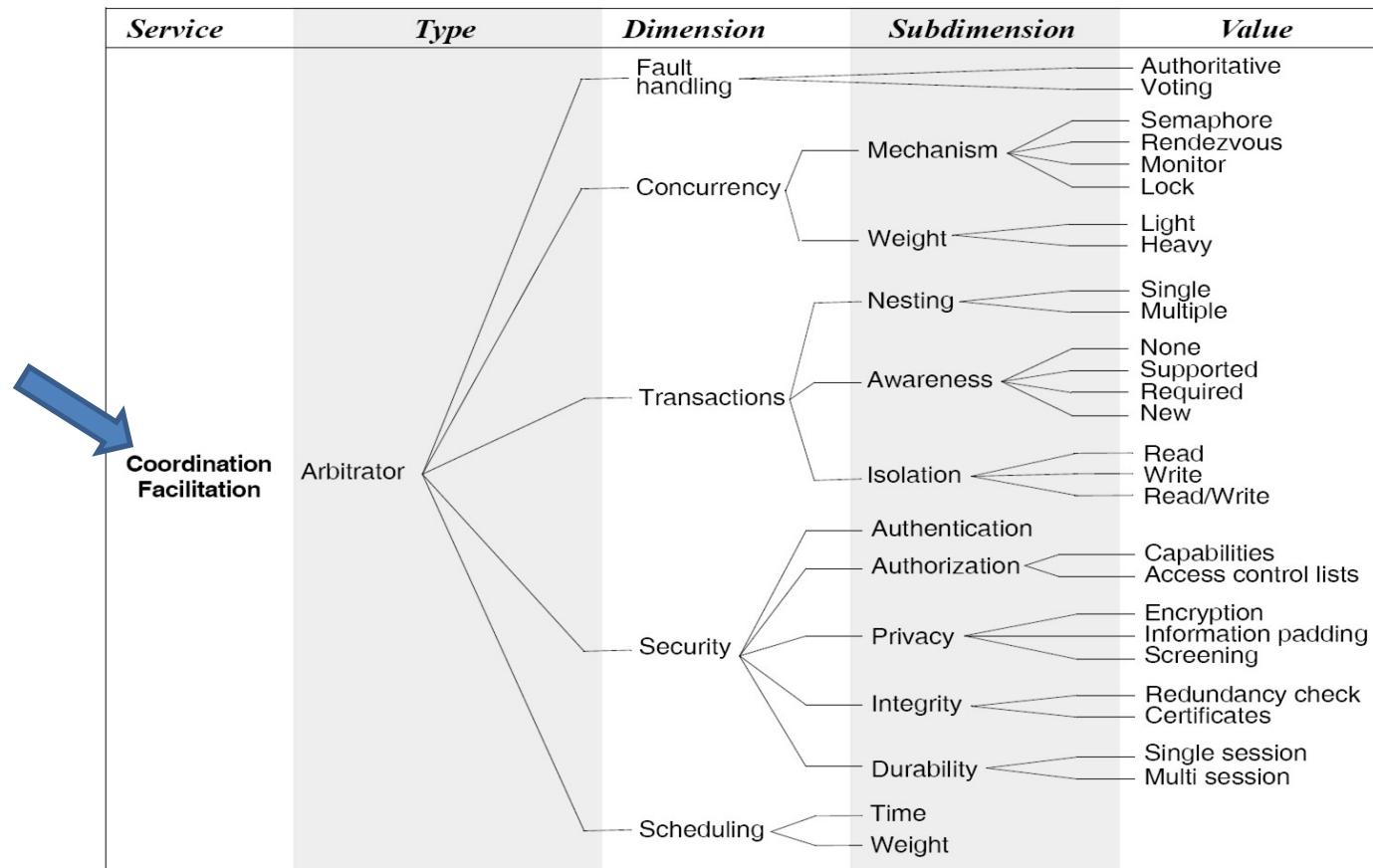
Stream Connectors



Unix pipe?

Service	Type	Dimension	Subdimension	Value
Communication Stream	Delivery		Best effort Exactly once At most once At least once	
	Bounds		Bounded Unbounded	
	Buffering		Buffered Unbuffered	
	Throughput		Atomic units Higher-order units	
	State		Stateless Stateful	
	Identity		Named Unnamed	
	Locality		Local Remote	
	Synchronicity		Synchronous Asynchronous Time out synchronous	
	Format		Raw Structured	
Cardinality	Binary		Multi sender Multi receiver	
	N-ary		Multi sender/receiver	

Arbitrator Connectors

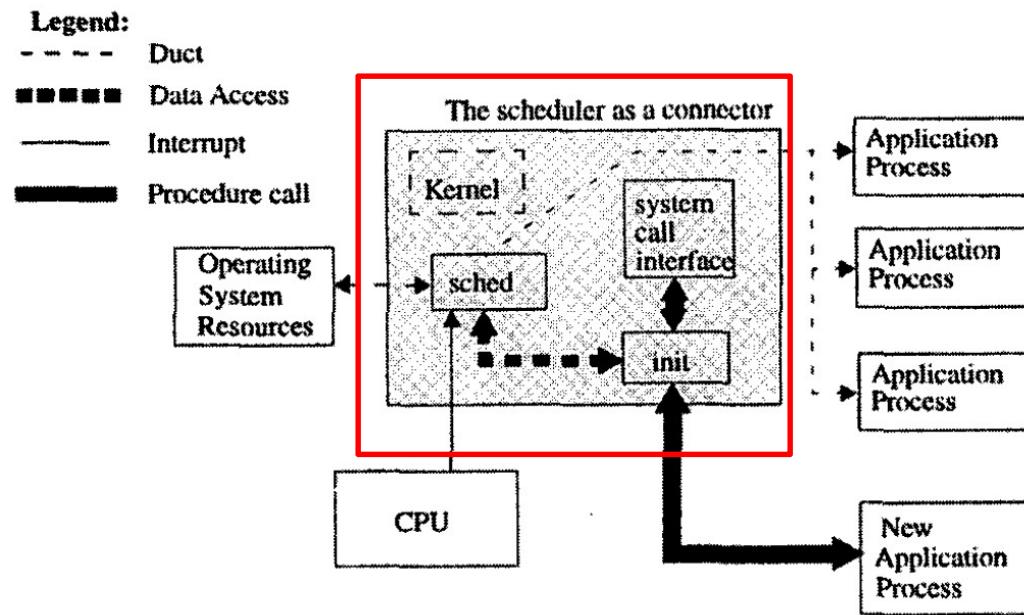


Coordination
Facilitation

Process Scheduler as an Arbitrator Connector



- Switch system resources between multiple user-level processes
- Avoid deadlocks and resource starvation
- Optimize utilization of system resources

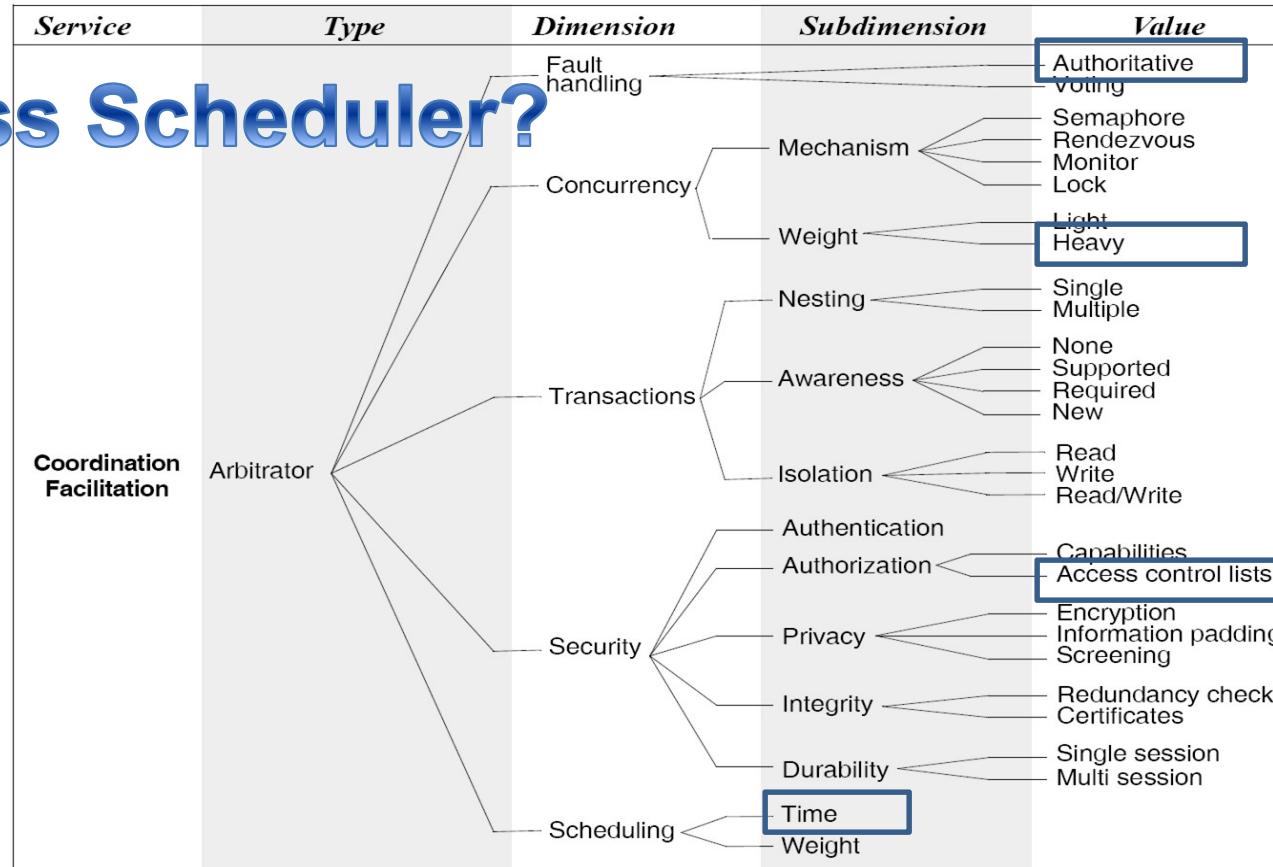


Mehta, N.R.; Medvidovic, N.; Phadke, S., *Towards a taxonomy of software connectors*. ICSE 2000



Arbitrator Connectors

← →
Process Scheduler?



Or...may redesign scheduling with prioritized queue or cooperative multitasking algorithm

Adaptor Connectors



Service	Type	Dimension	Subdimension	Value
Conversion	Adaptor	Invocation conversion Packaging conversion Protocol conversion Presentation conversion	Address mapping Marshalling Translation Wrappers Packagers	



Distributor Connectors



Service	Type	Dimension	Subdimension	Value
Facilitation	Distributor	Naming	Structure based	Hierarchical Flat
		Attribute based		
		Delivery	Semantics	Best effort Exactly once At most once At least once
			Mechanism	Unicast Multicast Broadcast
		Routing	Membership	Bounded Ad-hoc
			Path	Static Cached Dynamic



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

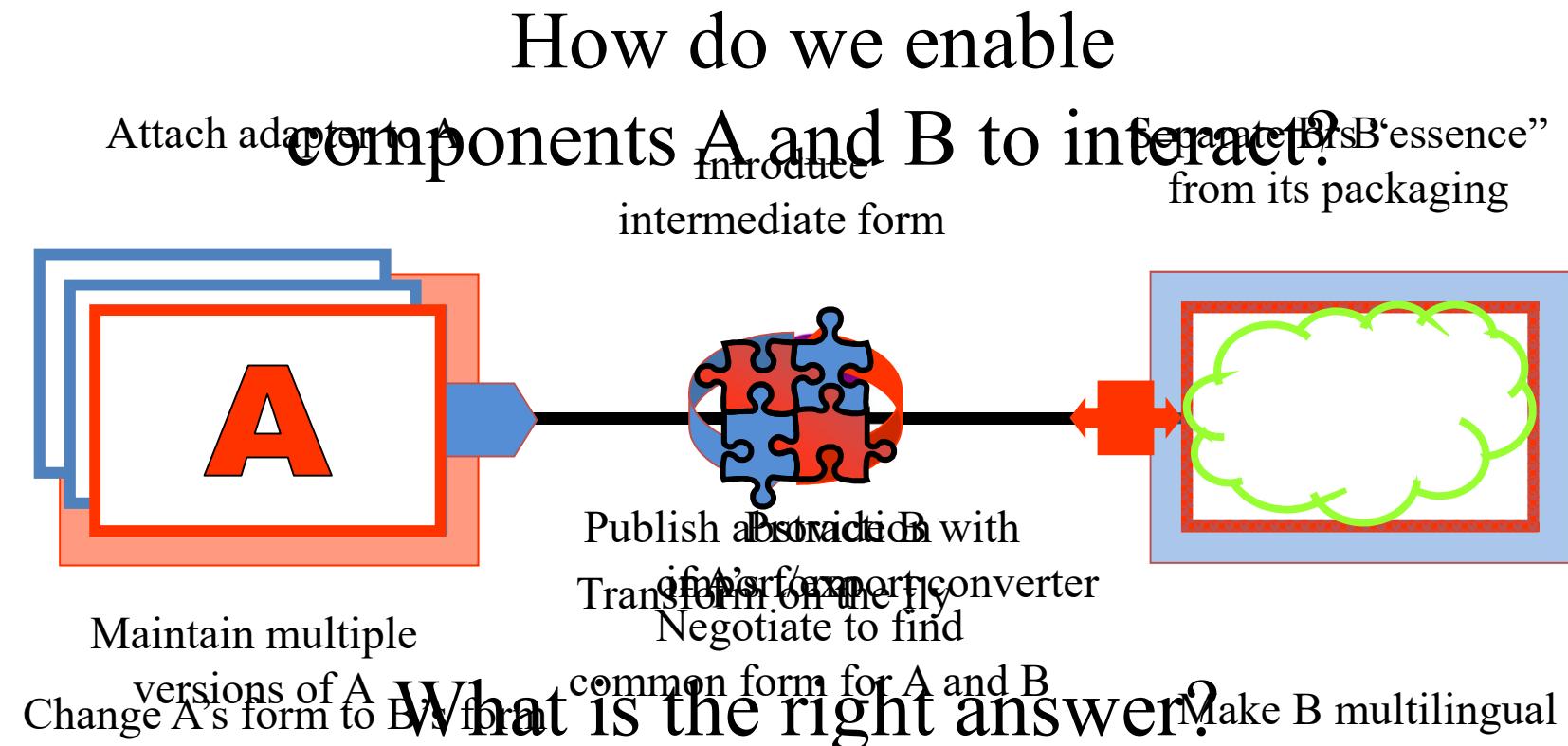


COMPOSITE CONNECTORS

CSS 553, Spring 2023, 8c Composite
Connectors



Role and Challenge of Software Connectors



How Does One Select a Connector?



- Determine a system's interconnection and interaction needs
 - Software interconnection models can help
- Determine roles to be fulfilled by the system's connectors
 - Communication, coordination, conversion, facilitation
- For each connector
 - Determine its appropriate type(s)
 - Determine its dimensions of interest
 - Select appropriate values for each dimension
- For multi-type, i.e., composite connectors
 - Determine the atomic connector compatibilities

Simple Example



- System components will execute in two processes on the same host
 - Mostly intra-process
 - Occasionally inter-process
- The interaction among the components is synchronous
- The components are primarily computation-intensive
 - There are some data storage needs, but those are secondary

Simple Example (cont'd)



- Select procedure call connectors for intra-process interaction
- Combine procedure call connectors with distributor connectors for inter-process interaction
 - RPC
- Select the values for the different connector dimensions
 - What are the appropriate values?
 - What values are imposed by your favorite programming language(s)?

Procedure Call Connectors Revisited



<i>Service</i>	<i>Type</i>	<i>Dimension</i>	<i>Subdimension</i>	<i>Value</i>
Communication Coordination	Procedure call	<ul style="list-style-type: none"> Parameters Entry point Invocation Synchronicity Cardinality Accessibility 	<ul style="list-style-type: none"> Data transfer Semantics Return values Invocation record Multiple Single Explicit Implicit Fan out Fan in Reference Value Name Default values Keyword parameters Inline parameters Push from L to R Push from R to L Hash table Method call Macro call Inline System call Exceptions Callbacks Delegation Asynchronous Synchronous Private Protected Public 	

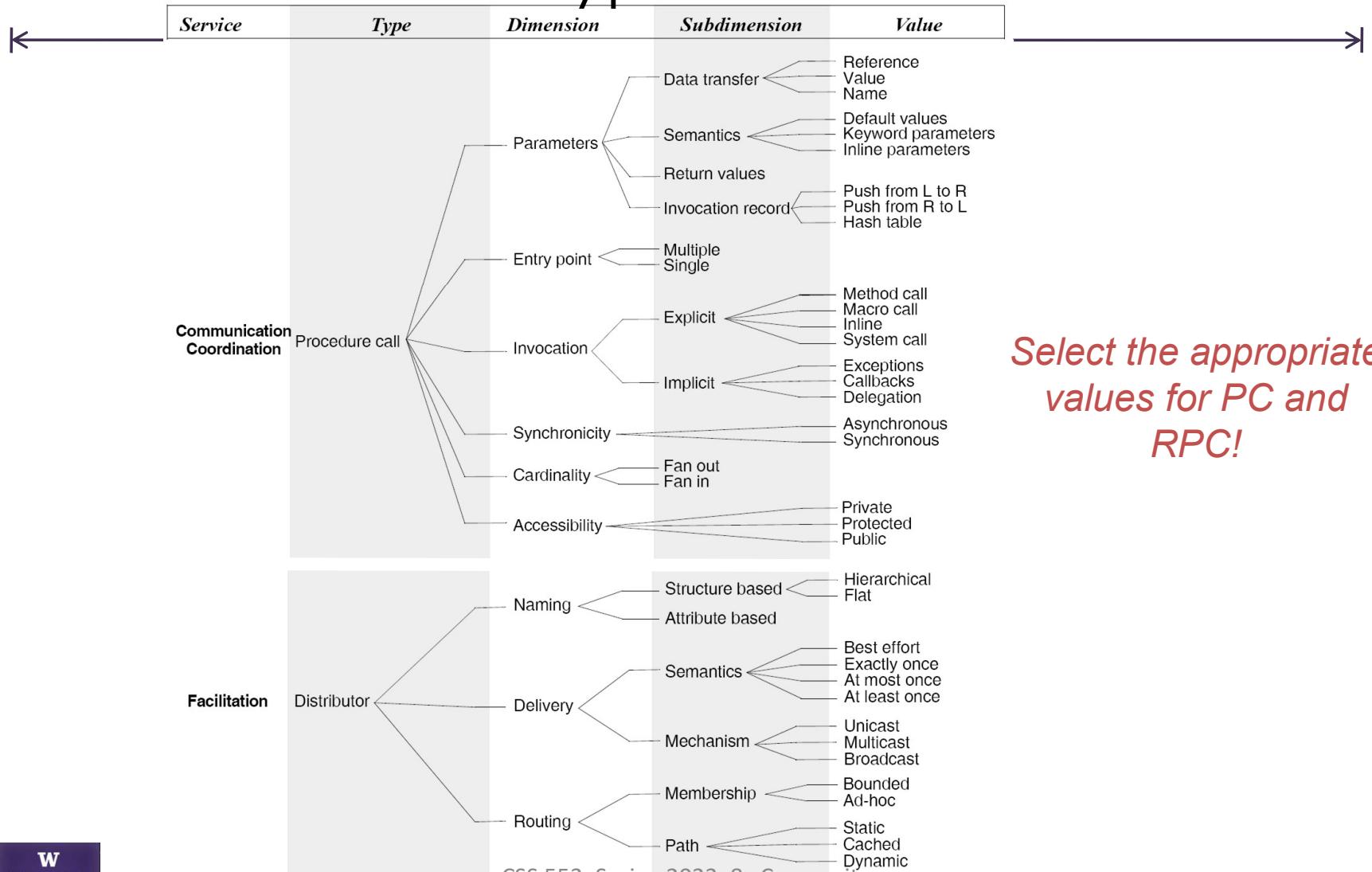
Distributor Connectors Revisited



<i>Service</i>	<i>Type</i>	<i>Dimension</i>	<i>Subdimension</i>	<i>Value</i>
Facilitation	Distributor	Naming Delivery Routing	Structure based Attribute based Semantics Mechanism Membership Path	Hierarchical Flat Best effort Exactly once At most once At least once Unicast Multicast Broadcast Bounded Ad-hoc Static Cached Dynamic



Two Connector Types in Tandem



Select the appropriate values for PC and RPC!



Composing Basic Connectors



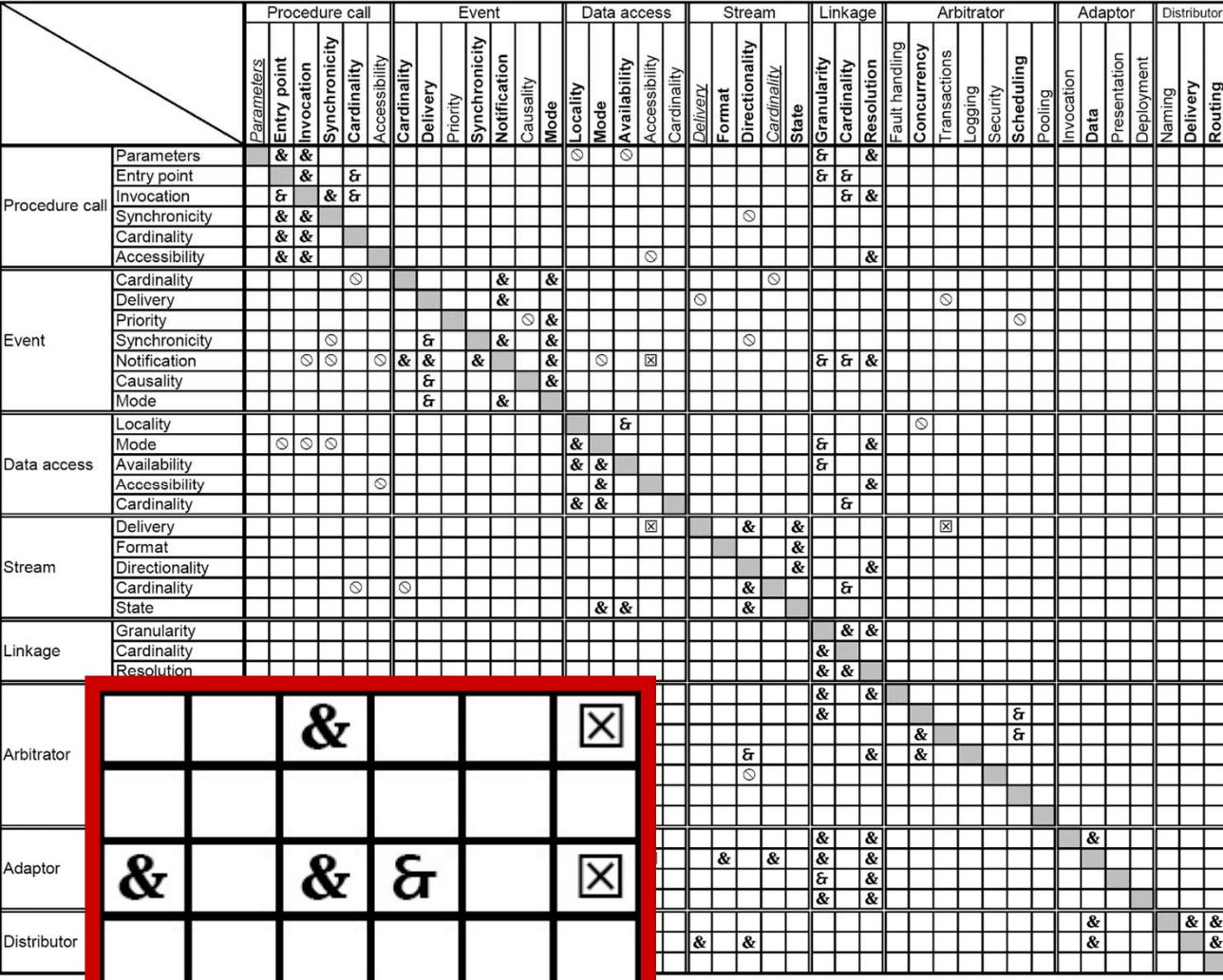
- Why? Real-world situations often require sophisticated connectors
 - E.g. RPC with privacy and integrity
 - Hard to systematically compose hybrid connectors
- All connectors cannot be composed
 - Some are naturally interoperable
 - Some are incompatible
 - All are likely to require trade-offs
- The composition can be considered at the level of connector type dimensions and subdimensions

Connector Dimension Inter-Relationships



- Requires – 
 - Choice of one dimension mandates the choice of another
- Prohibits – 
 - Two dimensions can never be composed into a single connector
- Restricts – 
 - Dimensions are not always required to be used together
 - Certain dimension combinations may be invalid
- Cautions – 
 - Combinations may result in unstable or unreliable connectors

Dimension Inter-Relationships in a Nutshell



The diagram is a grid representing the relationships between different system dimensions. The columns and rows are labeled with dimension names, and the intersections show how one dimension relates to another. A red box highlights a specific cluster of cells in the bottom-left corner.

	Procedure call	Event	Data access	Stream	Linkage	Arbitrator	Adaptor	Distributor
Procedure call	Parameters Entry point Invocation Synchronicity Cardinality Accessibility	Procedure call Delivery Priority Synchronicity Notification Causality Mode	Data access Locality Mode Availability Accessibility Cardinality	Stream Format Directionality Cardinality State	Linkage Granularity Cardinality Resolution	Arbitrator Concurrency Transactions Logging Security Scheduling Pooling	Adaptor Invocation Data Presentation Deployment Naming Delivery Routing	Distributor
Event	Cardinality Delivery Priority Synchronicity Notification Causality Mode	Delivery Priority Synchronicity Notification Causality Mode	Locality Mode Availability Accessibility Cardinality	Format Directionality Cardinality State	Granularity Cardinality Resolution	Concurrency Transactions Logging Security Scheduling Pooling	Invocation Data Presentation Deployment Naming Delivery Routing	
Data access	Locality Mode Availability Accessibility Cardinality	Delivery Priority Synchronicity Notification Causality Mode	Delivery Format Directionality Cardinality State	Format Directionality Cardinality State	Granularity Cardinality Resolution	Concurrency Transactions Logging Security Scheduling Pooling	Invocation Data Presentation Deployment Naming Delivery Routing	
Stream	Delivery Format Directionality Cardinality State	Delivery Format Directionality Cardinality State	Delivery Format Directionality Cardinality State	Format Directionality Cardinality State	Delivery Format Directionality Cardinality State	Delivery Format Directionality Cardinality State	Invocation Data Presentation Deployment Naming Delivery Routing	
Linkage	Granularity Cardinality Resolution	Delivery Format Directionality Cardinality State	Delivery Format Directionality Cardinality State	Delivery Format Directionality Cardinality State	Delivery Format Directionality Cardinality State	Delivery Format Directionality Cardinality State	Invocation Data Presentation Deployment Naming Delivery Routing	
Arbitrator	&							
Adaptor								
Distributor								

CSS 553, Spring 2023, 8c Composite

Connectors



Well Known Composite Connectors



- Grid connectors (e.g., Globus)
 - Procedure call
 - Data access
 - Stream
 - Distributor
- Peer-to-peer connectors (e.g., BitTorrent)
 - Arbitrator
 - Data access
 - Stream
 - Distributor
- Client-server connectors
- Event-based connectors

Selecting an OTS Connector



- Identify your requirements for a connector
- Find an OTS that closely matches your requirements



Using OTS in C2 Style



- Requirements
 - Platform and language support
 - Inter- and intra-process communication support
 - Similarity to software connectors
 - Ease of integration and use
 - Multiple instances in an application
 - Support for dynamic change
 - Performance



Evaluating OTS Connectors for C2



- Q
- Polylith
- Java RMI

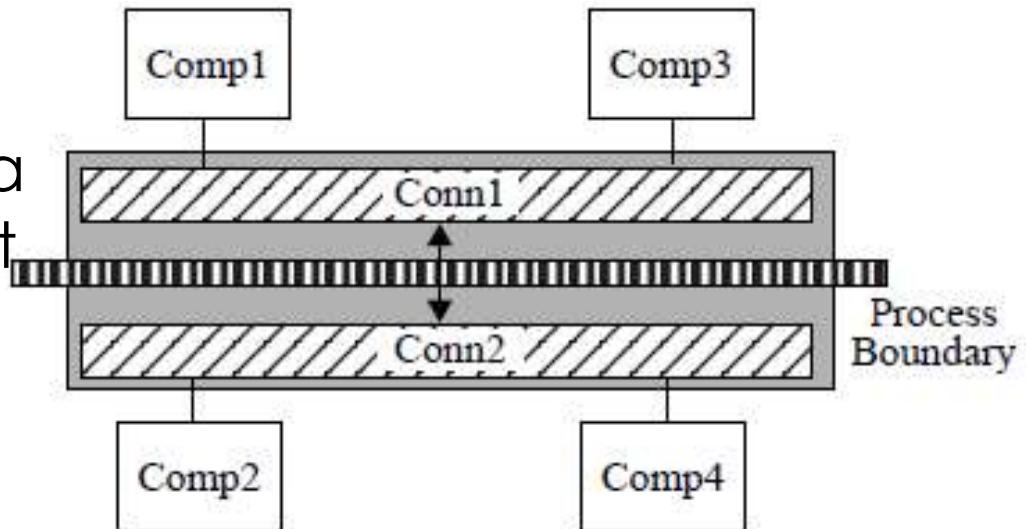
For more info, see Medvidovic, N., Dashofy, E.M., Taylor, R.N., Employing Off-the-Shelf Connector Technologies in C2-Style Architectures, In Proc of the California Software Symposium (CSS'98), pages 21-30, Irvine, CA, October 23, 1998.



Q System



- Provides asynchronous message interface on top of standard RPC
- Encapsulate Q inside a C2 connector (Q is not a software bus) → “Q-C2 connector”
- Split the connector across a process boundary



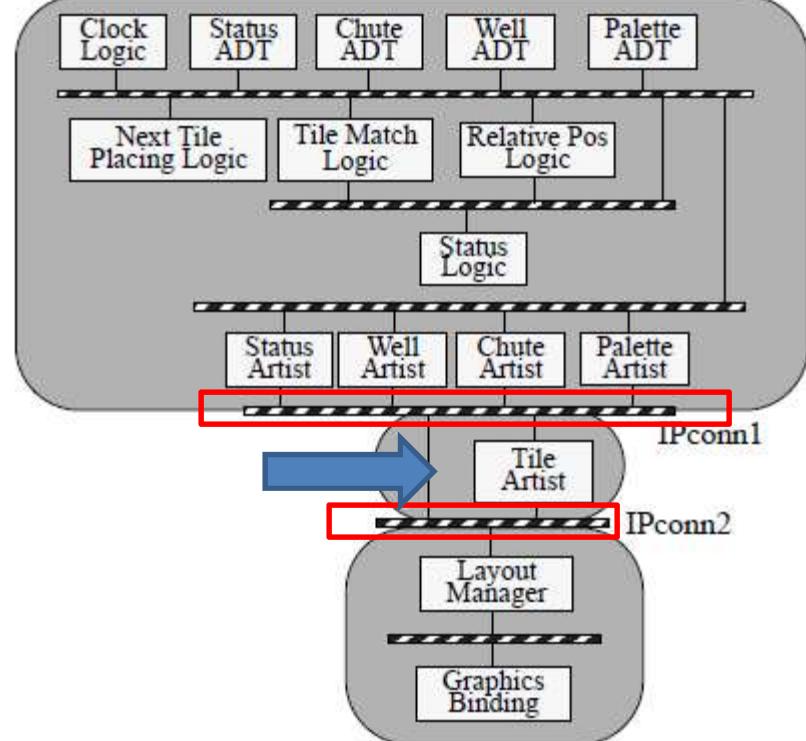
Taken from Medvidovic, N., Dashofy, E.M., Taylor, R.N., Employing Off-the-Shelf Connector Technologies in C2-Style Architectures, CSS'98, Irvine, CA, October 23, 1998.

CSS 553, Spring 2023, 8c Composite
Connectors



Q System

- Split Klax across three processors
- Able to exchange Tile Artist in C++ with Tile Artist in Ada during runtime
 - although some messages were lost
- Performed well - adjusted the Clock to use short time intervals



Taken from Medvidovic, N., Dashofy, E.M., Taylor, R.N., Employing Off-the-Shelf Connector Technologies in C2-Style Architectures, CSS'98, Irvine, CA, October 23, 1998.

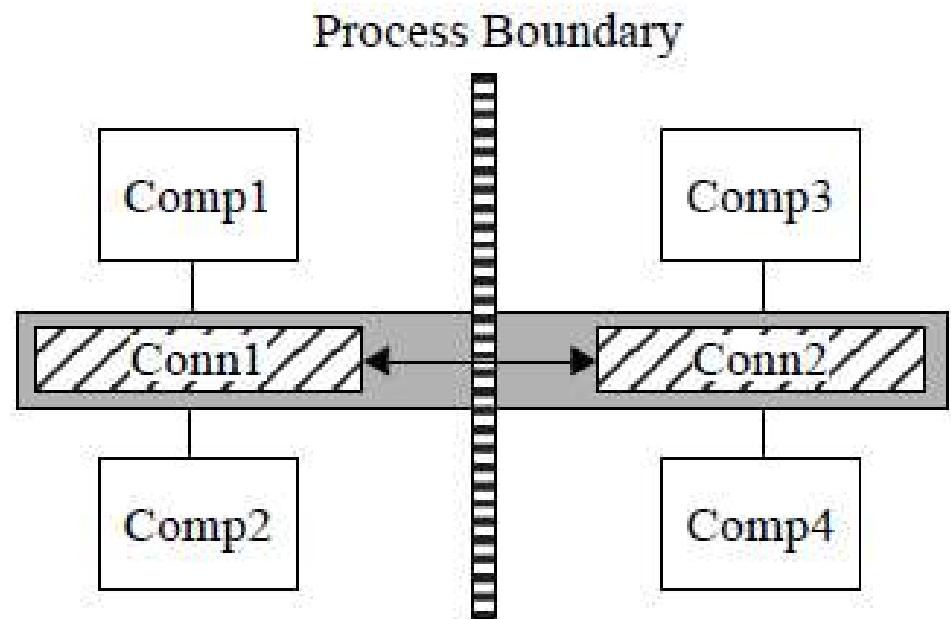
CSS 553, Spring 2023, 8c Composite
Connectors



Polylith



- Software bus, implemented in C
- Built a Polylith-C2 connector and split a connector horizontally
- Version of Polylith used UNIX process scheduling → messages handled in bursts rather than continuously → poor performance



Taken from Medvidovic, N., Dashofy, E.M., Taylor, R.N., Employing Off-the-Shelf Connector Technologies in C2-Style Architectures, CSS'98, Irvine, CA, October

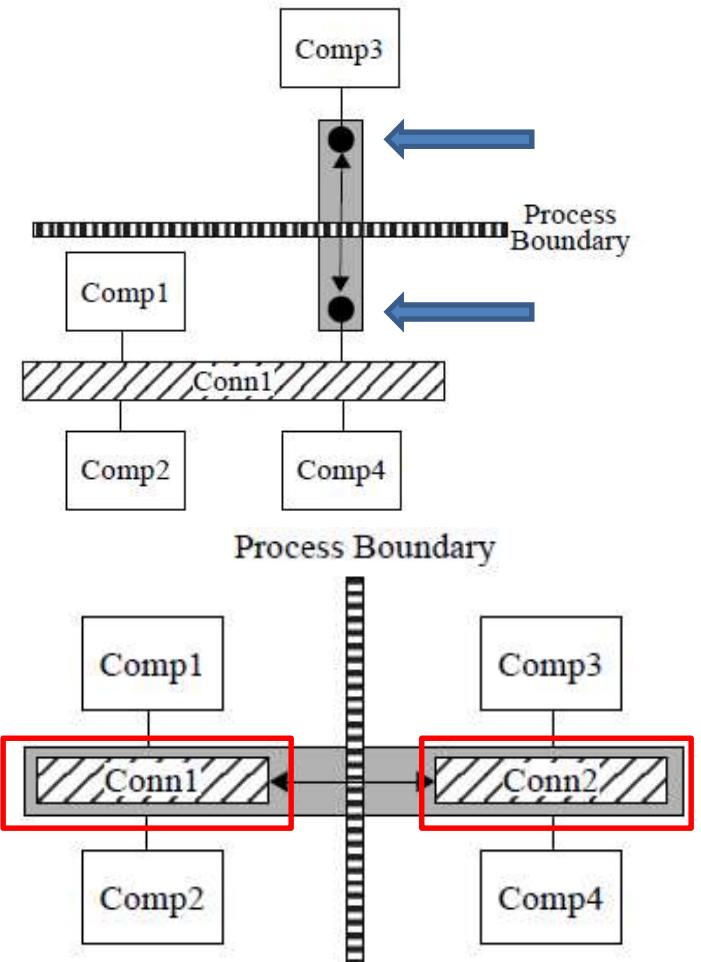
CSS 553, Spring 2023, 8c Composite
Connectors

23, 1998.


Java RMI

◀

- Integrated using the virtual port approach and lateral weld approach
- Pass messages across the virtual ports (can be marshalled, instead of passing object references)
- RMI-C2 connector has all the capabilities of a single process connector



Taken from Medvidovic, N., Dashofy, E.M., Taylor, R.N., Employing Off-the-Shelf Connector Technologies in C2-Style Architectures, CSS'98, Irvine, CA, October 23, 1998.



Java RMI



- Well-suited with C2
- Minimal modification required to convert a single process into an multi-process C2 application
- Does not preclude other technologies – can use Q as well



Other Possible Technologies



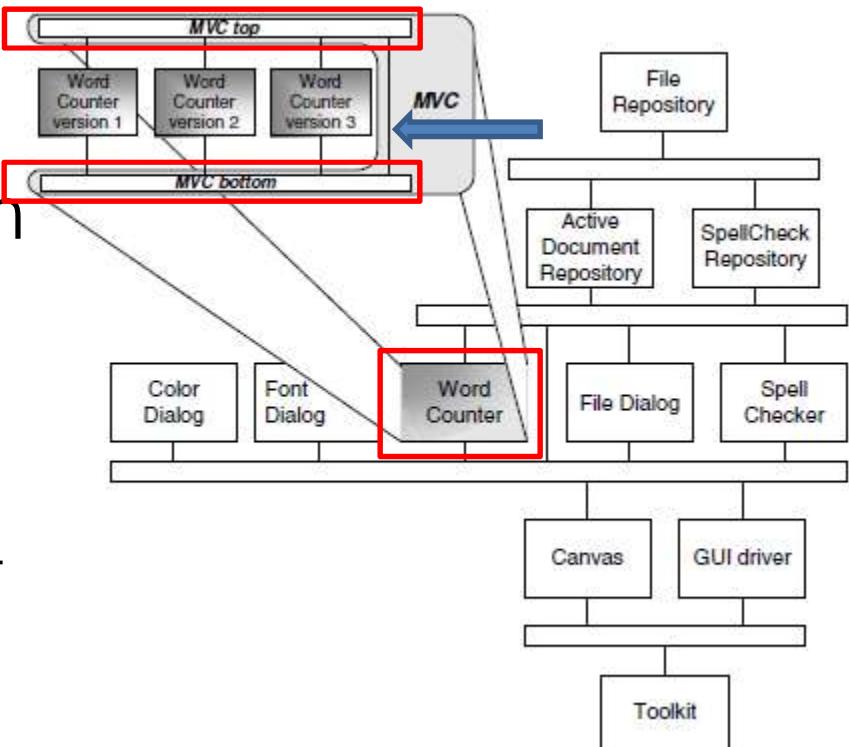
- JMS
 - Java Messaging Service
 - <http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html>
- Microsoft AMQP
 - Advanced Message Queueing Protocol
 - <https://www.amqp.org/>
- MQTT
 - Message Queueing Telemetry Transport
 - One of the main components of Internet of Things (IoT) ecosystem
 - <http://mqtt.org/>



Using Connectors to Support Dynamism in C2



- May be infeasible to “test” a component on a heterogeneous system
- Approach:
 - Use a multi-version connector (“MvC”) to simulate one component (Top & Bottom MvCs)
 - Add new component versions while keeping the old version



Rakic, M., Medvidovic, N., Increasing the confidence in off-the-shelf components: a software connector-based approach. Proc of Symposium of Software Reusability 2001



Using Connectors to Support Dynamism



- Approach (cont'd)
 - Messages are sent to different versions
 - Track new component's behavior using MvC logging of component interactions
 - MvC “swallows” all output of the other versions, and only allows the output of the authoritative version to the rest of the system
 - MvC logging also allows to revert to a previous state
 - Useful when adding/removing components



Questions



- Identify one connector from your open source project. What role does it play and what type?
- How can connectors support dynamism? Provide specific example.
- Pick one connector from lecture and find an example code that represents this connector. This source code may be obtained from any open source project (i.e., does not have to be your assigned project).





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Agenda



- Review
- Answer questions
- G4 posted (submit in 2 places)
 - Presentation – discussion board for Q&A
 - Rest of deliverables
- G3 Q&A
- -----
- Design Presentation
- Activity / Groupwork

Review



- Connectors
 - Roles?
- Basic connector types
- Composite connector types
- MvC \neq MVC

Questions

Wk 7

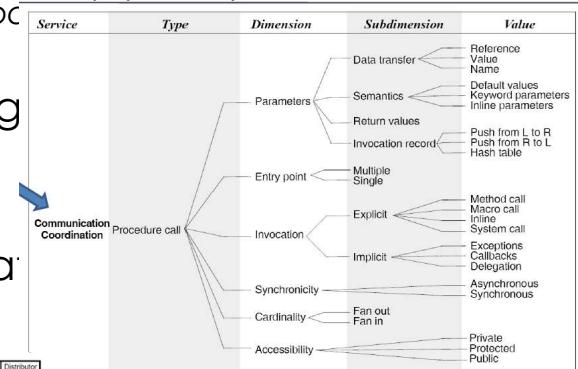
- Section 4.3.6 in the book mentions “freedom of constraints”. What does this mean? **Supreme, Team 4**
 - Limited by specific rules (restriction, arch style or some technology choices), freedom from increased complexity (more predictable), from uncertainties

Wk 8

- How can connectors support dynamism? Provide a specific example.
Dawn
 - MvC connectors – can enable dynamism, add new component versions, messages are sent to different versions, use MvC logging (compare performance & behavior), MvC swallows output of different components, MvC logging allows to rollback to previous state – C2 arch style
- Do connectors have a separate implementation from components?
The Boffins
 - Yes, separate implementation, connectors for communication. E.g., adapter
- What is an advantage to treating connectors independently? **To be decided**
 - Reusability, heterogeneity, assist with arch style gyle, improves quality of software

Questions

- ← Pick 2 software connector roles and explain them **Ludus, Team 9**
 - Conversion – converting between mismatched components (e.g., MPI, database adapters – ODBC, data mapping, conversion)
 - Communication – data transfer or message, exchange of info (e.g., Proc call, REST API – everything encapsulated in the REST message)
 - Coordination – required between 2 components – if one component controls another component (publish-subscribe, filtering)
- How would you use this connector variation chart, given **The Magratheans**
- When would you use the following compatibility matrix? **Open Source Web Services**



		Procedure call	Event	Data access	Stream	Linkage	Arbitrator	Adaptor	Distributor										
		Parameters	Entry point	Synchronicity	Cardinality	Locality	Availability	Concurrency	Format	Directionality	State	Granularity	Cardinality	Failure handling	Transactions	Deployment	Naming	Delivery	Routing
Procedure call	Parameters	& &																	
	Entry point		& &																
Event	Invocation	&	&																
	Synchronicity			& &															
Data access	Cardinality																		
	Delivery																		
Stream	Priority																		
	Synchronicity																		
Linkage	Notification																		
	Causality																		
Arbitrator	Mode																		
	Locality																		
Adaptor	Mode																		
	Accessibility																		
Distributor	Cardinality																		
	Delivery																		

Questions



Wk 8

- Pick one connector from lecture and find an example code that represents this connector. This source code may be obtained from any open source project (i.e., does not have to be your assigned project).

→ Good answers posted in Reflection – some have code examples → encourage you to review them and ask questions (reply to posting)

Question from Reflection



- “we wonder if the wrapper class should be considered an connector as a whole or a component that is revoking wrapping interfaces.”
- Adaptor connector – performs conversion, so wrapper class is a connector

G3 Clarifications



- What is provided interface & required interface?
- Provided interface – where the interface is defined

```
public int getScore() {  
    //definition of interface is given  
}
```
- Required interface – where the interface is called
Server s;
s.getScore(); ← required interface

Explicit invocation

Implicit invocation – simply mention events, messages,

DESIGN PRESENTATIONS

ACTIVITIES

Design Exercise from previous week



From last week...

- Improve your software design with the following new requirements. It must be robust (i.e. handle hardware malfunctions). It must also require minimal downtime for software fixes / patches. It must be functional for at least 5 years.
- ...using whatever method you like...
- ...using whatever architectural style / pattern ...
- ...Cost is a concern, and this will be your final design that you will hand off to your development team...
- ...do this exercise with your group...

Reflection Questions



- How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?
- Who did you keep in mind when making your design?
- What was your goal with your design?
- Did you have more than one goal?
- Did you reach the goal(s) with your design?

READ THIS FIRST - 20 minutes

Topic: Architecture Styles

This activity is designed to

- Give you practice in designing
- Tie up loose ends regarding upcoming assignment(s)

Roles:

• **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.

• **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.

• **Note Taker** - takes record of the group’s discussion in the Google doc (below).

• **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity or the assignments due tonight, type in Red text

All teams will do the following task:

Previous design exercise

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

- ...using whatever method you like...
- ...using whatever architectural style / pattern we covered thus far in class...
- Cost is a concern...and this is the final design that you will hand off to your development team

Team Name: Dawn

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

A four-tier architecture is a well-established software design pattern that can be employed in this scenario. It involves the segregation of an application into three logical layers: the Presentation Layer, the Business Logic Layer, and the Data Access Layer. Here's a breakdown of how this might work for the traffic control system.

1. Presentation Layer

This would be the user interface of the system, used by traffic managers or other administrators to monitor and manage the traffic control system. It would display real-time and historical traffic data, system status, and potentially allow manual override of traffic signals when necessary. This layer would be a web-based interface, ensuring accessibility from multiple devices and locations.

2. Hardware Layer

This layer will connect the data from hardware, such as traffic lights, cameras. This layer is designed for data collection.

List Roles and Student names

Moderator : Chloe

TimeKeeper: Abdul

NoteTaker: Luo

Reporter: Barack

3. Business Logic Layer

This is the core of the application, containing all the logic for processing input data and controlling the traffic signals. It can be divided into several components, each handling a specific aspect of the system:

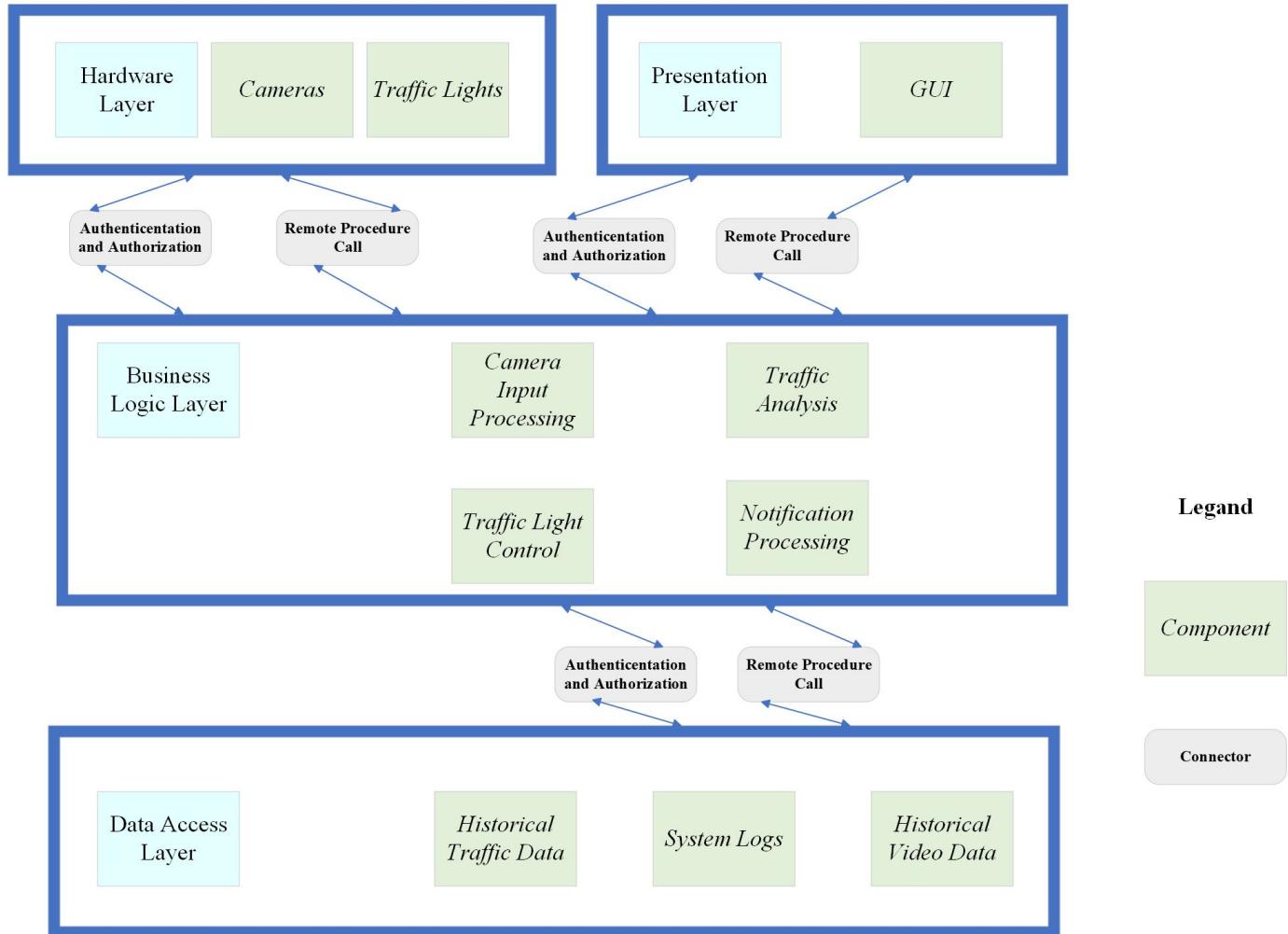
Camera Input Processing Component: This component would take real-time feeds from street cameras, preprocess the data (e.g., image enhancement, object detection), and forward it to the Traffic Analysis Component.

Traffic Analysis Component: This component would analyze the preprocessed data from the Camera Input Processing Component to identify the current state of traffic at different intersections. Techniques such as machine learning and computer vision can be used to understand traffic density and flow patterns.

Traffic Light Control Component: This component would receive the traffic state information from the Traffic Analysis Component and use it to control the traffic lights. The decision logic can be based on algorithms designed to optimize traffic flow and reduce congestion.

4. Data Access Layer

This layer would handle all interactions with the system's data storage, ensuring data consistency and integrity. It would store all necessary data such as historical traffic data, system logs, and so on. A mix of SQL and NoSQL databases could be used, depending on the specific requirements of each type of data.



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: To be decided

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Previous design exercise

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches Traffic light - Camera - Traffic lights control system - load balancing server
- Functional for at least 5 years

Create a design for this software...

- ...using whatever method you like...
- ...using whatever architectural style / pattern we covered thus far in class...
- ...Cost is a concern...and this is the final design that you will hand off to your development team

List Roles and Student names

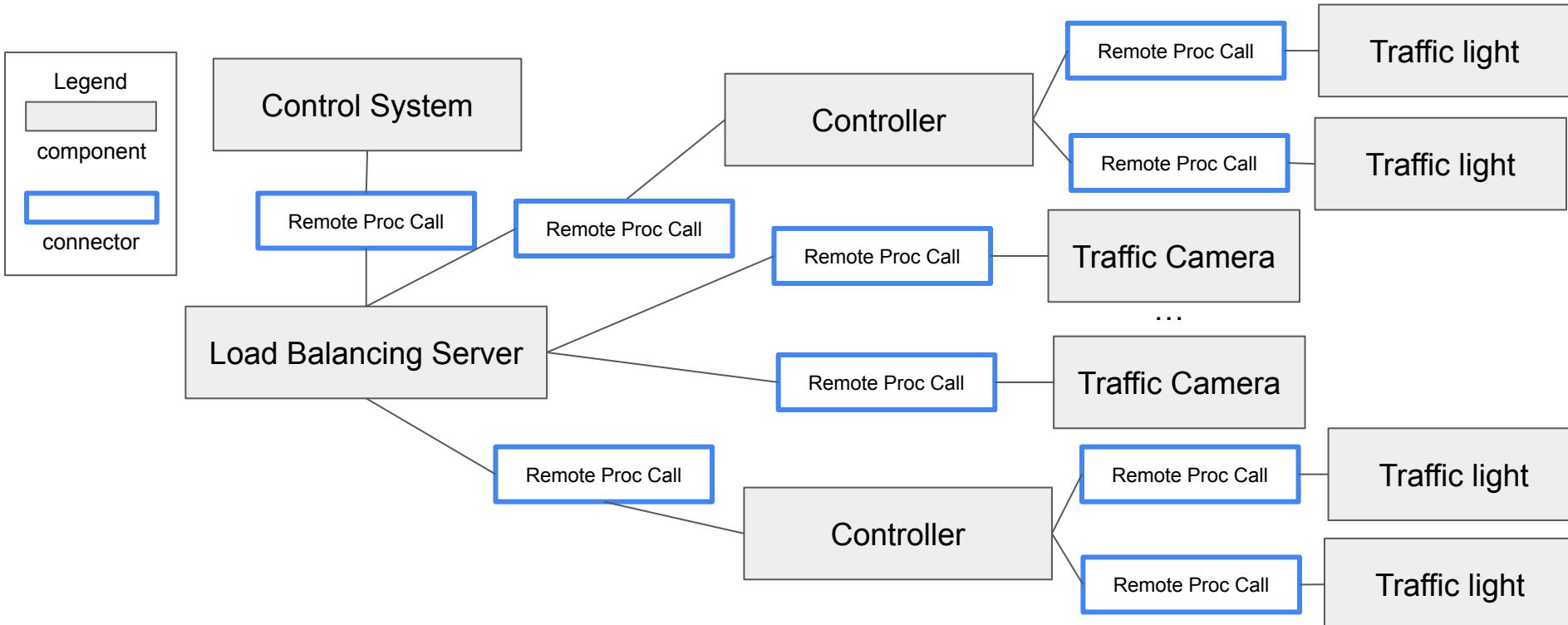
REST style; its standardized and general interfaces allows for extendibility (something that would remain in tact for over 5 years).

We want to implement load balancing for the sake of facilitating the higher-order more complex interaction in terms of the flow of control

- prevents the control system from being overwhelmed by a specific stream of data

Remote Procedure Calls coordinate, communicate, and facilitate the data

- each set of traffic lights have a centralized controller



Team Name: Team 4

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

...using whatever method you like...

...using whatever architectural style / pattern we covered thus far in class...

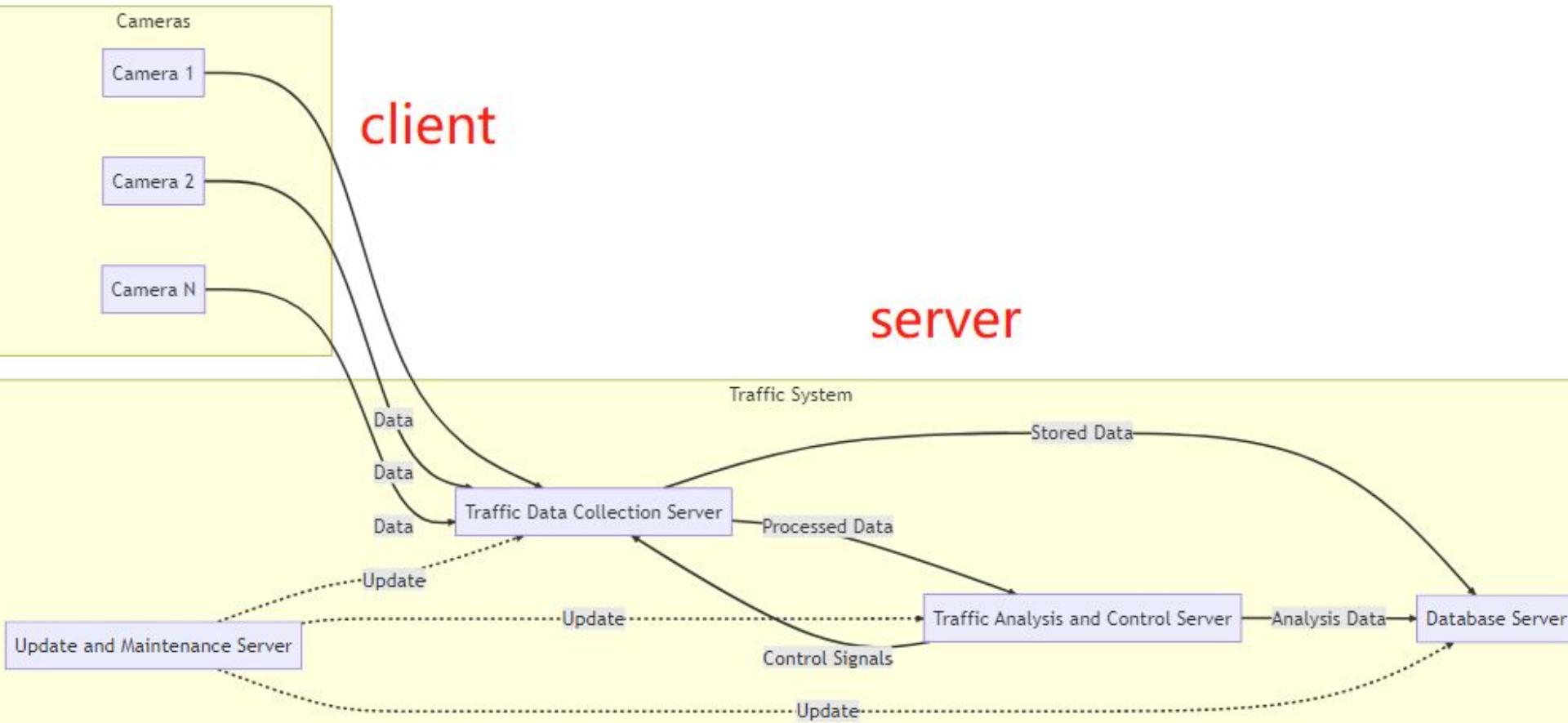
...Cost is a concern...and this is the final design that you will hand off to your development team

Functional for 5 years

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Sense-Compute-Control for real-time part

Robust



Team Name: Ludus

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Not C2

Module based

Event - Sensor

Overall Control

P2P

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Magratheans

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

- ...using whatever method you like...
- ...using whatever architectural style / pattern we covered thus far in class...
- ...Cost is a concern...and this is the final design that you will hand off to your development team

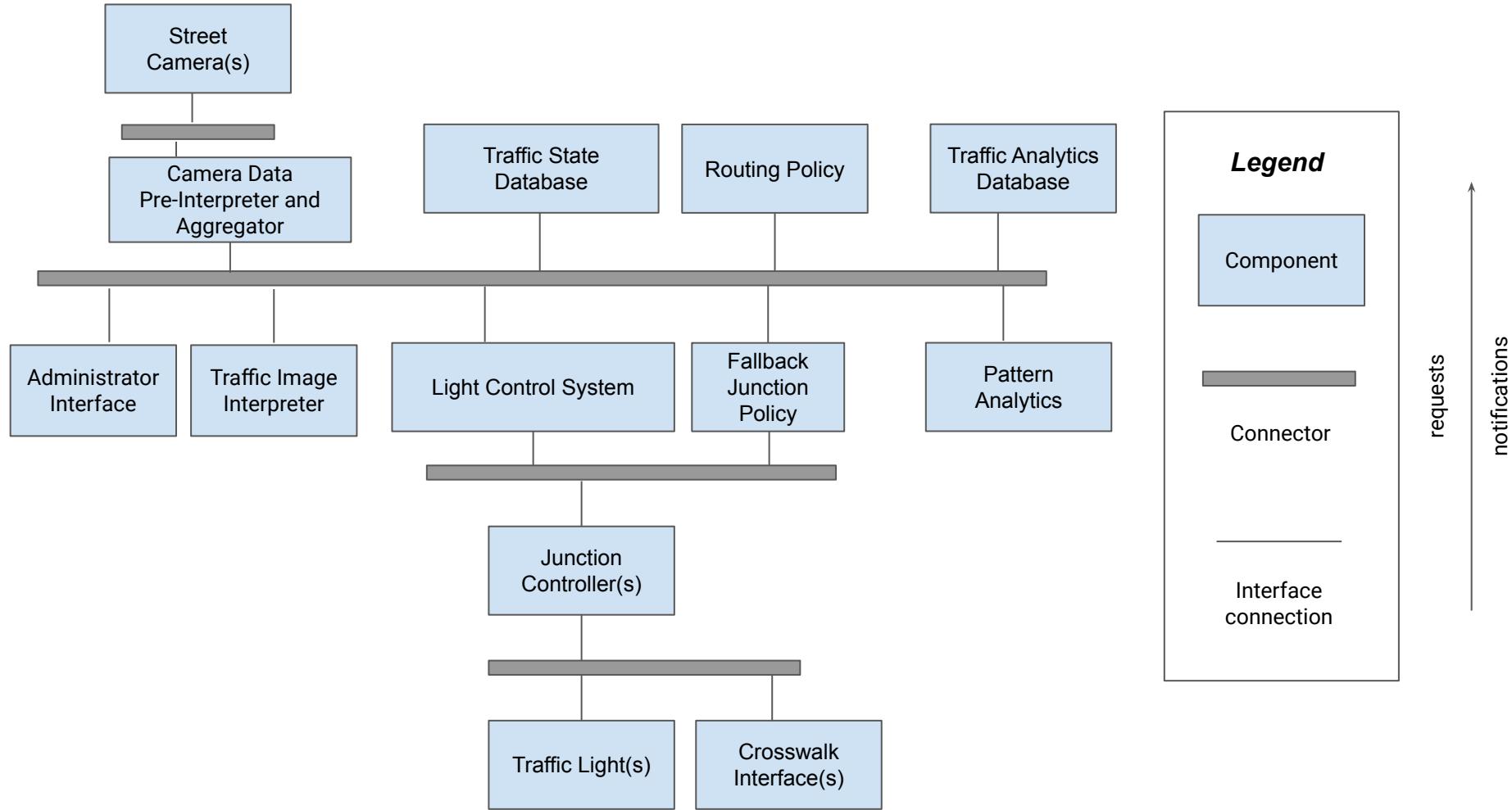
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:



Team Name: Team 9

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Using C2 Architecture

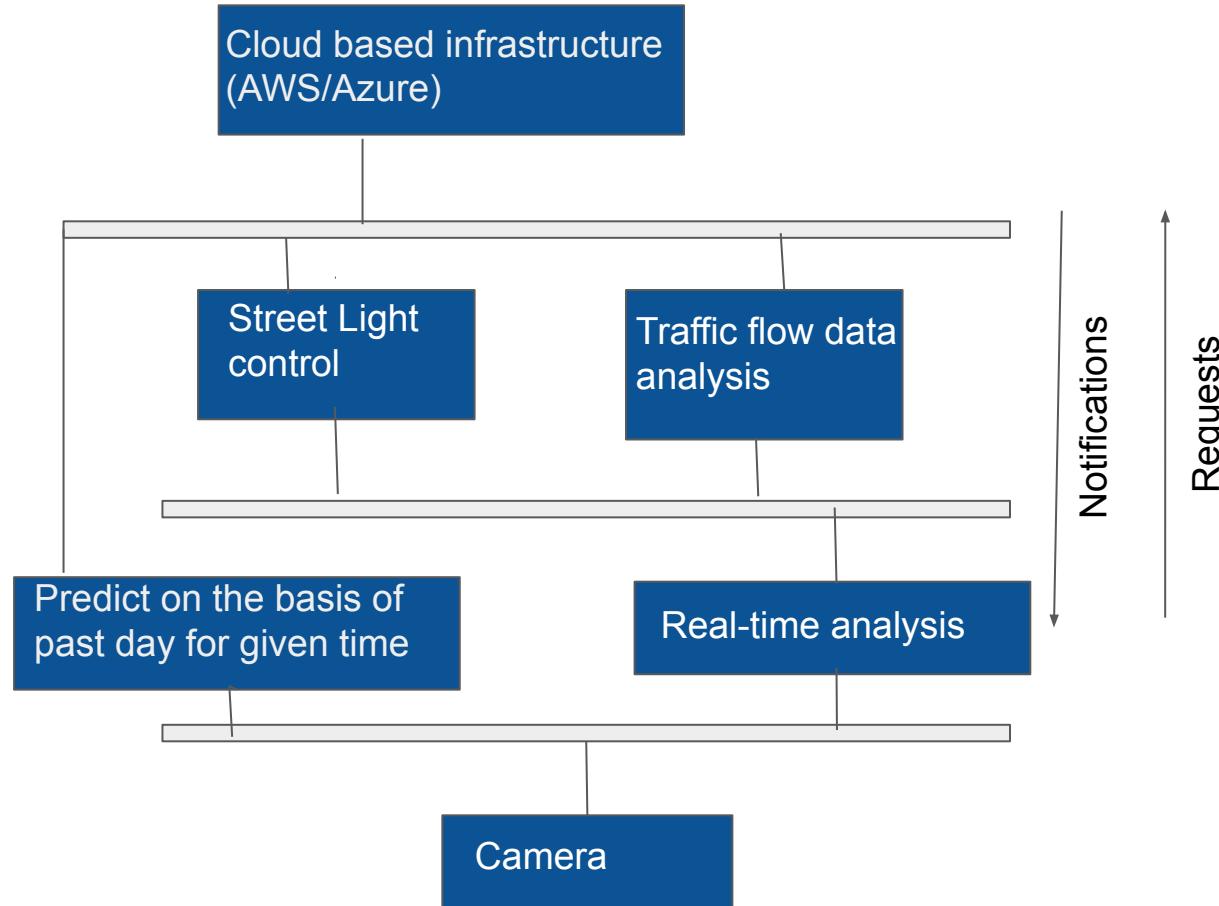
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Open Source Web Services

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

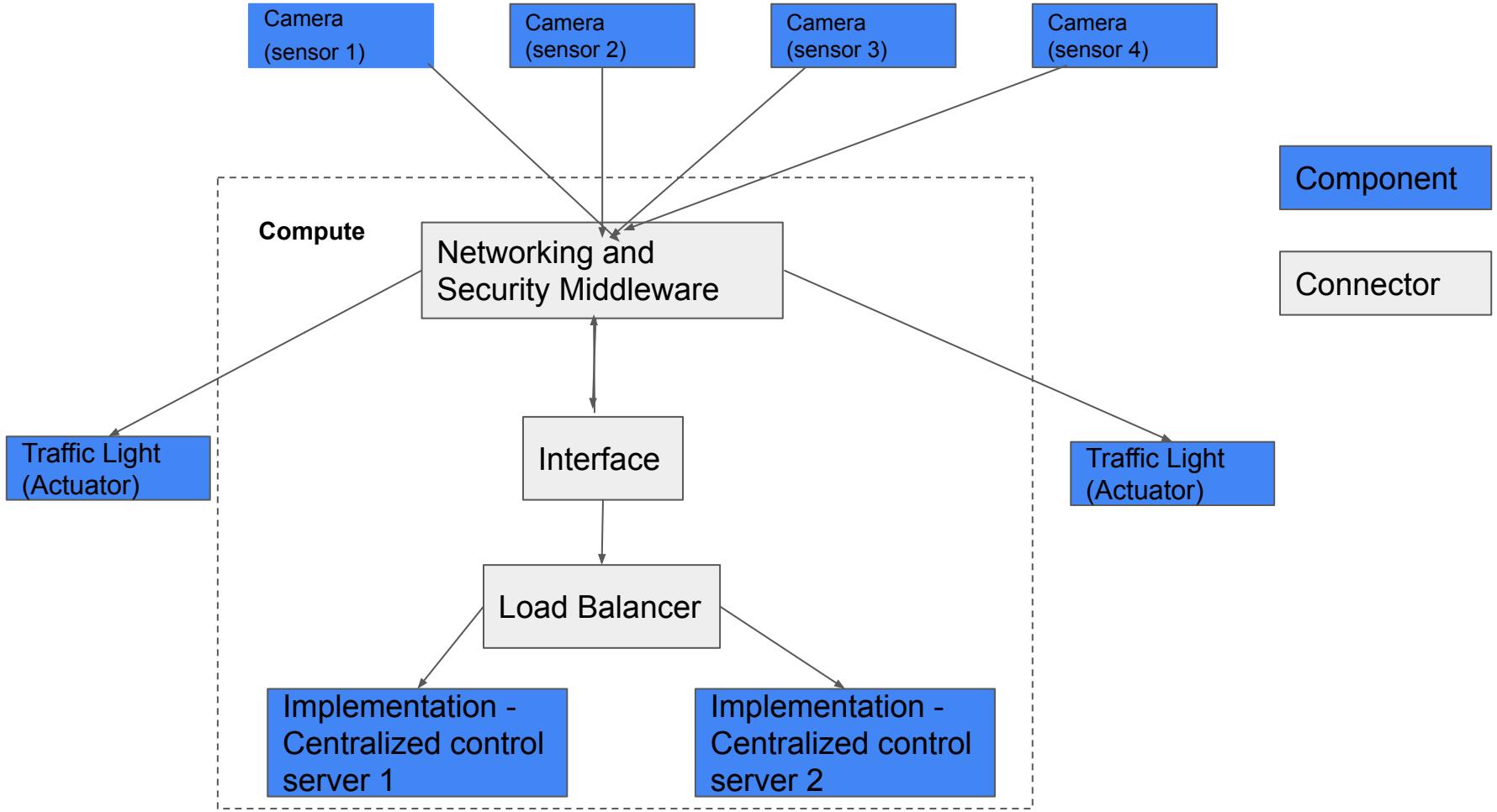
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



NETWORK-BASED APPLICATIONS



Commercial Internet-Scale Applications



- Akamai
 - Caching to the max
- Google
 - Google distributed file system (GFS)
 - MapReduce

Akamai



- Proxies – can cache data
- Problem: flash crowds
 - proxies unable to support frequent updates on a webserver & flash crowds
- NFP's required: scalability, reliability, performance
- Solution: replicate server at many different locations
 - “edge server”
- Backstory
 - Started as a research project at MIT – to reduce Internet congestion
 - 1998 – Company founded
 - In 2002, they had 12K servers in over 1K networks



<http://www.akamai.com/>

Akamai – Mapping to an Edge Server

- HTTP request to a7.g.akamai.net
- Resolver chooses a root name server and asks it to resolve the URL
- Root name server returns with IP addresses of name servers that handle .net domain requests
- Resolver queries the .net DNS for .akamai.net
- Return with Top-level DNS (S 1)
- Resolver queries Top-level DNS for .g.akamai.net;
- Return with low-level DNS (edge server)
- Resolver queries low-level DNS
- Returns IP address of content provider (TTL of sec/min) and requests content

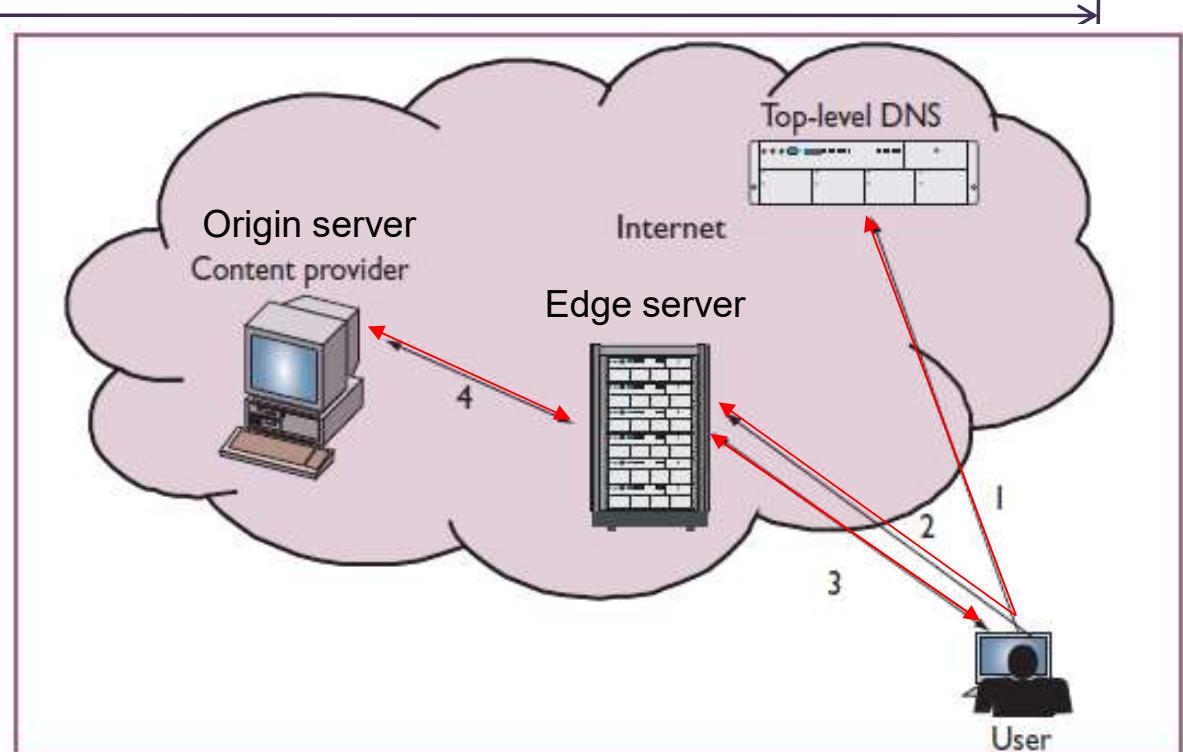
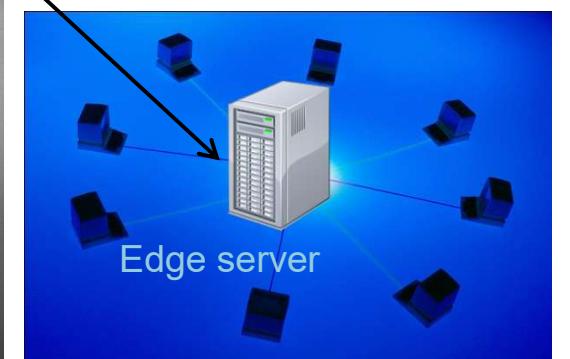
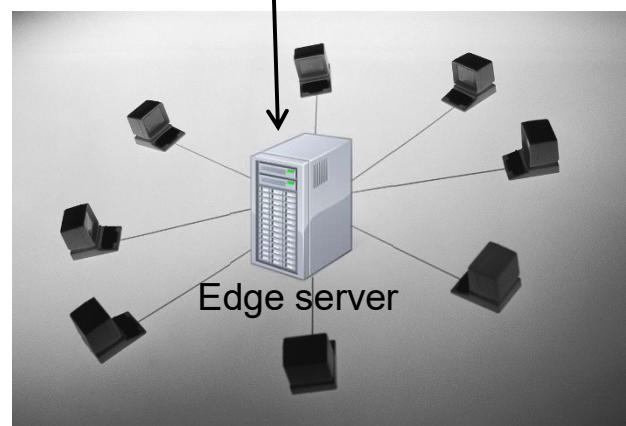
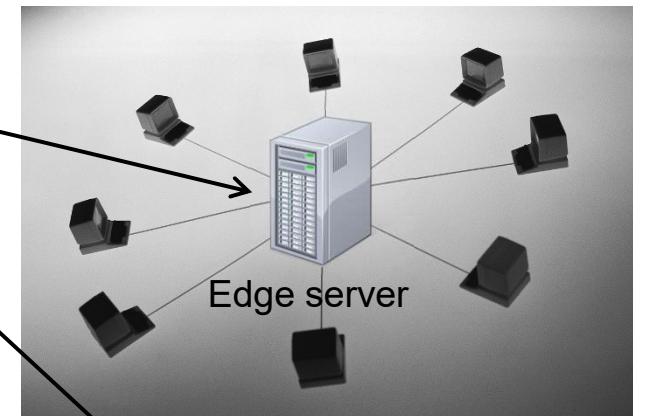
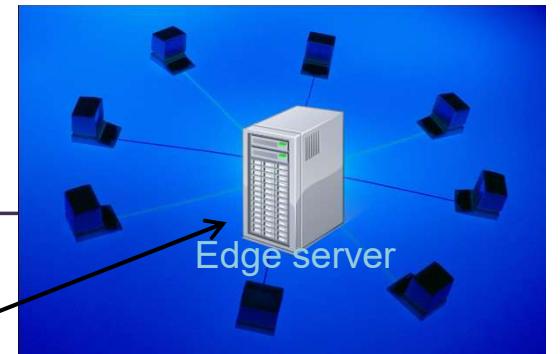
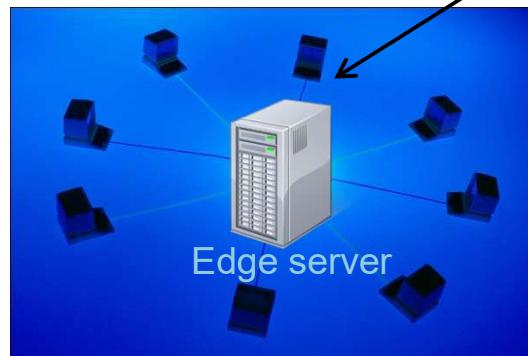


Figure 1. Client HTTP content request. Once DNS resolves the edge server's name (steps 1 and 2), the client request is issued to the edge server (step 3), which then requests content (if necessary) from the content provider's server, satisfies the request, and logs its completion.

Taken from <http://www.akamai.com/>

Akamai

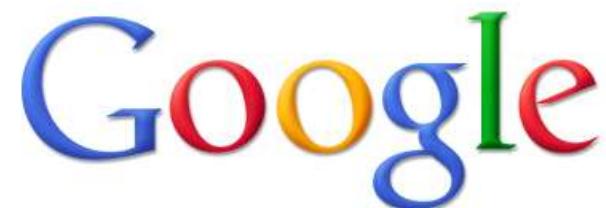
- Runs on top of REST Style
- Logically centralized (but distributed and replicated) administration



Google



- Ability to manipulate very large quantities of information
 - Terabytes of data from the Web are stored, studied, and manipulated
- Buy cheap and plan for failures

The Google logo, featuring the word "Google" in its signature multi-colored font: blue, red, yellow, and green.

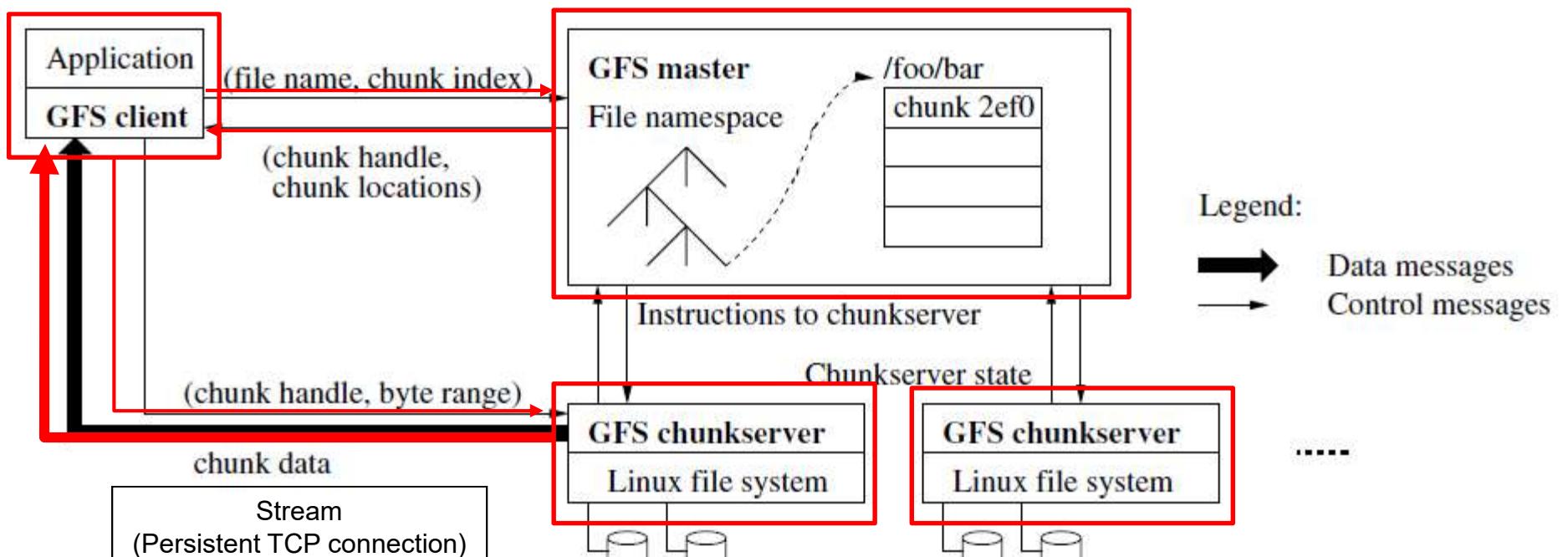
Google File System (GFS)



- Component failures are the norm
- Files are huge (multi-GB)
- Files are changed by appending new data, rather than overwriting existing data
 - No random writes
 - Sequential reads
- Co-designed applications and file system API
 - Allow multiple clients to write to a file
- Spreads a file's data across storage servers
- No caching below file system interface
- NFPs:
 - Performance, scalability, reliability, availability

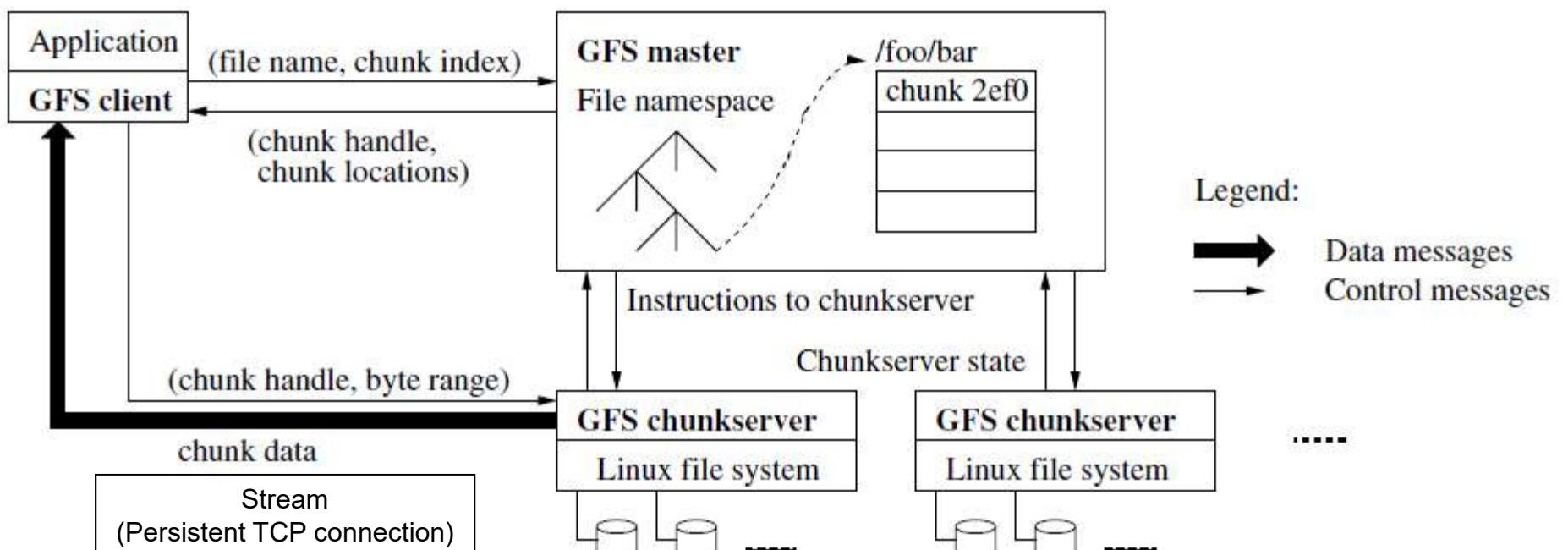
Google File System (GFS)

- GFS cluster consists of
 - One master
 - Multiple chunkservers
 - Multiple clients
- Avoid bottleneck at master
 - Decouple data from control flow
- Arch style?



Google File System (GFS)

- GFS cluster consists of
 - One master
 - Multiple chunkservers
 - Multiple clients
- Avoid bottleneck at master
 - Decouple data from control flow
- Arch style: Modified Client-Server



Google File System (GFS)



- Single master
 - Maintains all file system metadata (namespace, mapping files to chunks, locations of chunks)
 - System-wide tasks (e.g. garbage collection)
- Multiple chunkservers
 - Contains the data
 - Keeps track of replicas
- Multiple clients
 - Implements the file system API
 - Asks GFS master which chunkserver it should contact
 - Interacts with the chunkserver directly

Google: MapReduce



- Runs on top of GFS
- Data selection and reduction
- All parallelization done automatically
 - Provides a simple interface that can be used by programmers without experience in parallel and distributed systems

Architectural Lessons from Google



- **Abstraction layers abound:**
 - GFS hides details of data distribution and failure
 - MapReduce hides the intricacies of parallelizing operations
- By designing, from the outset, for **living with failure** of processing, storage, and network elements, a highly robust system can be created
- **Scale is everything:** Google's business demands that everything be built with scaling issues in mind

Architectural Lessons from Google (cont'd)



- By **specializing the design to the problem domain**, rather than taking the generic “industry standard” approach, high performance and very cost-effective solutions can be developed
- By **developing a general approach** (MapReduce) to the data extraction/reduction problem, a highly reusable service was created



CSS 553, Spring 2023, 9a Network





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



DECENTRALIZED ARCHITECTURES



Decentralized Architectures



- Networked applications where there are multiple authorities
- In other words
 - Computation is distributed
 - Parts of the network may behave differently, and vary over time

It's just like collaboration in the real world

The Grid World

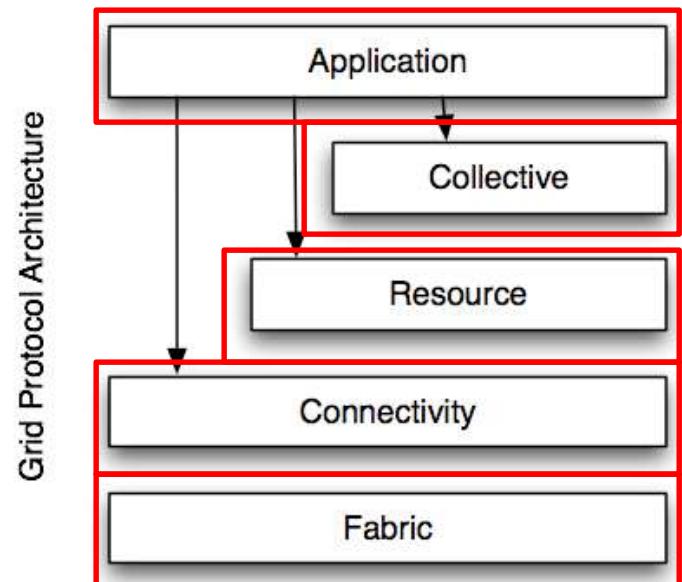


- Coordinated resource sharing in a decentralized environment
 - Allow a team of researchers to temporarily bring together computing resources to work on a problem
 - Behave as if under one authority at this time
 - Single sign-on – one of the goals
 - E.g. Folding@home
 - <https://foldingathome.org/>

Grid Protocol Architecture

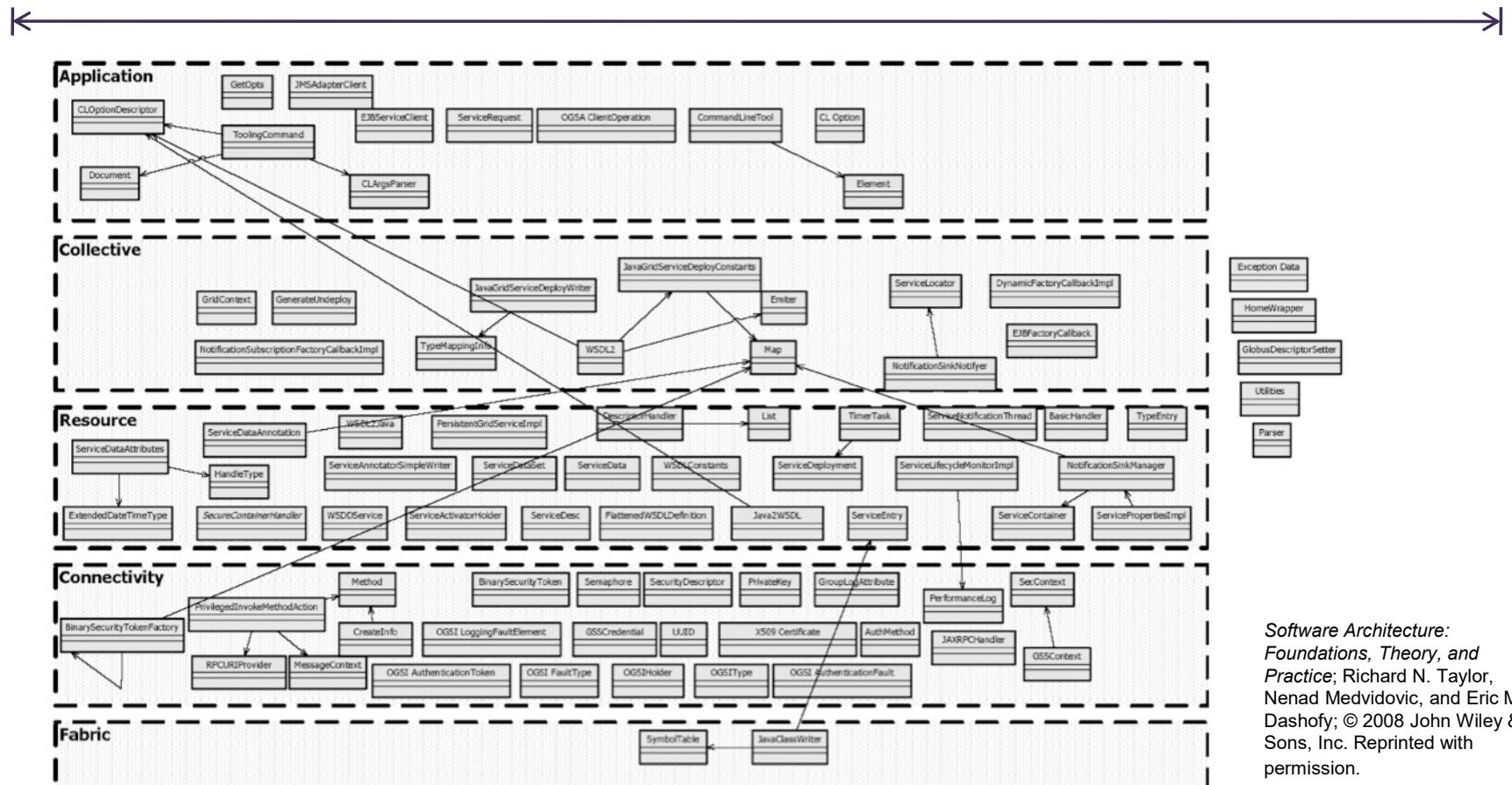


- GLOBUS
 - A commonly used infrastructure
- Arch-Style?
 - Layered
- “Standard architecture”
 - Fabric manages low-level resources
 - Connectivity: communication and authentication
 - Resource: sharing of a single resource
 - Collective: coordinating resource usage
 - Application: implements a user’s system



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

Grid GLOBUS (Recovered)



Peer-to-Peer Architectures

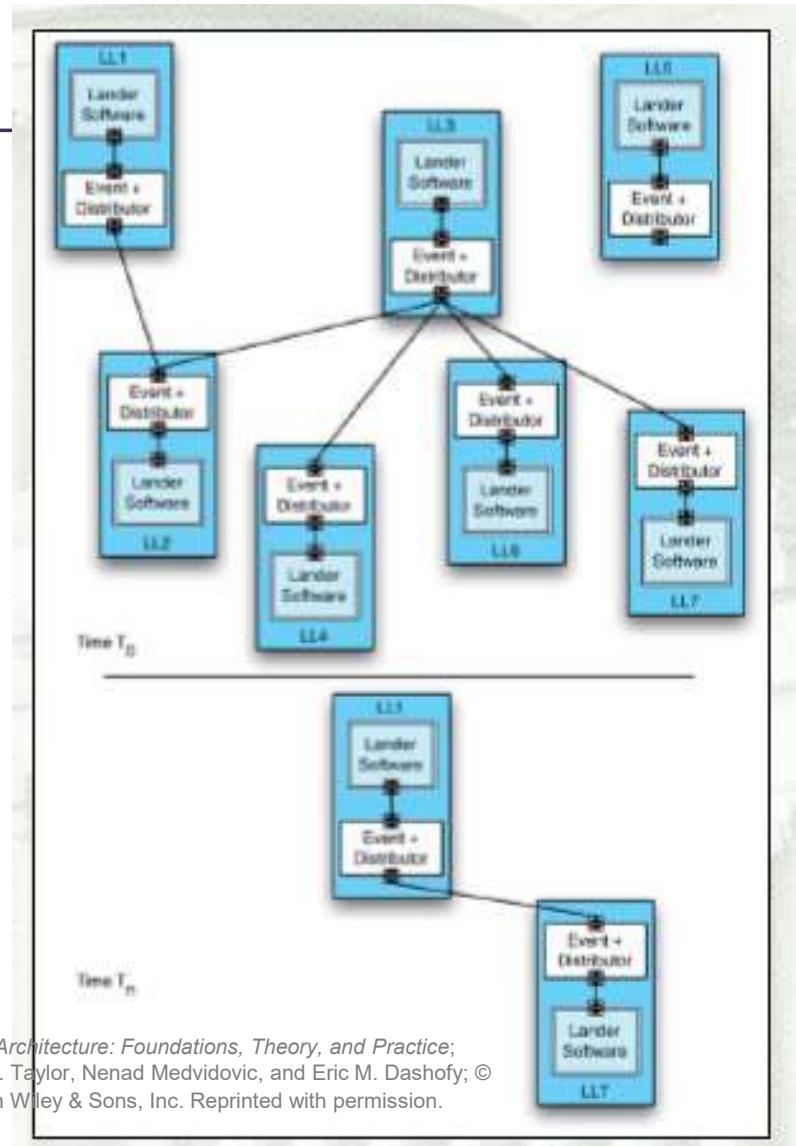


- Decentralized resource sharing and discovery
 - Napster
 - Gnutella
 - Skype
 - BitTorrent

Peer-to-Peer LL



- State and behavior are distributed among peers which can act as either clients or servers.
- NFPs:
 - Highly robust – failure of a given node
 - Scalable - access to resources and computing power.



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.

Peer-to-Peer Style

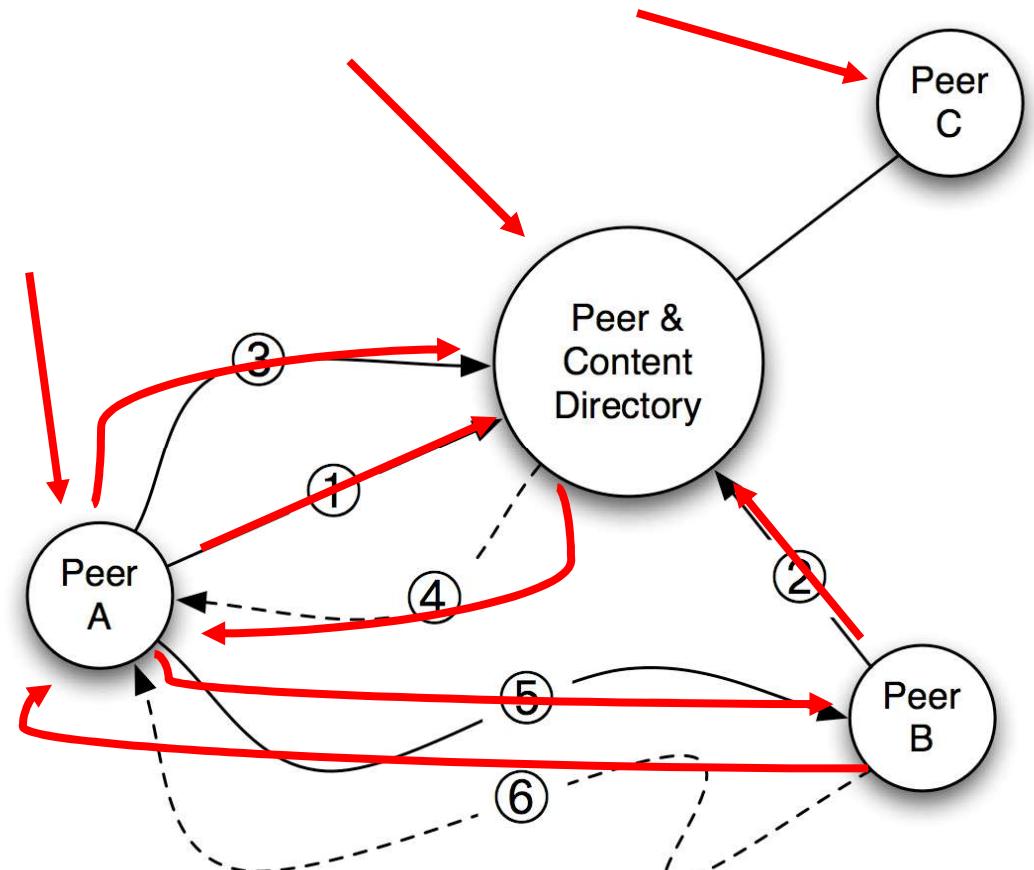


- Components: independent peers, having their own state and control thread.
- Connectors: Network protocols, often custom.
- Data Elements: Network messages
- Topology: Network (may have redundant connections between peers); can vary arbitrarily and dynamically

Napster

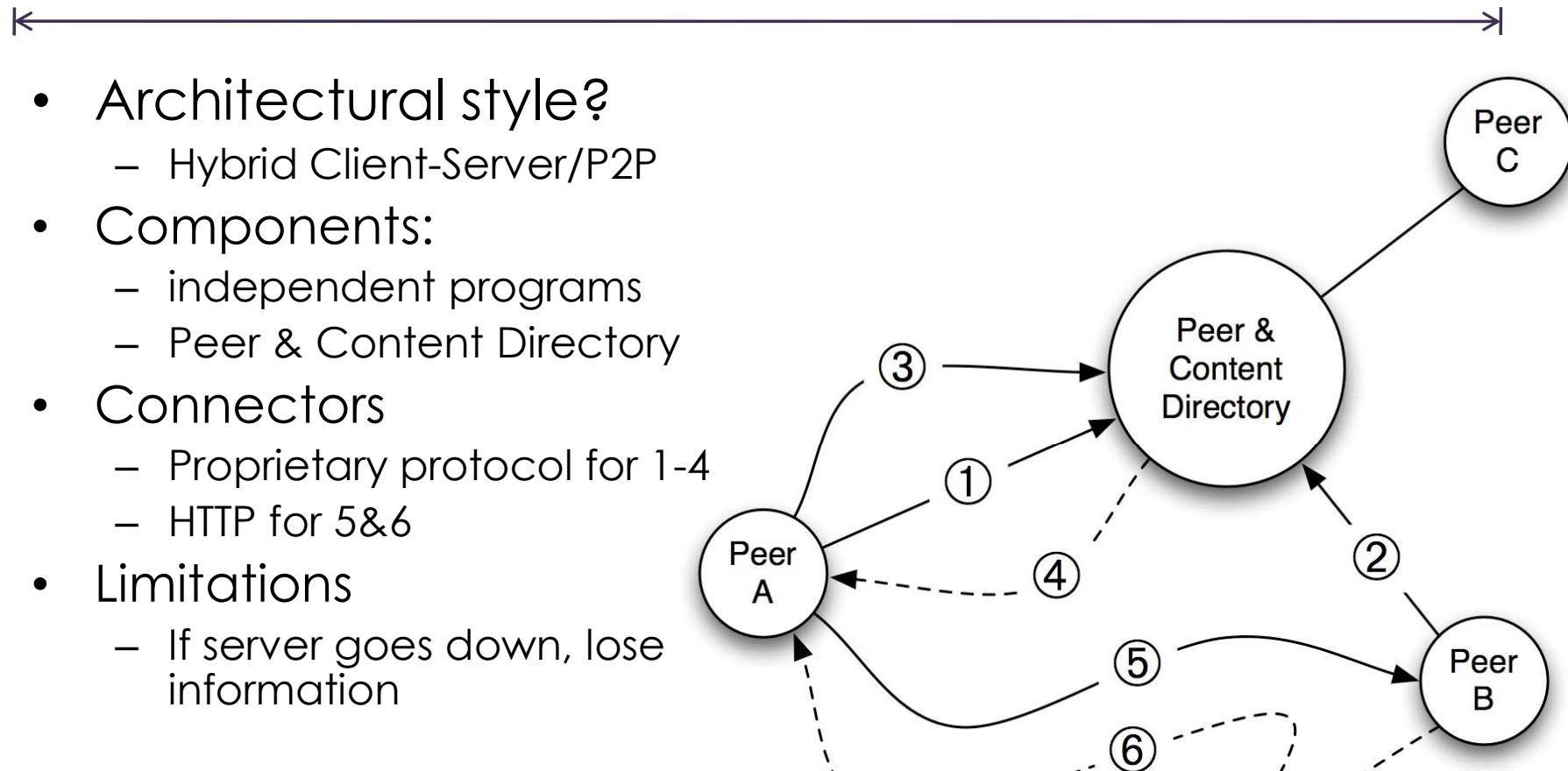


- Architectural style?
 -
- Components:
 -
- Connectors
 -
- Limitations
 -



*Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.*

Napster

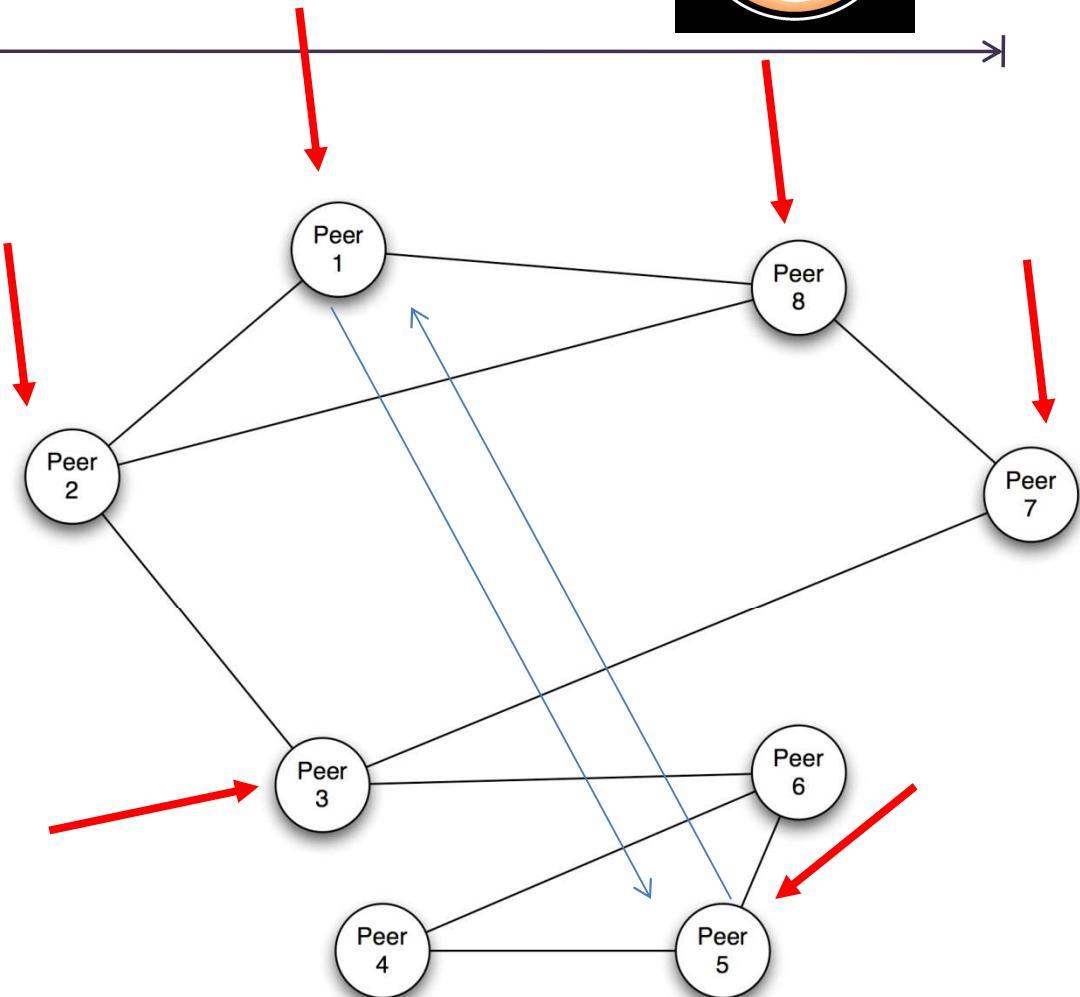


*Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.*

Gnutella (original)



- Architectural style?
 -
- Components:
 -
- Connectors
 -
- Limitations
 -

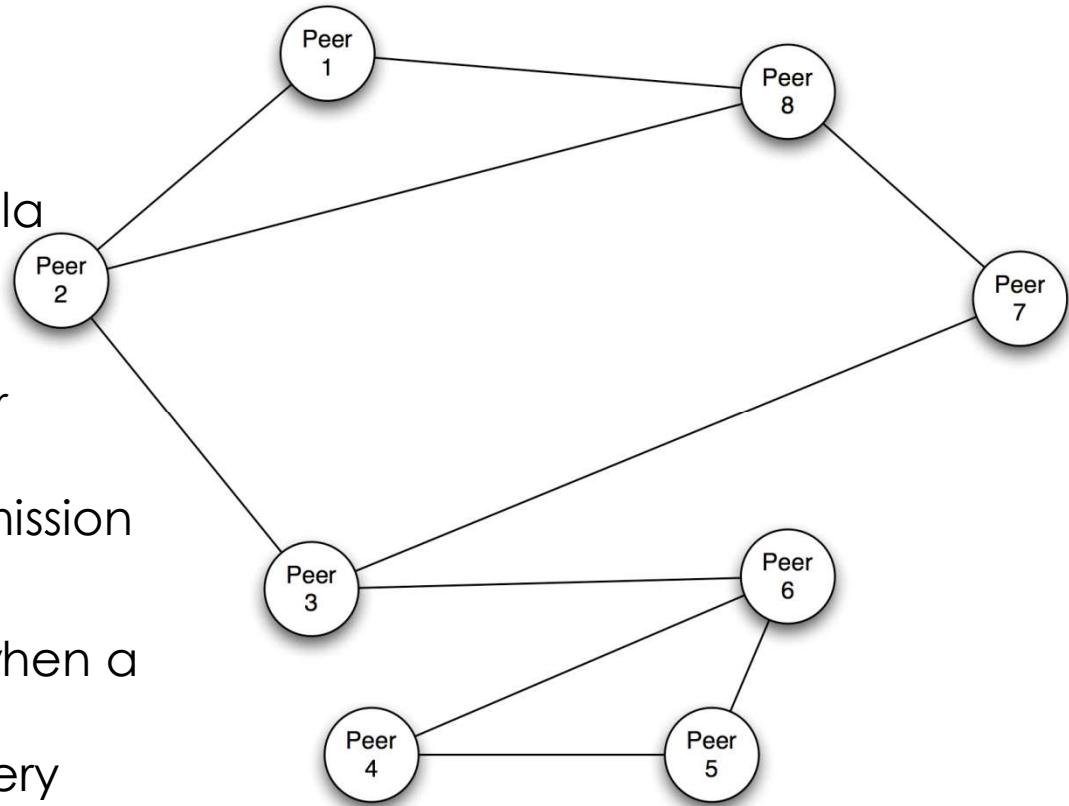


*Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.*

Gnutella (original)



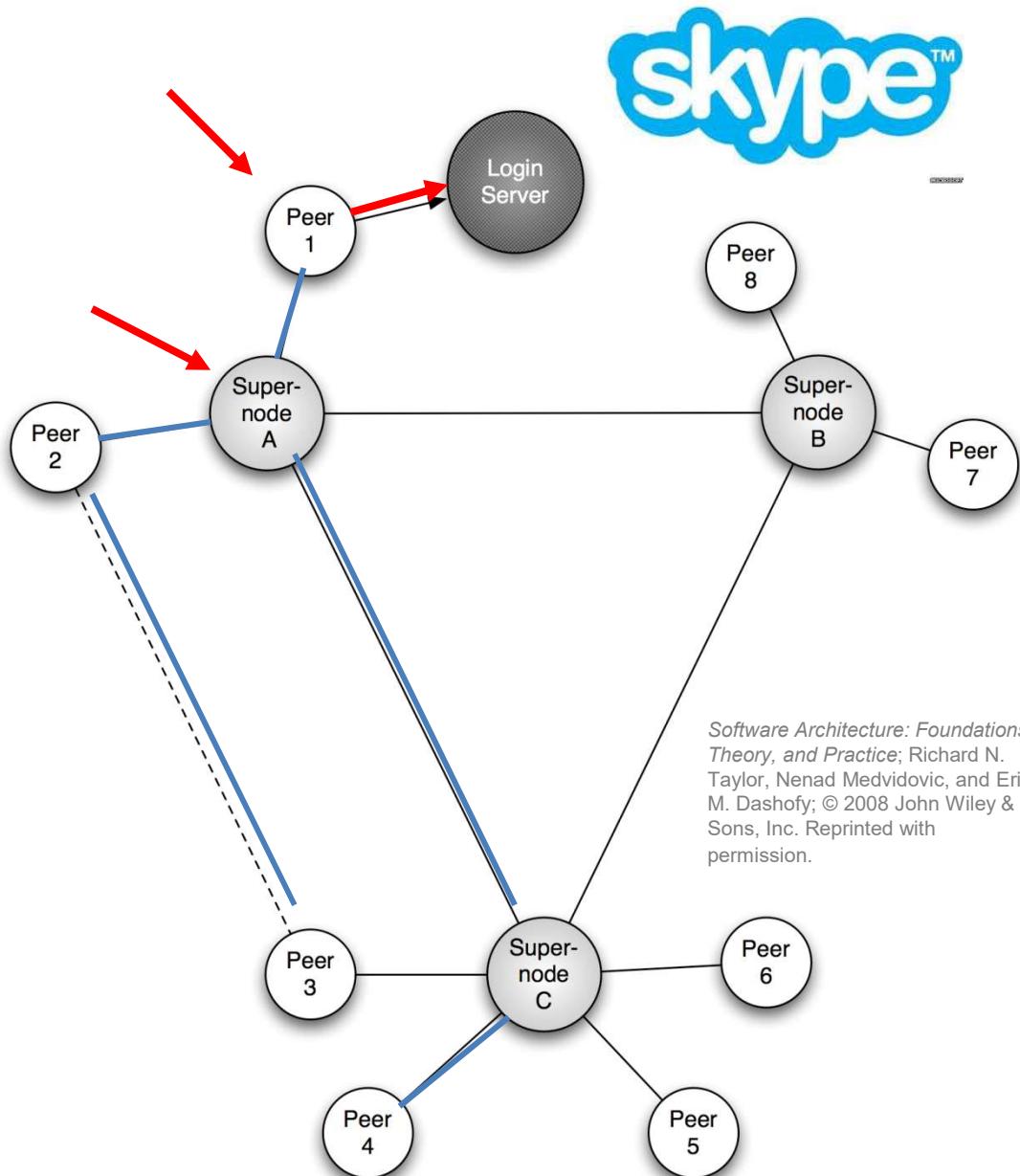
- Architectural style?
 - Pure P2P
- Components:
 - Run software that implements the Gnutella protocol
- Connectors
 - Proprietary protocol for discovery
 - HTTP for content transmission
- Limitations
 - How to inform others when a new peer comes on?
 - How long is the discovery process?
 - Open environment – how to avoid viruses/malware?



*Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; ©
2008 John Wiley & Sons, Inc. Reprinted with permission.*

Skype (original)

- Architectural style?
 -
- Components:
 -
- Connectors
 -

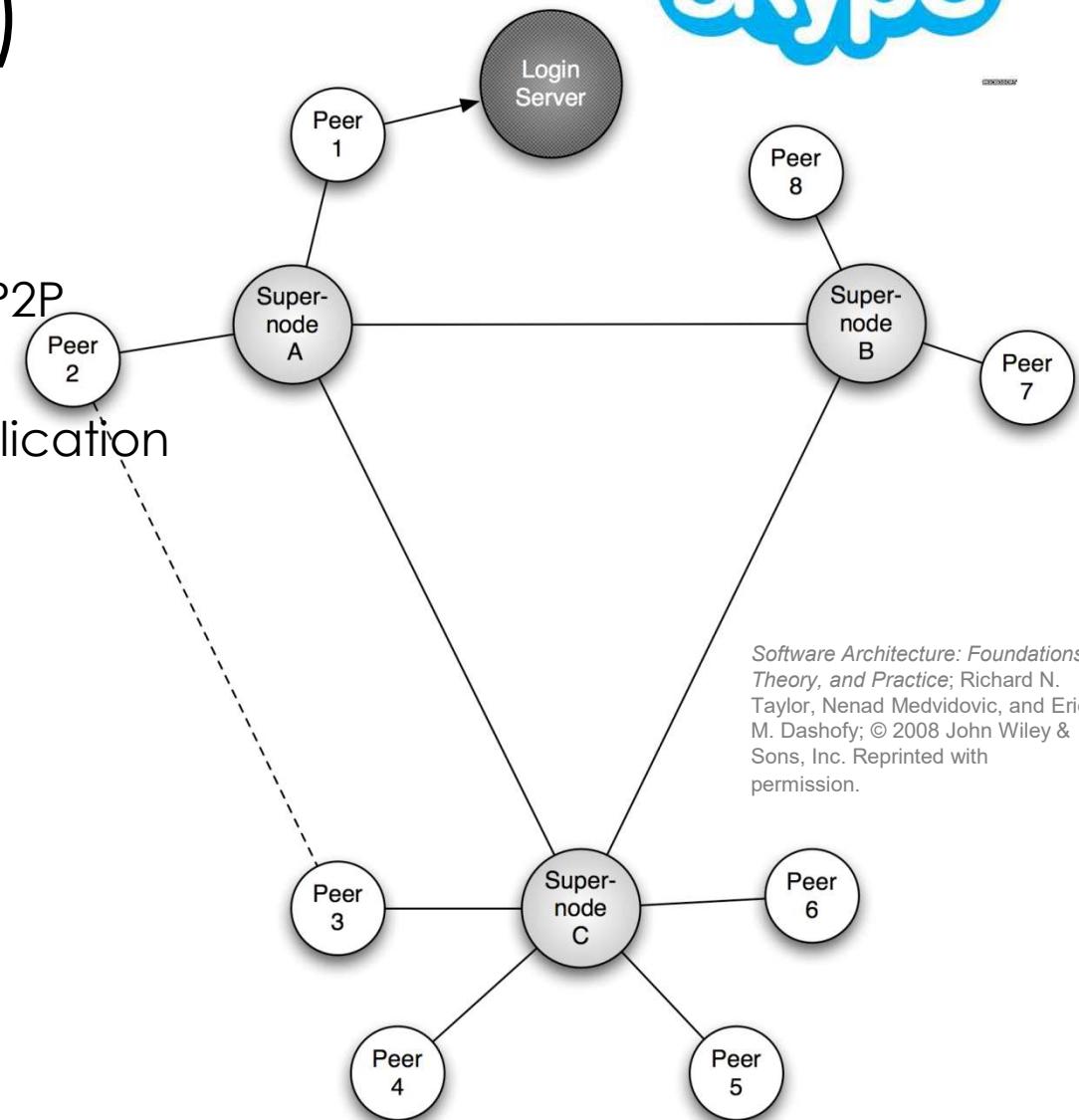


Skype (original)



←

- Architectural style?
 - Mixed Client-Server / P2P
- Components:
 - Proprietary Skype application (no open source implementation)
 - Supernodes
 - Login Server
- Connectors
 - Proprietary protocol



Insights from Skype



- Discovery problem (Gnutella)
 - mixed client-server and peer-to-peer architecture
- Scalability/robustness problem in Napster:
 - Replication and distribution of the directories, in the form of supernodes,
- Malicious clients
 - Restriction of participants to clients issued by Skype
- Latency and slow performance
 - Promotion of ordinary peers to supernodes
- Lack of privacy in calls
 - A proprietary protocol employing encryption

BitTorrent



- P2P application – support replication of large files
- Address problem of flash crowds:
 - Distribute parts of a file to many peers
 - Requesting peer responsible to also pass the file on to other requesting peers
- Principal Design Decisions
 - Discovery – driven by the user (Search engine / App Studio client)
 - Centralized machine (“tracker”) – in charge of tracking where the file and parts of the file are distributed
 - Meta-data associated with file – used to piece the parts of the file together
 - Each peer runs a BitTorrent application
 - What piece to download next?
 - Which peer to obtain this next piece?



CSS 553, Spring 2023, 9b Decentralized





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



SERVICE-ORIENTED ARCHITECTURES

Service-Oriented Architectures (SOA) & Web Services



- SOA - Directed at supporting business enterprises on the Internet
- Web Services – one way of providing SOAs
 - Offers various approaches, standards, and technologies
- How to describe these architectures?
 - Components?
 - Connectors?
 - Data?
 - Application as a whole?

Service-Oriented Architectures (SOA) & Web Services

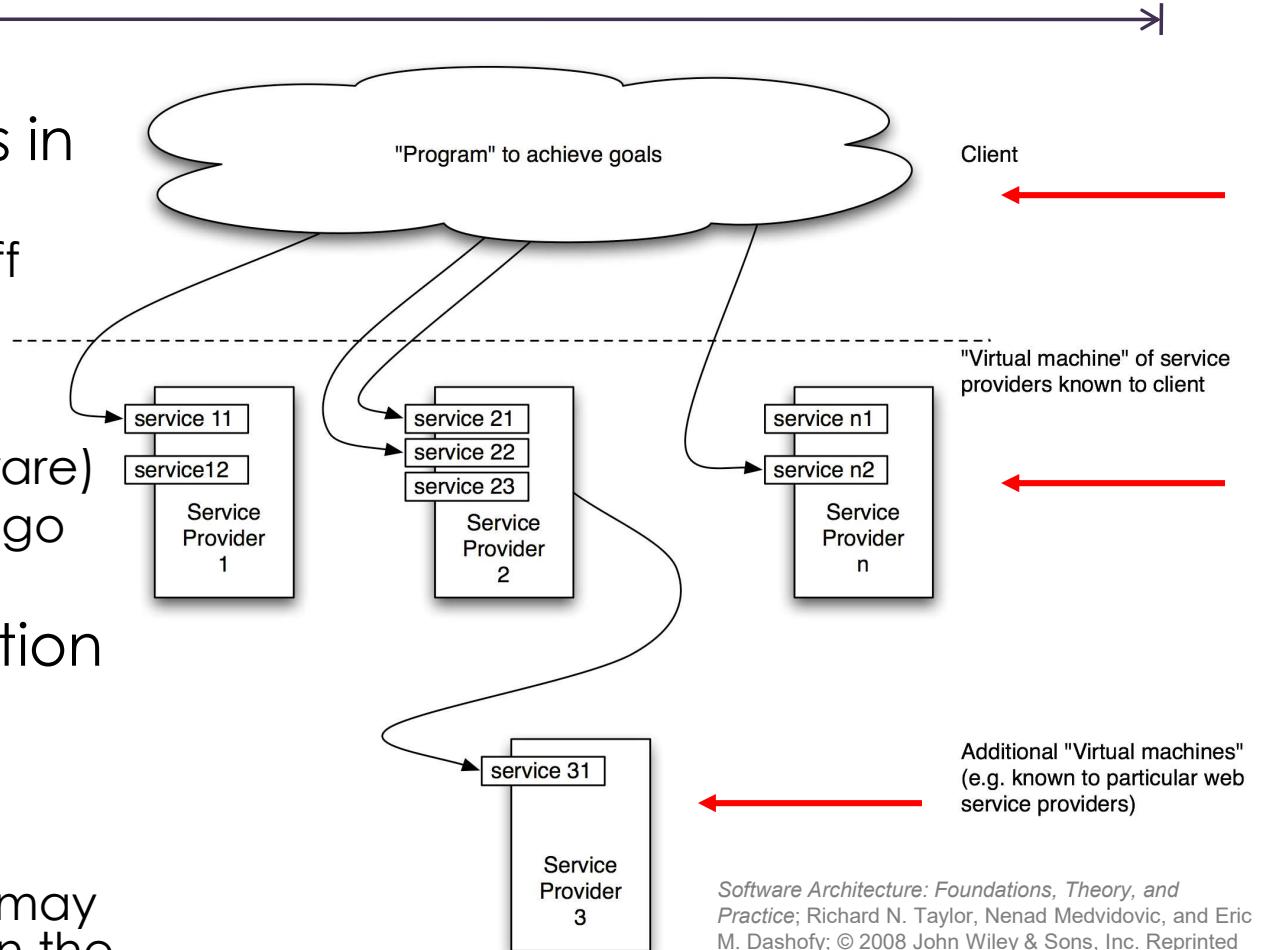


- SOA - Directed at supporting business enterprises on the Internet
- Web Services – one way of providing SOAs
 - Offers various approaches, standards, and technologies
- How to describe these architectures?
 - Components? Services
 - Connectors? SOAP: asynchronous event / asynchronous request-response + stream + distributor (e.g. TCP/ HTTP)
 - Data? XML documents
 - Application as a whole?
 - WS-BPEL – Web Services Business Process Execution Language

Service-Oriented Architectures & Web Services



- Differences with Virtual Machines in Ch 4: Services...
 - controlled by diff organizations
 - implemented differently
 - have risks (malware)
 - may come and go
- WSDL – Web Services Description Language
 - Describes the services as a collection of operations that may be performed on the typed data sent to/from the service



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

Service-Oriented Architectures & Web Services



- WSDL - Example of a service providing stock quotes

```
<?xml version="1.0"?>
<definitions name="StockQuote"
  targetNamespace="http://example.com/stockquote.wsdl",
  xmlns:tns="http://example.com/stockquote.wsdl",
  xmlns:xsd1="http://example.com/stockquote.xsd",
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/",
  xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
  <schema targetNamespace="http://example.com/stockquote.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="TradePriceRequest">
      <complexType>
        <all> <element name="tickerSymbol" type="string"/> </all>
      </complexType>
    </element>
    <element name="TradePrice">
      <complexType>
        <all> <element name="price" type="float"/> </all>
      </complexType>
    </element>
  </schema>
</types>

<message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>
```

<http://www.w3.org/TR/wsdl>

CSS 553, Spring 2023, 9c SOA



Service-Oriented Architectures & Web Services



```
<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
</portType>

<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
    <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>

<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>

</definitions>
```

<http://www.w3.org/TR/wsdl>

Web Services



- Web services != distributed objects
- Interchange of XML documents in a form that can be understood by both parties
- Envelope – encapsulates the messages
 - SOAP envelope
 - Soap Header – system info
 - Soap Body – XML document to process
- SOAP != Simple Object Access Protocol

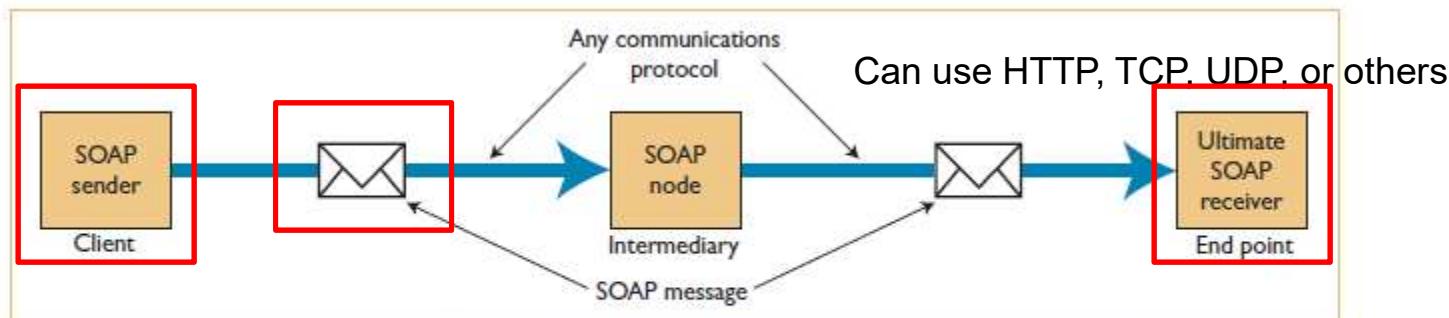


Figure 1. SOAP's transport independence. Web service documents encapsulated in a SOAP message can be delivered directly to a destination over a single transport or via a collection of intermediaries over a variety of transports.

Vogels, W., Web Services are not Distributed Objects, IEEE Internet Computing, 2003

Web Services



- SOAP request example

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
```

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Body xmlns:m="http://www.example.org/stock">    Namespace of function
        <m:GetStockPrice>                                Request
            <m:StockName>IBM</m:StockName>                Request Parameter
        </m:GetStockPrice>
    </soap:Body>

</soap:Envelope>
```

<http://www.w3schools.com/>

More about Web Services



- RESTful web services
 - HTTP is an application protocol
 - Does not send bits
 - Transfers representational state
 - If the body of a POST or PUT is not a piece of representational state, not using REST as an application protocol
 - List of HTTP methods fixed for all resources

Courtesy of Dr. Yongjie Zheng
California State University San Marcos



More about Web Services



- Non-RESTful web services
 - Treat HTTP as a transport protocol like TCP
 - Build new protocols (SOAP) and tunnel over existing application protocols (HTTP)

Courtesy of Dr. Yongjie Zheng
California State University San Marcos

Questions



- What architecture style (or combined styles) is used to address the issue of flash crowds? Explain how this style addresses this challenge.
- Explain at least two modifications to Client-Server style to enable it to handle processing terabytes of data.
- You were given lessons learned from products developed at Google. Identify **one** lesson you learned from the other systems discussed this week. Explain how you can apply this lesson when designing software.
- What is a difference between distributed and decentralized computing? What architecture style(s) support decentralized computing and how?
- Identify **one** issue with pure P2P systems. How can this issue be addressed?
- Is SOAP the same as distributed objects? Explain why or why not.
- Pick any open source project. Is it scalable? Explain what you mean by scalability in terms of this project. If it is not scalable, what design change would you need to do to make it scalable?

READ THIS FIRST - 20 minutes

Topic: Architecture Styles

This activity is designed to

- Give you practice in designing
- Tie up loose ends regarding upcoming assignment(s)

Roles:

• **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.

• **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.

• **Note Taker** - takes record of the group’s discussion in the Google doc (below).

• **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity or the assignments due tonight, type in Red text

All teams will do the following task:

Previous design exercise

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

....using whatever method you like...

....using whatever architectural style / pattern we covered thus far in class...

....Cost is a concern...and this is the final design that you will hand off to your development team

Reflection

•How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?

•Who did you keep in mind when making your design?

•What was your goal with your design?

•Did you have more than one goal?

Did you reach the goal(s) with your design?

Team Name: Dawn

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

A four-tier architecture is a well-established software design pattern that can be employed in this scenario. It involves the segregation of an application into four logical layers: the Hardware Layer, the Presentation Layer, the Business Logic Layer, and the Data Access Layer. Here's a breakdown of how this might work for the traffic control system.

1. Hardware Layer

This layer will connect the data from hardware, such as traffic lights, cameras. This layer is designed for data collection.

2. Presentation Layer

This would be the user interface of the system, used by traffic managers or other administrators to monitor and manage the traffic control system. It would display real-time and historical traffic data, system status, and potentially allow manual override of traffic signals when necessary. This layer would be a web-based interface, ensuring accessibility from multiple devices and locations.

List Roles and Student names

Moderator : Chloe

TimeKeeper: Abdul

NoteTaker: Luo

Reporter: Barack

3. Business Logic Layer

This is the core of the application, containing all the logic for processing input data and controlling the traffic signals. It can be divided into several components, each handling a specific aspect of the system:

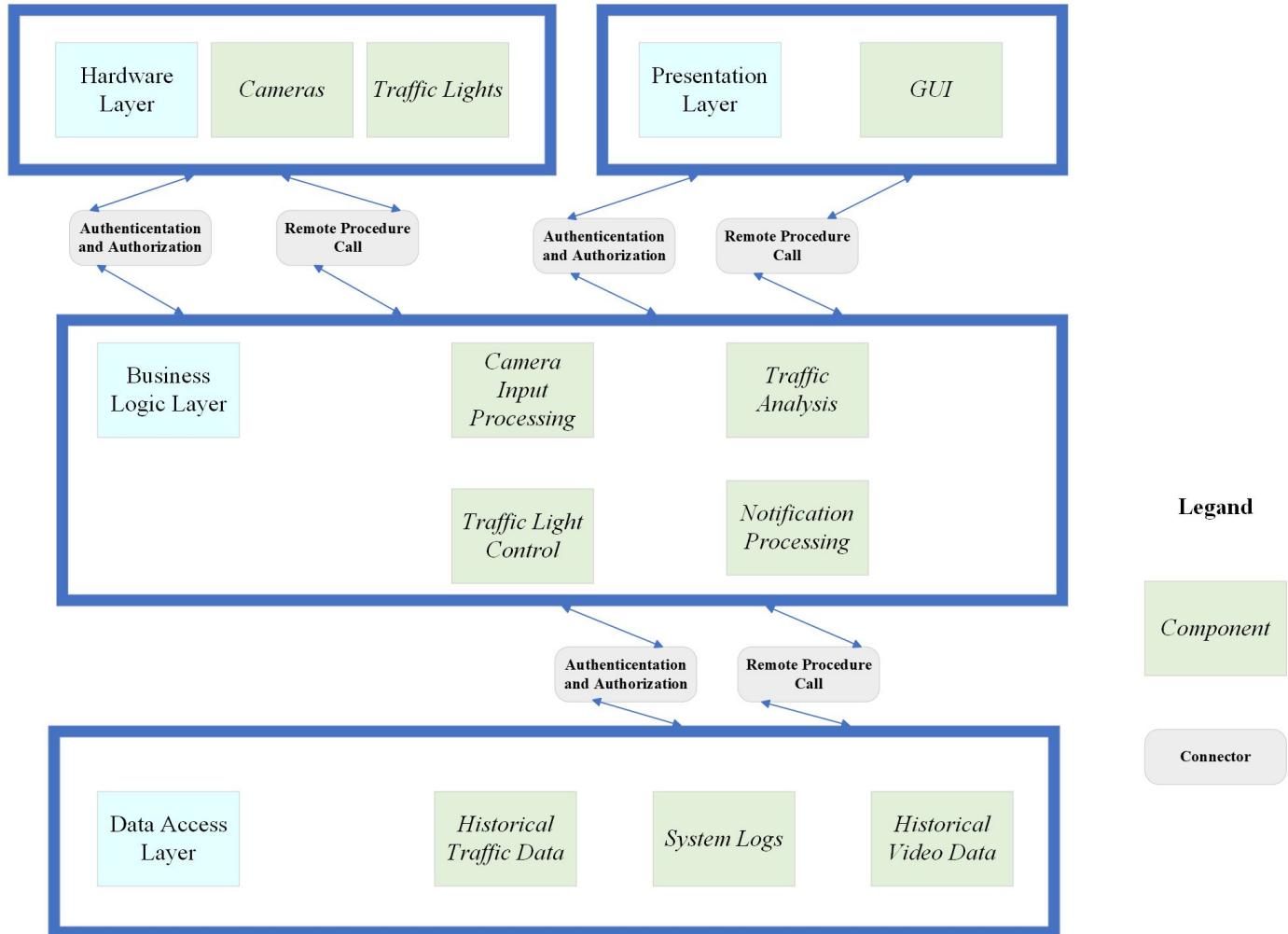
Camera Input Processing Component: This component would take real-time feeds from street cameras, preprocess the data (e.g., image enhancement, object detection), and forward it to the Traffic Analysis Component.

Traffic Analysis Component: This component would analyze the preprocessed data from the Camera Input Processing Component to identify the current state of traffic at different intersections. Techniques such as machine learning and computer vision can be used to understand traffic density and flow patterns.

Traffic Light Control Component: This component would receive the traffic state information from the Traffic Analysis Component and use it to control the traffic lights. The decision logic can be based on algorithms designed to optimize traffic flow and reduce congestion.

4. Data Access Layer

This layer would handle all interactions with the system's data storage, ensuring data consistency and integrity. It would store all necessary data such as historical traffic data, system logs, and so on. A mix of SQL and NoSQL databases could be used, depending on the specific requirements of each type of data.



•How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?

Comparatively, when juxtaposed with the previous exercise, we would categorize this task as having an average level of difficulty. While it required more intricate system understanding and a broader view of potential scenarios, it was also a valuable opportunity to apply and expand on the knowledge and skills gained from previous tasks.

•What did you keep in mind when making your design?

When designing the software, we focused on several key areas: understanding and meeting the product's objectives and requirements, and ensuring the software architecture supports these requirements. This approach included anticipating potential changes, scaling needs, and potential edge cases that could arise during the software's lifespan.

•What was your goal with your design?

The main goal was to design software capable of controlling the city's traffic light system effectively and reliably. This objective required creating a blueprint that adheres to established rules and standards. Additionally, given the critical nature of the system, I had to account for its longevity and robustness, ensuring that it could operate without fail over an extended period.

•Did you have more than one goal?

Absolutely, the design process catered to multiple goals. Beyond functional requirements, such as managing the traffic lights effectively, non-functional requirements also came into play. These included ensuring the system's robustness, scalability, ease of maintenance, and cost efficiency.

•Did you reach the goal(s) with your design?

Indeed, the current design successfully meets both functional and non-functional goals.

For functional requirements like controlling the traffic lights, the system considers aspects such as data collection, analysis, storage, and presentation, thereby facilitating efficient and accurate traffic management.

Regarding non-functional requirements, the system has been designed with high modularity and low coupling, promoting robustness. This design means that if a component fails, it can be replaced swiftly without significant impact on the overall system. The modularity also allows for quick error detection and resolution, ensuring minimal downtime for software fixes or patches.

The system's architecture was chosen with longevity in mind, providing stability for a lifespan of at least five years. Additionally, the design optimizes cost efficiency by maintaining clarity and simplicity, eliminating unnecessary components that could drive up costs and complicate maintenance.

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Reflection

- How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?

The design was more difficult to think about this time around, given the additional constraints on cost, timeline, and deliverables. Having to present something ready for the development team that took into consideration a budget and must be robust enough for a 5 year lifetime meant that some of our assumptions needed to change to handle real-world data loads.

- Who did you keep in mind when making your design?

We were mainly keeping in mind developers who would be working on this system and the funding we would receive. We made it as modular as possible so that improvements on the system fall in line with the general architecture and specific behavior patterns could be implemented as needed.

- What was your goal with your design?

Our goal with this design was to make a robust system that would require minimal cost, easy update, long lifetime, and high modularity for future improvement. Given that legislation for traffic ordinances does not change often but traffic patterns often can and do, we wanted to create a framework that allows for a simplistic and working implementation, while still leaving room for improvement without architecture degradation.

- Did you have more than one goal?

Yes, we also wanted to try and reach the cost goal, but we were unable to verify that this was met given that no budget was specified. We believe that our design was minimally complex, which should reduce the cost of implementing the system.

- Did you reach the goal(s) with your design?

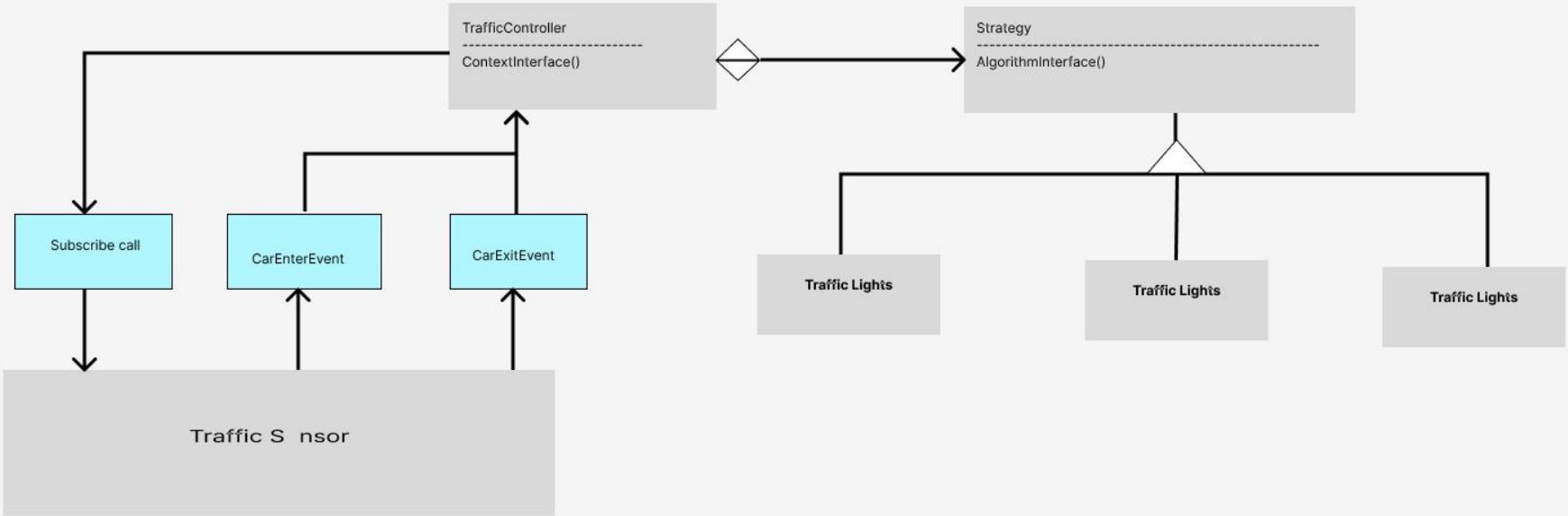
Yes, the proposed design allows for easy additions/modifications to the subscriber events, in case of needed updates. It also handles the bare minimum needed for functionality so we don't need to modify any large swaths of the design for maintenance or updates.

List Roles and Student names

Moderator : Ioana Preda

TimeKeeper: Holly East

NoteTaker: Jaron Want



Team Name: To be decided

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Previous design exercise

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches Traffic light - Camera - Traffic lights control system - load balancing server
- Functional for at least 5 years

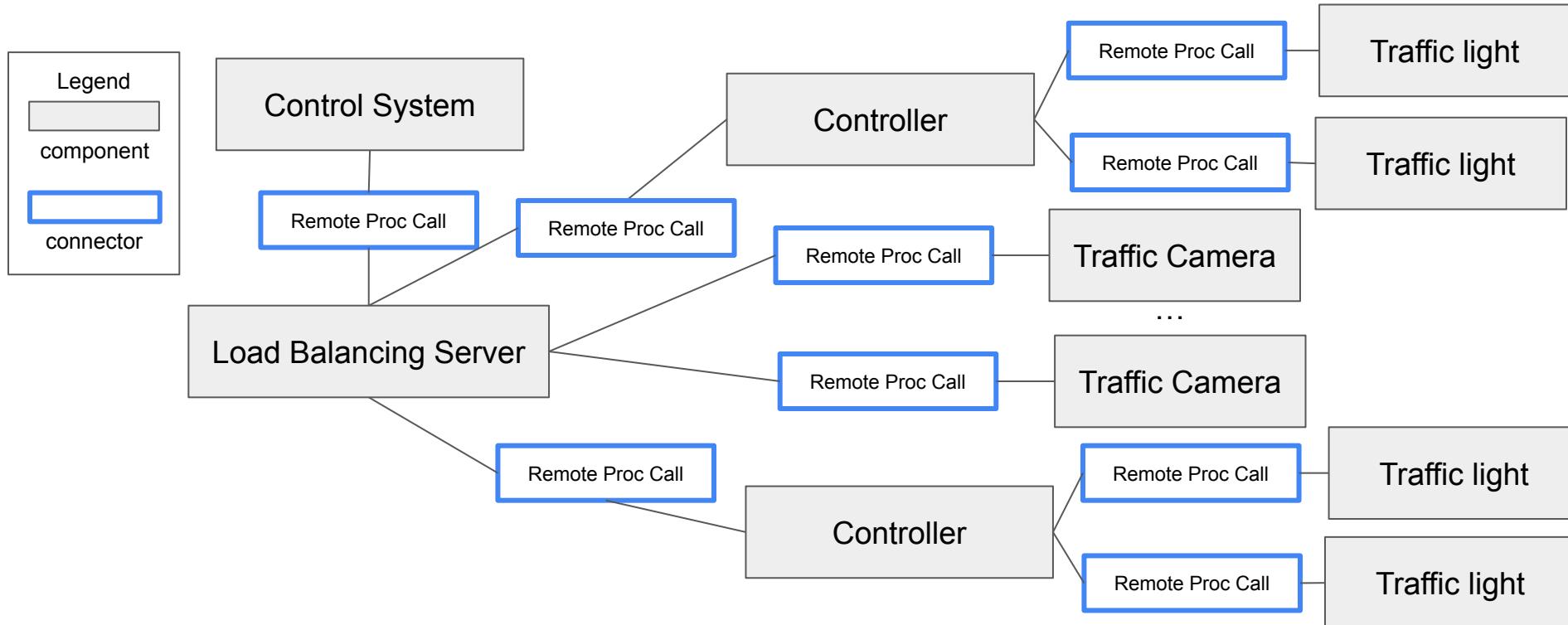
Create a design for this software...

- ...using whatever method you like...
- ...using whatever architectural style / pattern we covered thus far in class...
- ...Cost is a concern...and this is the final design that you will hand off to your development team

List Roles and Student names

REST style; its standardized and general interfaces allows for extendibility (something that would remain in tact for over 5 years). We want to implement **load balancing** for the sake of facilitating the higher-order more complex interaction in terms of the flow of control

- prevents the control system from being overwhelmed by a specific stream of data
- Remote Procedure Calls coordinate, communicate, and facilitate the data
- each set of traffic lights have a centralized controller



Reflection

- How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?
 - Who did you keep in mind when making your design?
 - What was your goal with your design?
 - Did you have more than one goal?
- Did you reach the goal(s) with your design?

Once you understand how to handle distributed activity, it wasn't that difficult to link it with the design

We were always thinking about reliability, scaling, and maintaining the performance for the system; we focused mostly on the system itself and its NFPs rather than a particular audience.

Choosing REST style felt right because it was an architecture to allow for heterogeneous systems to interact with each other through a standard medium. The system could then easily be scaled and have an assurance of interoperability and allows for the ability to upgrade to high-end components / technology that would enable it for better performance and distribution. It impacts the system in a positive way.

We also included load balancing to avoid bottlenecking on one server; it also brings in the idea of redundancy which in turn ensures reliability

Team Name: Team 4

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Suppose you are tasked to design the software that will control the traffic lights in your city. The software will be connected to street cameras and will adapt the traffic lights according to the flow of traffic, in order to minimize congestion.

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

...using whatever method you like...

...using whatever architectural style / pattern we covered thus far in class...

...Cost is a concern...and this is the final design that you will hand off to your development team

Functional for 5 years

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Sense-Compute-Control for real-time part

Robust

List Roles and Student names

Moderator : Joshua

TimeKeeper: Chengjun Xi

NoteTaker: Warren

Reporter:

Reflection

- How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?

More difficult, since this design has some requirements on reliability

- Who did you keep in mind when making your design?

Traffic management staff, developers, maintenance staff

- What was your goal with your design?

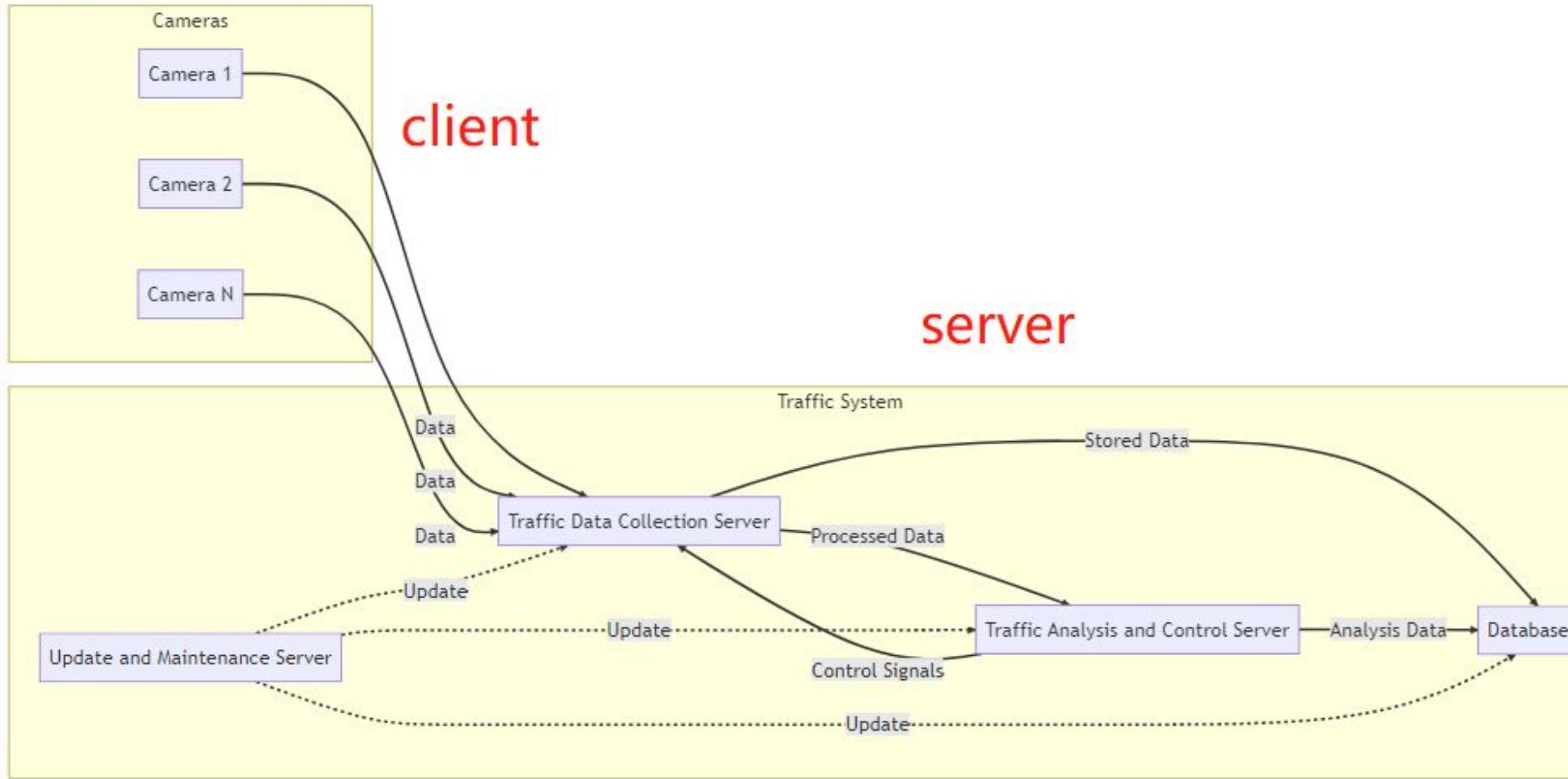
Reliable , long-longevity, and robust system that optimize

- Did you have more than one goal?

Yes

- Did you reach the goal(s) with your design?

yes



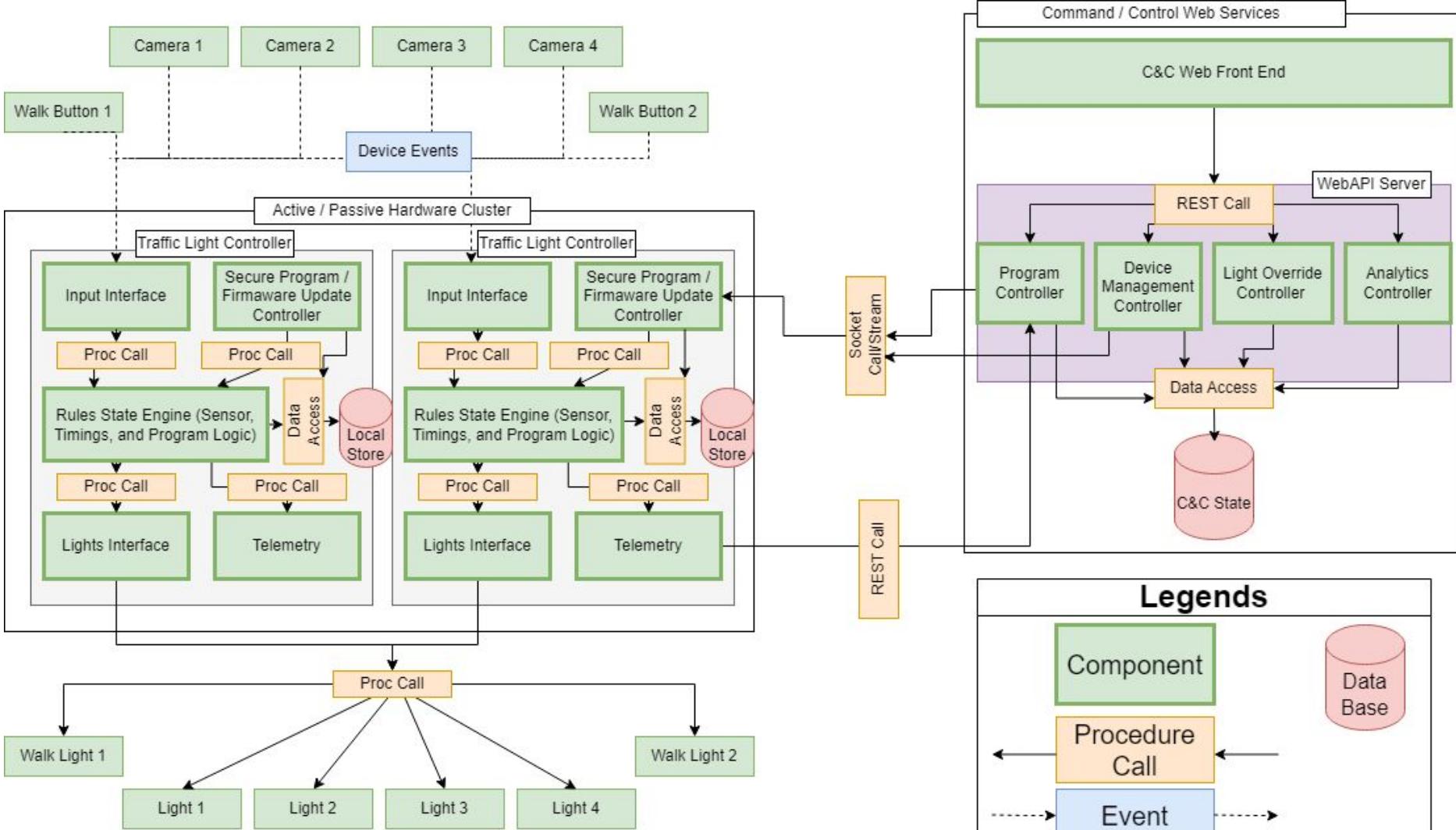
Team Name: Ludus

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

- How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?
 - Needed to combine many patterns to make a design
- Who did you keep in mind when making your design?
 - City government
 - Emergency Response teams
 - DOT stakeholders and pattern authors
 - City traffic users (Auto/Ped/Bike)
- What was your goal with your design?
 - Redundancy in case of embedded device breakage
 - Efficiency via cameras and sensors to properly control traffic
 - Ability to change pattern workflow/rules. IE: Camera/sensor + timer & program workflow + timer
 - Ability to remotely update the embedded devices
 - Ability for emergency services to override light patterns
 - Decentralized workflow/rules execution with central management and authoring
- Did you reach the goal(s) with your design?
 - Yes

List Roles and Student names

Moderator : TimeKeeper: NoteTaker: Reporter:



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Magratheans

- How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?

Much more difficult, as last exercise was very simple, and had simpler parameters

- Who did you keep in mind when making your design?

Traffic Policy Setters, Maintenance People, People who invent new kinds of each component

- What was your goal with your design?

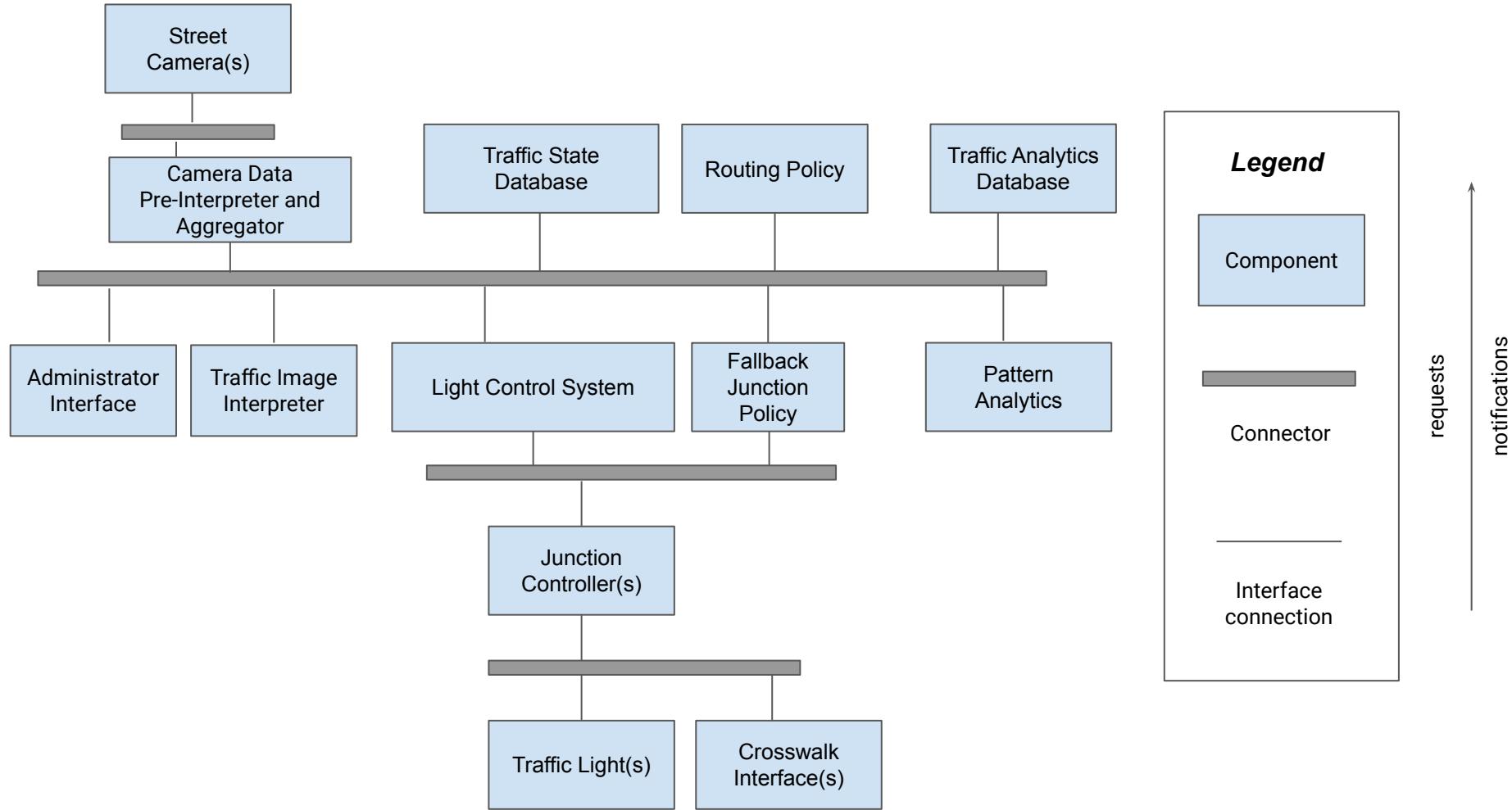
- 1) Meeting functional and nonfunctional requirements
- 2) Exploring C2

- Did you have more than one goal?

Yes.

Did you reach the goal(s) with your design?

Hopefully.



Team Name: Team 9

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Using C2 Architecture

Improve your design with the following new requirements:

- Must be robust
- Require minimal downtime for software fixes/patches
- Functional for at least 5 years

Create a design for this software...

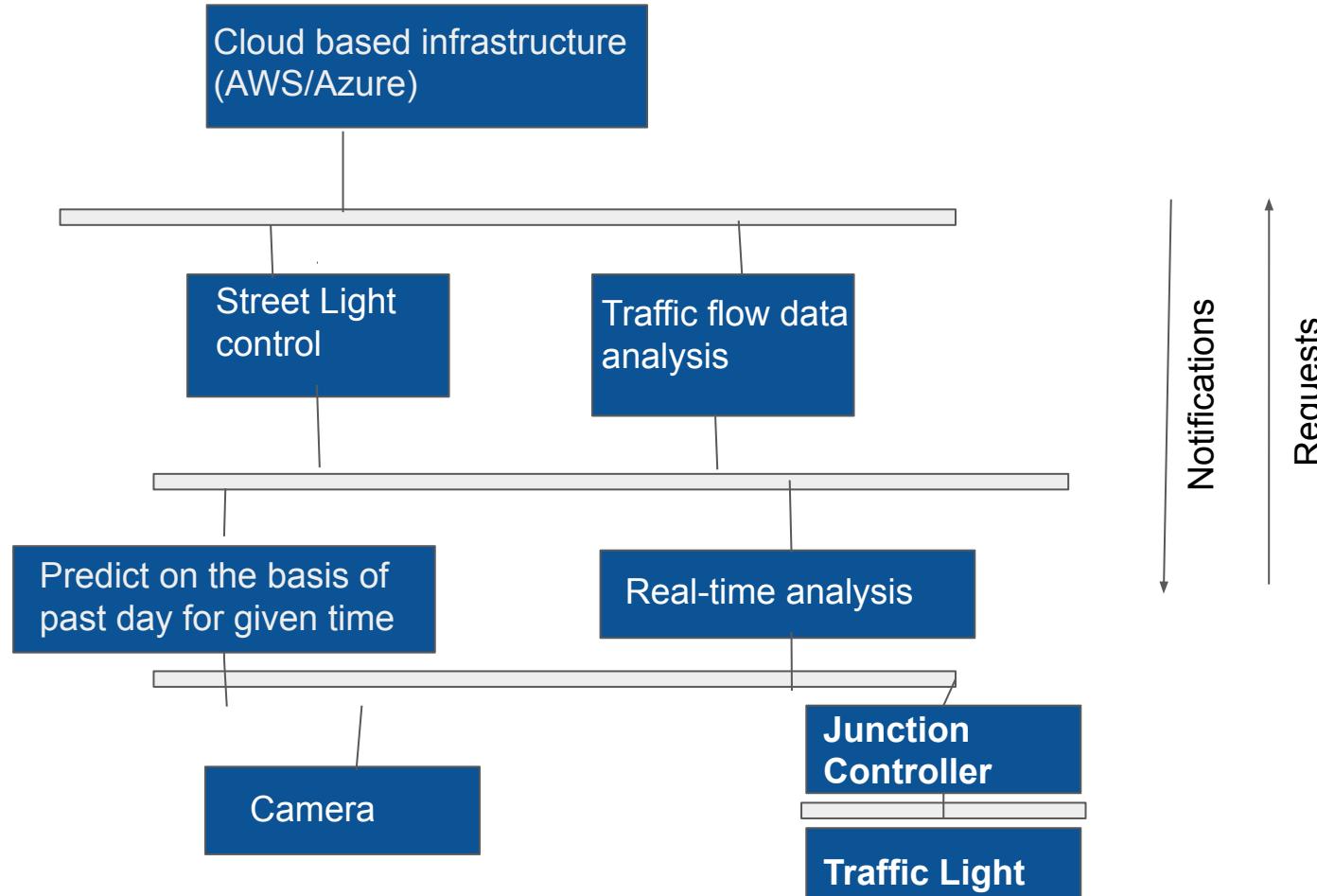
...using whatever method you like...

...using whatever architectural style / pattern we covered thus far in class...

...Cost is a concern...and this is the final design that you will hand off to your development team

• How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?

- This design was hard at first as we had to come up with the whole design and have constraints as well



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Open Source Web Services

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

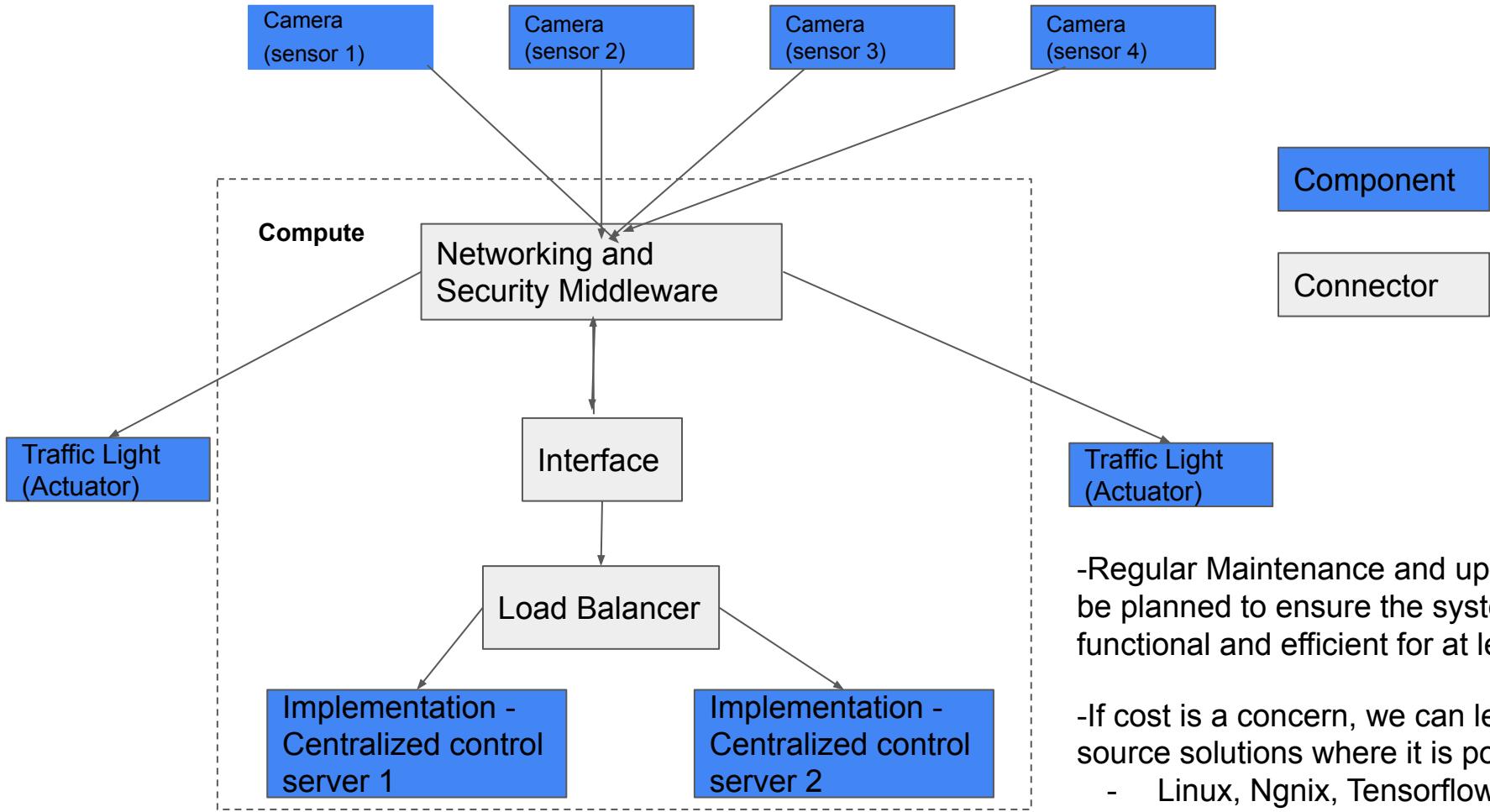
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:



-Regular Maintenance and updates
be planned to ensure the systems remain functional and efficient for at least 5 years.

-If cost is a concern, we can leverage open source solutions where it is possible:
- Linux, Nginx, Tensorflow, MySQL, PostgreSQL

Reflection

•How hard was it to think the design? How do you compare the level of difficulty with the previous exercise?

It was comparatively easy as we had this design for our previous in class activity. We believe C2 design was the most difficult in class activity.

•Who did you keep in mind when making your design?

Engineer who will be maintaining this application

•What was your goal with your design?

Make it robust, scalable, cheap.

•Did you have more than one goal?

Yes, we tried to meet all the requirements asked by the activity.

Did you reach the goal(s) with your design?

Yes, we did.

READ THIS FIRST - 15-20 minutes

Topic: Modern Systems

This activity is designed to

- Learn about design of modern systems
- Tie up loose ends regarding upcoming assignment(s)

Roles:

- **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.
- **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.
- **Note Taker** - takes record of the group’s discussion in the Google doc (below).
- **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity or the assignments due tonight, type in Red text

All teams will do the following task:

- Work in groups
- Look at the architecture of the following systems and determine what architectural styles they use
- Recreate these architectures with components and connectors labeled and post them to the google slides
- List the NFPs achieved by the architecture style

Use a Component-Connector notation. Do not specify the architecture style - class will determine

The urls below are simply starting points

1.Android application: http://elinux.org/Android_Architecture - **Dawn**

2.Amazon AWS: <https://aws.amazon.com/blogs/architecture/lets-architect-designing-well-architected-systems/> - **The Boffins**

3.MS Azure: <https://www.interviewbit.com/blog/azure-architecture/> - **To be decided**

4.Google search engine: <https://krazytech.com/technical-papers/how-does-google-search-engine-work> - **Team 4**

5.eBay: http://www.cs.mcgill.ca/~mahmed26/eBay_Architecture_Study.pdf - **Ludus**

6.Twitter: https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale - **The Magratheans**

7.YouTube: <http://highscalability.com/youtube-architecture> - **Team 9**

8.Linkedin :

https://www.slideshare.net/r39132/linkedin-data-infrastructure-qcon-london-2012/9-LinkedIn_Architecture_A_web_page **Open source web services**

Team Name: Dawn

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator : Chloe

TimeKeeper: Abdul

NoteTaker: Luo

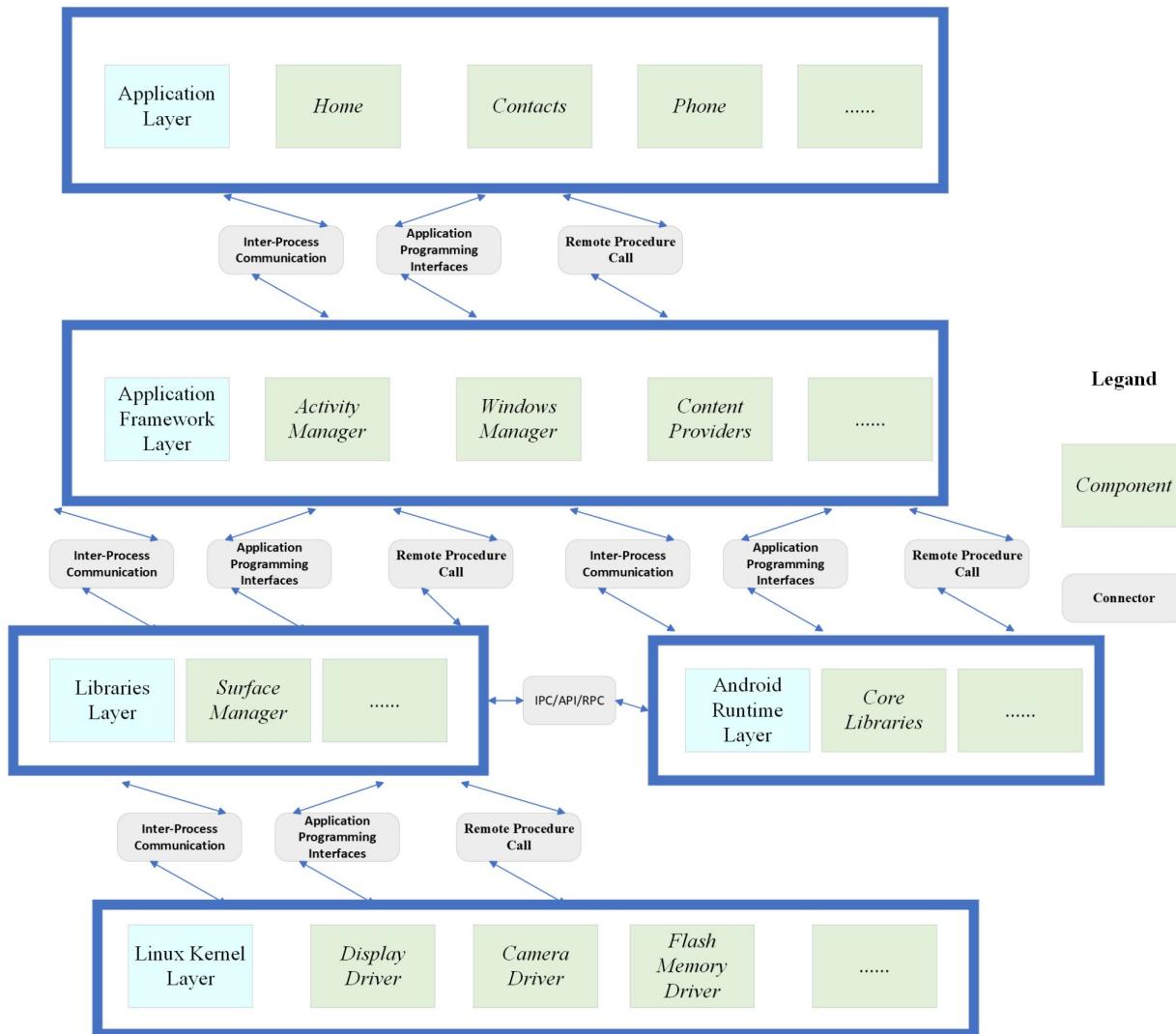
Reporter: Barack

Android architecture is divided into four main parts:

1. **Applications**: This is where all the applications like contacts, browser, games, etc. are located. These applications are written in Java programming language.
2. **Application Framework**: This layer provides higher-level services to applications in the form of Java classes. It includes managers that manage various types of data and services like resource manager, notification manager, activity manager, content providers, etc.
3. **Libraries and Android Runtime**: This layer is a set of C/C++ libraries used by various components of the Android system. It provides a wide range of libraries to perform different tasks. The Android Runtime includes the Dalvik Virtual Machine and Core Java libraries.
4. **Linux Kernel**: This is the foundation of the Android platform. It provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display, etc.

Question: List NFPs for architecture style

1. **Reliability:** The layered architecture style contributes to the reliability of the system. If one layer fails, the impact on other layers can be minimized.
2. **Scalability:** The component-based architecture of the application layer allows for easy scaling. New applications can be added without affecting existing ones.
3. **Reusability:** In the application layer, each application is developed as a separate component that can function independently. This allows for the reuse of applications across different devices or contexts without the need for significant changes.
4. **Security:** The Microkernel architecture of the Linux Kernel layer helps enhance security. The separation of user-space processes from kernel functionalities reduces the risk of system crashes and security breaches.
5. **Portability:** The use of the Java programming language for applications and most framework code, along with the Dalvik VM, contributes to the portability of Android. Applications can run on any device that has a Dalvik VM, regardless of the underlying hardware.
6. **Maintainability:** The clear separation of concerns in the Layered architecture and the Component-based architecture simplifies maintenance. Issues can be isolated to specific layers or components within that layer, making it easier to identify and fix problems.
7. **Performance:** The use of a Virtual Machine (VM) allows for efficient use of system resources, which can enhance performance. Also, the Native Libraries, Daemons, and Services layer, written in C/C++, can offer high performance for lower-level functionalities.



Team Name: The Boffins

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

NFPs:

- Scalability: Elastic Load Balancing allows requests to scale to large volumes
- Security: Each request passes through an EC2-driven security group and outputs all logs to an S3 bucket for auditing purposes
- Availability: Customers can leverage CloudFront distributions to reach wide availability zone coverages
- Performance: AWS utilizes Elastic Block Storage (EBS) to process data efficiently

List Roles and Student names

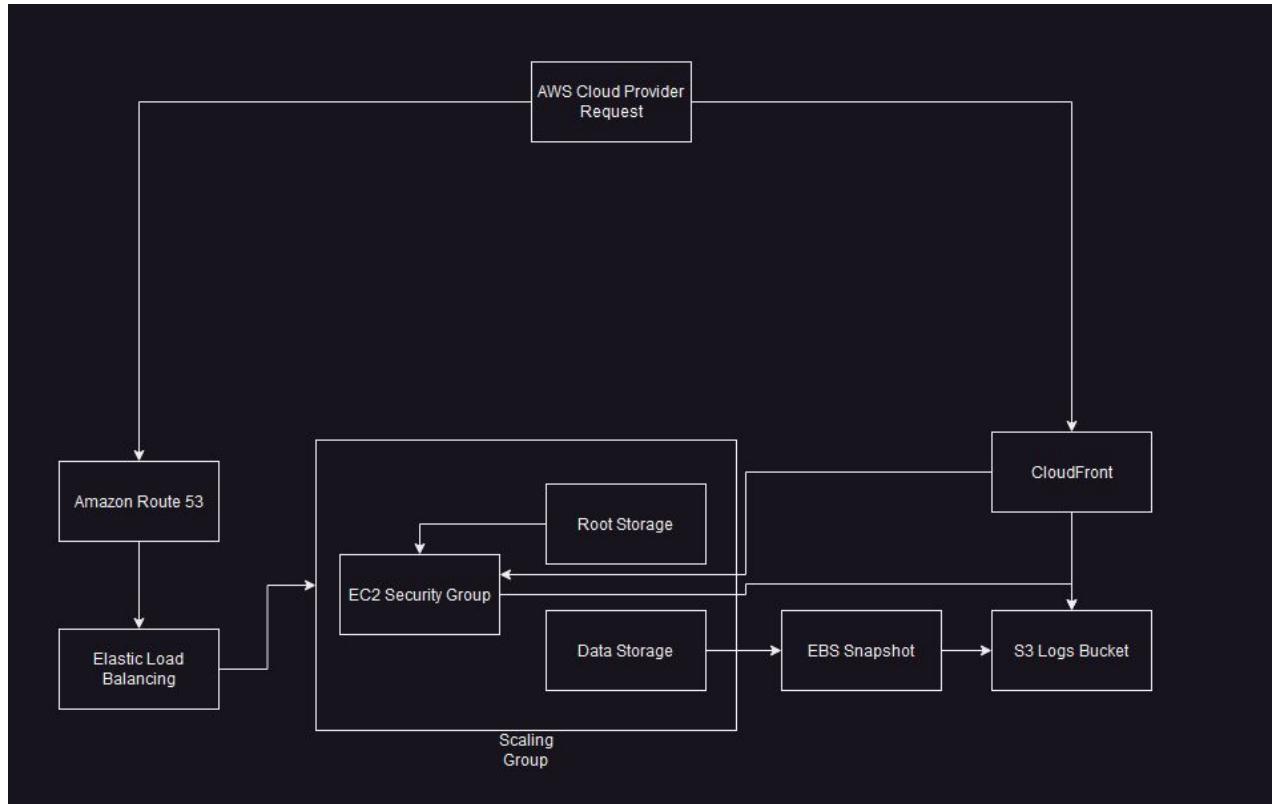
Moderator :

TimeKeeper:

NoteTaker:

Reporter:

AWS Cloud Provider Architecture Diagram



Team Name: To be decided

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

“...is a major technique for reducing the amount of physical hardware required for a server farm”

“Each application is discrete, but the subsequent sources can assist”

everything is automated

each rack has a bunch of different servers

control portal (manage stuff)

Azure Resource Manager (talks to the service provider) - similar to GFS

portal talks to ARM, the expert / controller is the portal that talks to all of the other services

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

“...is a major technique for reducing the amount of physical hardware required for a server farm”

“Each application is discrete, but the subsequent sources can assist”

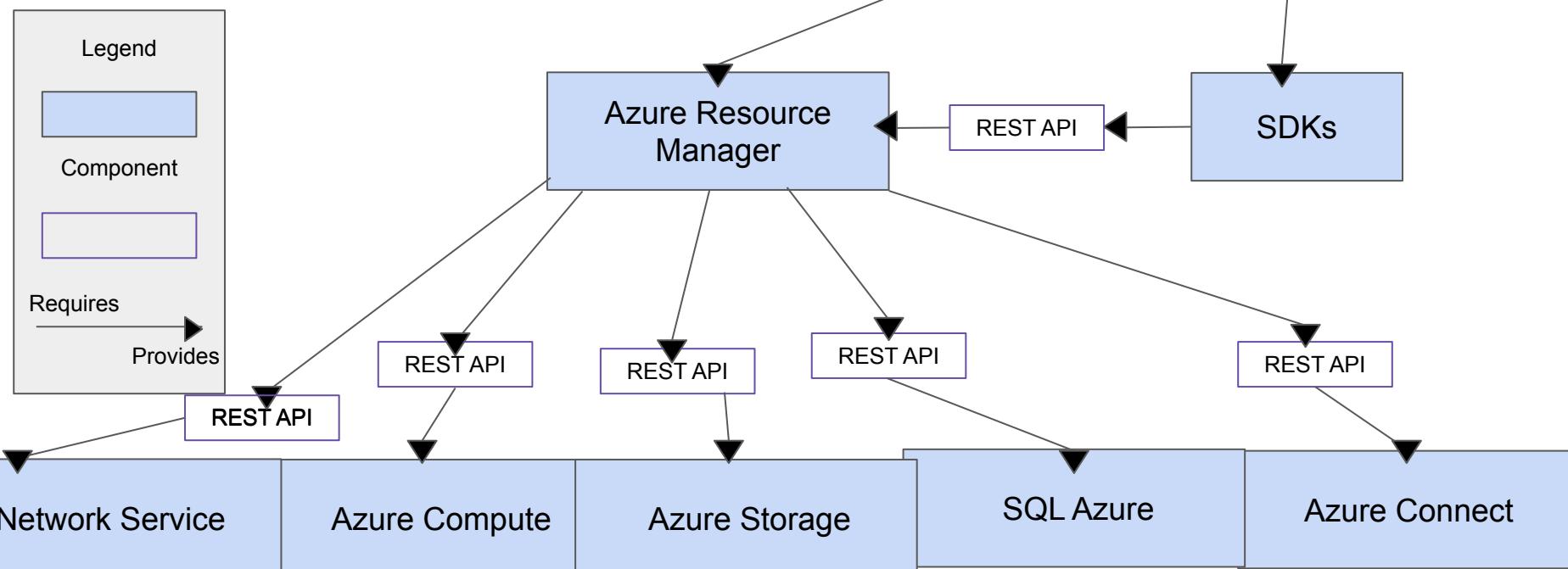
everything is automated

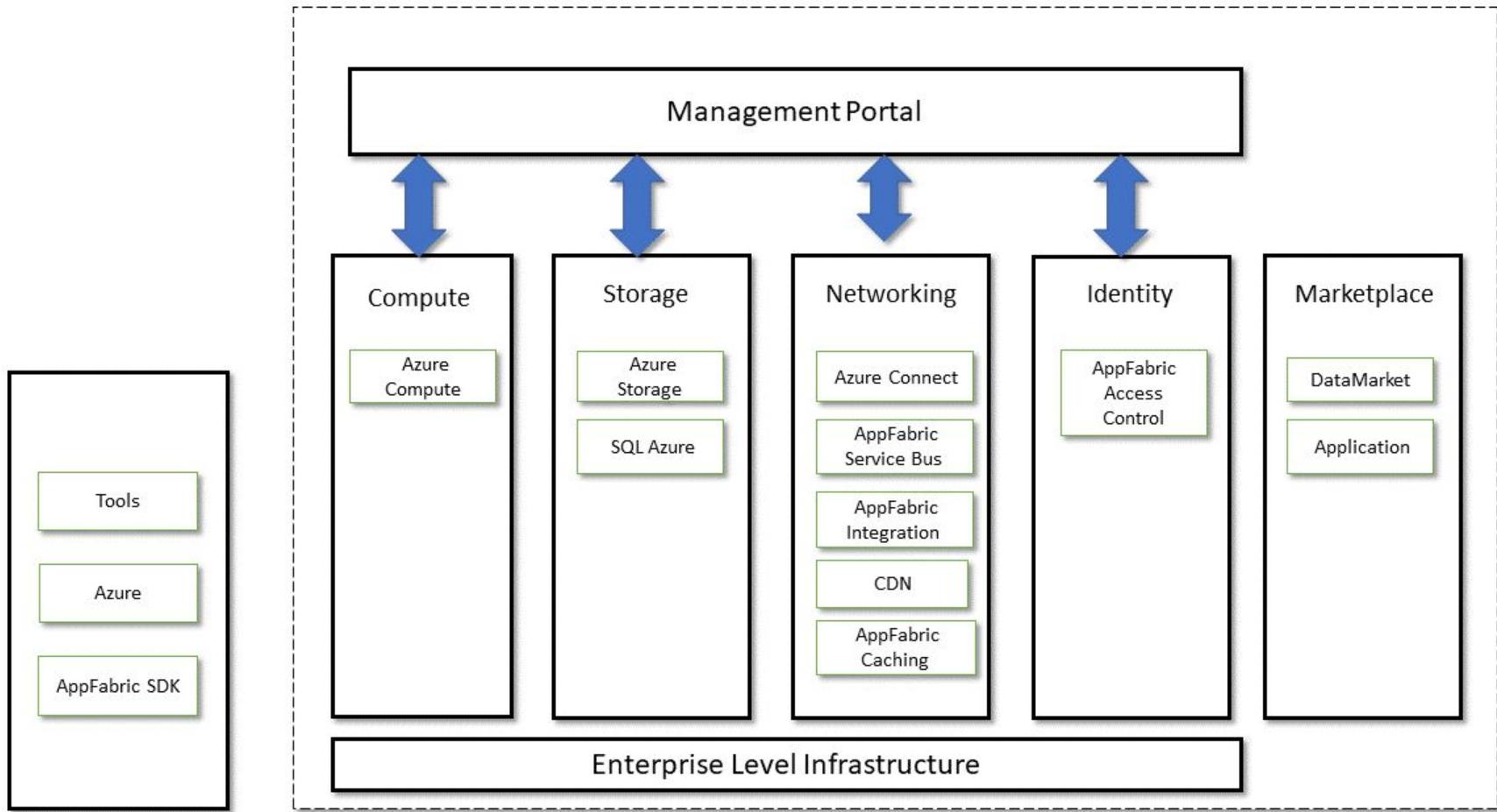
each rack has a bunch of different servers

control portal (manage stuff)

Azure Resource Manager (talks to the service provider) - similar to GFS

portal talks to ARM, the expert / controller is the portal that talks to all of the other services





Team Name: Team 4

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

4. Google search engine: <https://krazytech.com/technical-papers/how-does-google-search-engine-work> - Team 4

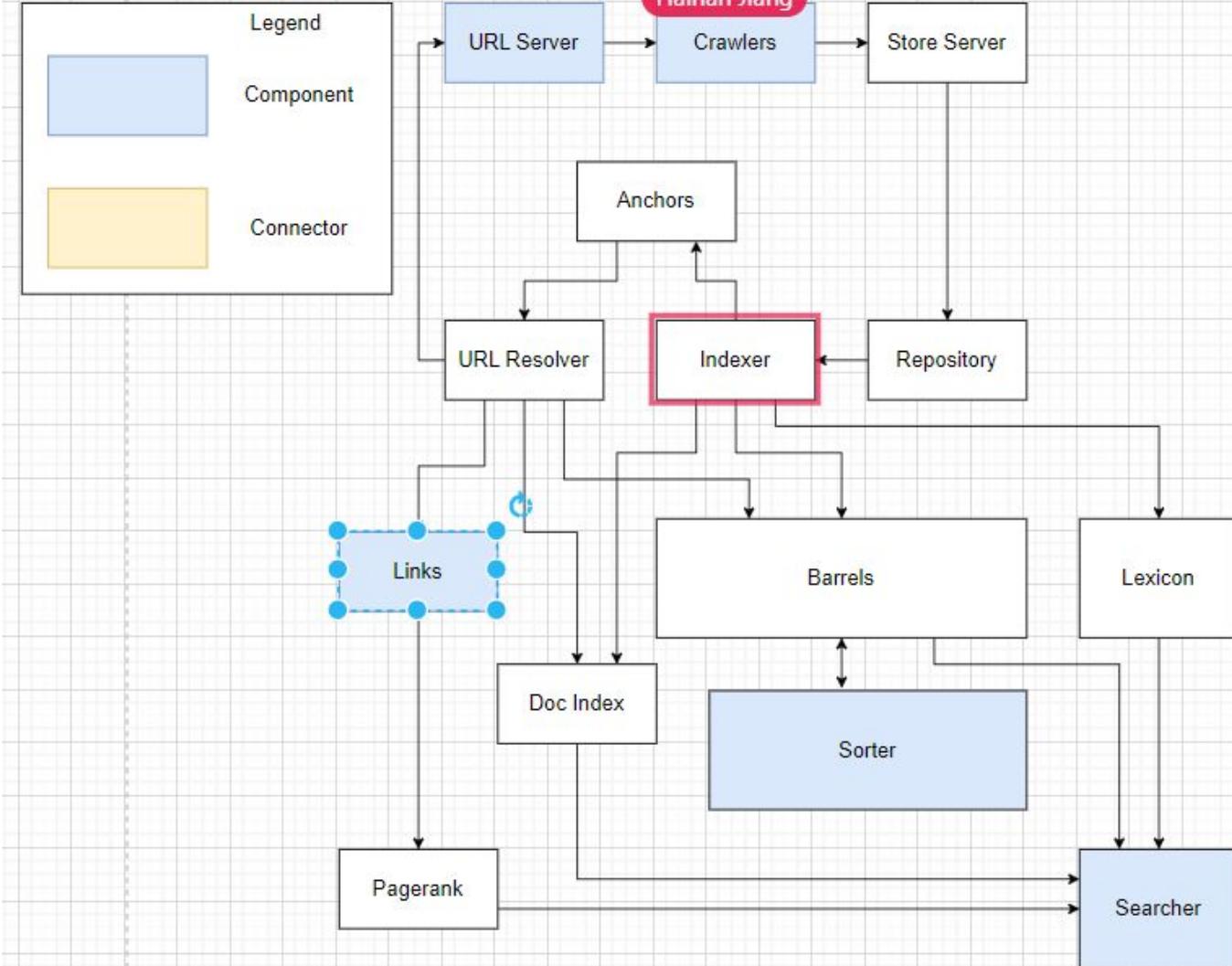
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:



Team Name: Ludus

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

eBay: http://www.cs.mcgill.ca/~mahmed26/eBay_Architecture_Study.pdf - *Ludus*

- Work in groups
- Look at the architecture of the following systems and determine what architectural styles they use
 - 3 tiered client server
- Recreate these architectures with components and connectors labeled and post them to the google slides
- List the NFPs achieved by the architecture style
 - Availability/reliability
 - Security
 - Quality
 - Modifiability
 - High performance
 - Scalability

Legends

Component

Conector

Data Acess

Data Base

Client

HTTPs Request

DNS Load Balancer

Proxy Server

Load Balancer

HTTPs Request

Web Servers

RPC / REST

Application
Servers

Data Acess

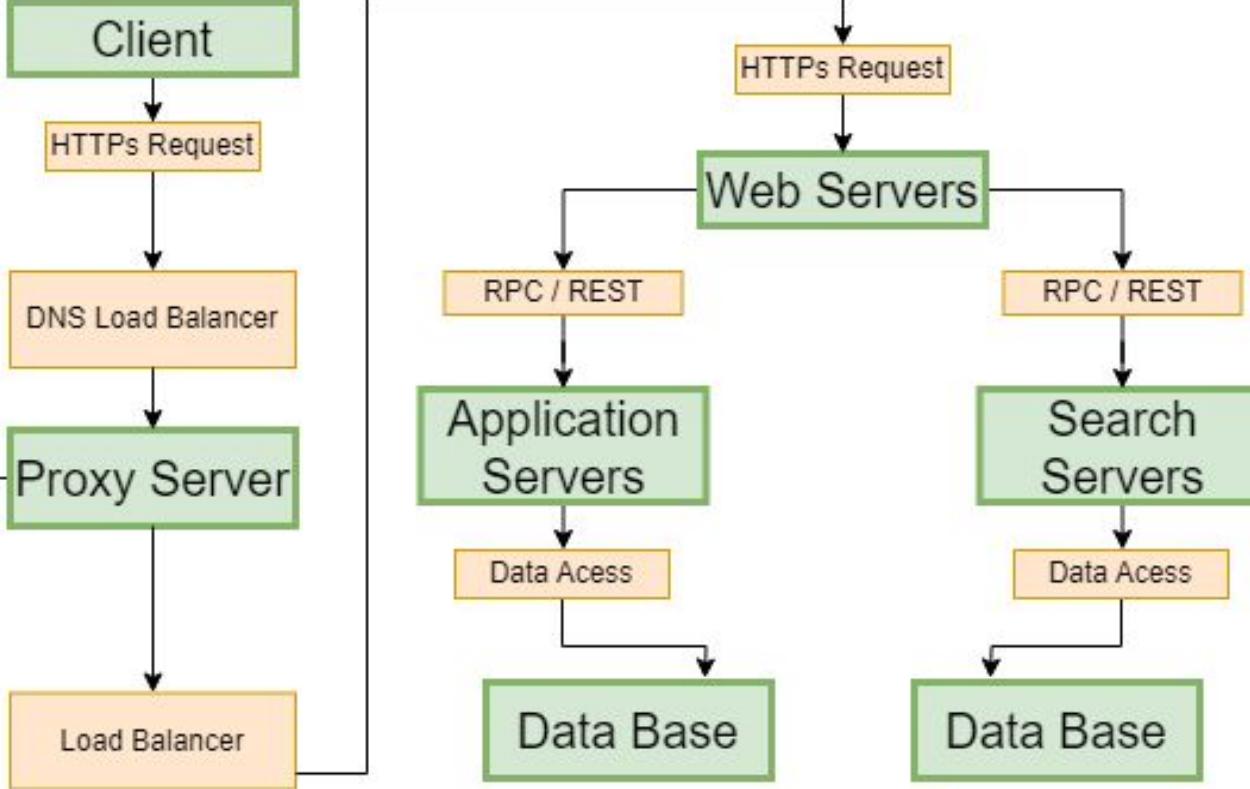
Data Base

RPC / REST

Search
Servers

Data Acess

Data Base



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: MAGRATHEANS

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

- Look at the architecture of the following system and determine what architectural styles they use
- Recreate these architectures with components and connectors labeled and post them to the google slides
- List the Non Functional Properties achieved by the architecture style
 - Scalability
 - Reliability
 - Availability
 - Speed
 - Security
 - Efficiency

https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale

Legend

Component

Connector



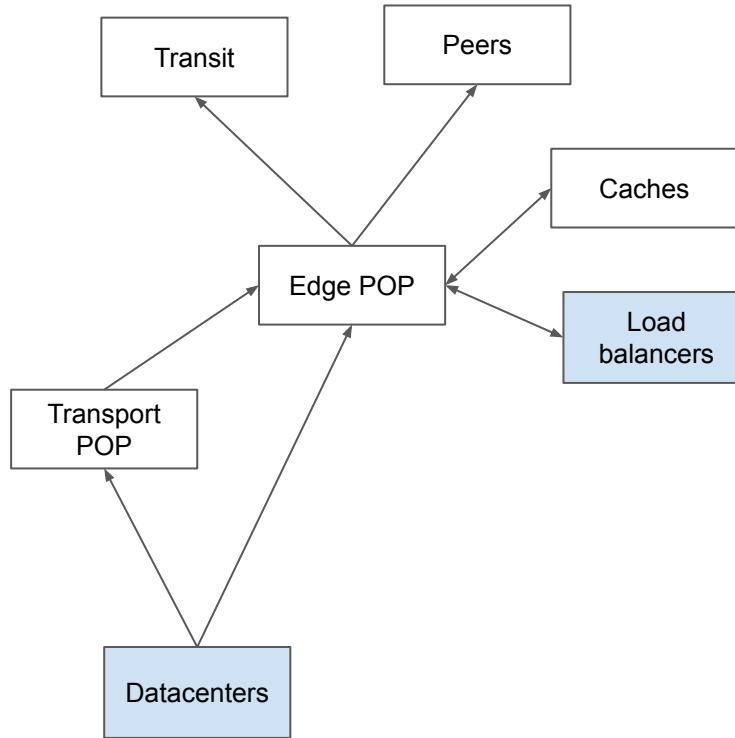
Interface connection



Requires



Provides



Team Name: Team 9

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

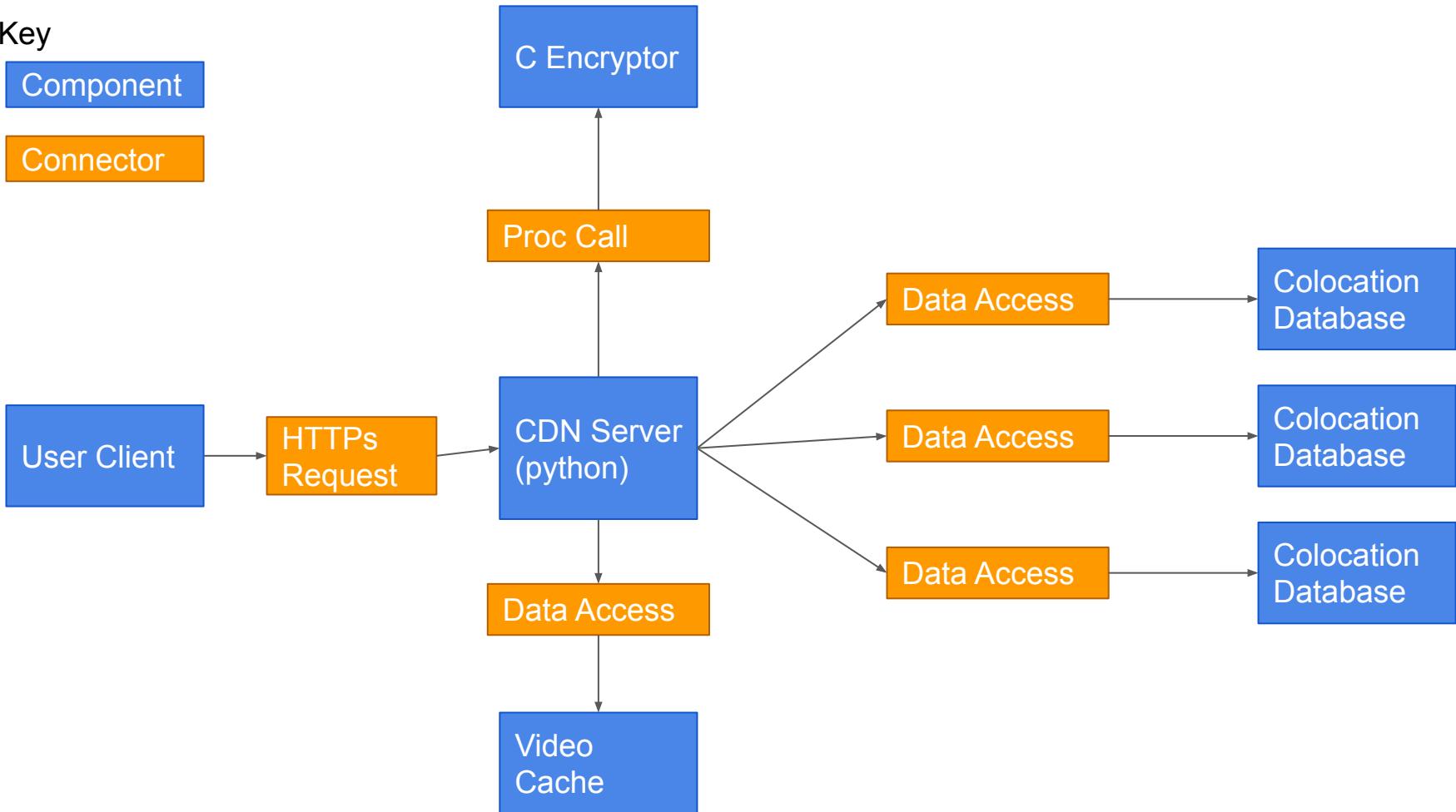
NoteTaker:

Reporter:

Key

Component

Connector



Team Name: Open Source Web Services

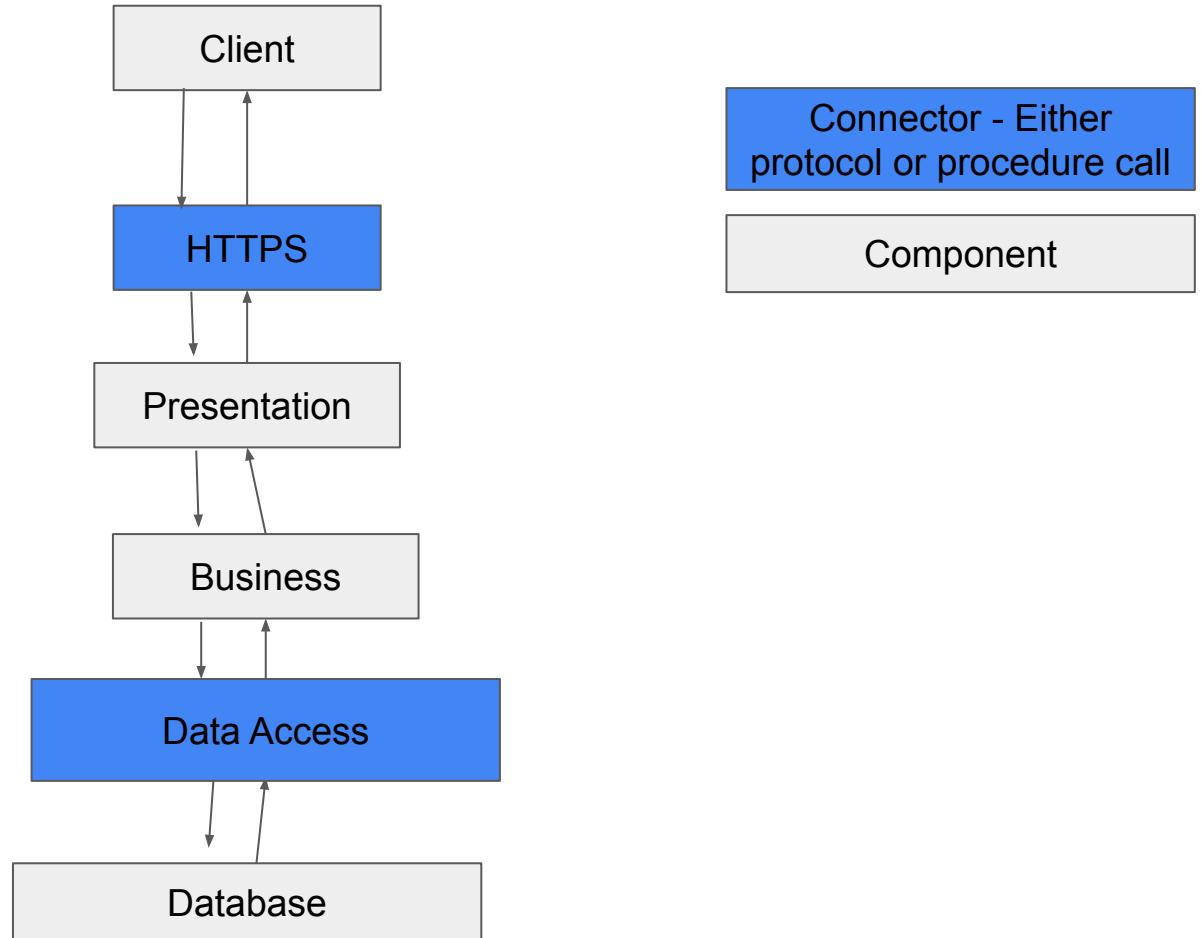
Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

- Look at the architecture of the following systems and determine what architectural styles they use
- Layered Architecture style**
- Recreate these architectures with components and connectors labeled and post them to the google slides
 - List the NFPs achieved by the architecture style}
 - **Maintainability**
 - **Modularity**
 - **Portability**
 - **Reusability**
 - **Interoperability**
 - **Scalability**
 - **Security**





Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Agenda



- Review
- Review Questions
- G4 questions
- -----
- Last week's activity
- Activity this week

Review



- Networked systems
- P2P systems
 - Napster vs Gnuttela vs Skype vs BitTorrent
 - Napster – Content Directory Server – hybrid CS with P2P, single point of failure
 - Gnuttela – Pure P2P, discovery can take a while
 - Skype “original” – super nodes – “distributed Content directory” – hybrid CS with P2P
 - BitTorrent – also solves problem of flashcrowds – cut a file into multiple parts and distributed across different peers, requires peers who got the info to also distribute to other peers
- Web services

Questions



- What architecture style (or combined styles) is used to address the issue of flash crowds? Explain how this style addresses this challenge. **Dawn**
 - Flash crowds? User traffic increases a lot – failed to provide service
 - Client-Server (Edge servers) – requests go to edge servers
 - Edge server – replicate main server content
 - Akamai – one specific example
 - Another way is to use P2P – BitTorrent – divide file into many pieces – stored among different peers
- Identify **one** issue with pure P2P systems. How can this issue be addressed? – **the Boffins**
 - Malicious clients – use authentication server – to authenticate the peers

Questions



- What is a difference between distributed and decentralized computing?
What architecture style(s) support decentralized computing and how - **To be decided, Team 4**
 - Distributed computing – distribute among different locations (act as one unit, centralized control)
 - Decentralized – no centralized control – each system act independently, but cooperate
 - P2P – decentralized computing – states & behavior among peers – can act as client or servers – no centralized control
 - Mobile code – distributed computing – send code to worker machines
- How to handle scalability? Give one specific architecture style. **Ludus, Open Source Web services**
 - GFS – modified CS – client talks to a main manager and routes to chunk server, rather than contact only server (analogy engineer asks a manager who is the “expert”, manager will direct engineer to the expert)
 - Micro-services architecture – independent applications that only do one thing. More will be posted about this architecture. Compare and contrast with web services
- How is SOAP different from REST? **The Magratheans**
- Can you use JSON with SOAP? **Team 9**

Questions



- How is SOAP different from REST? **The Magratheans**
 - REST – resource is a key concept, context free interaction, user agents as components, connector (HTTP, libwww)
 - SOAP – distributes services by exchange XML docs – data is services as XML docs, components are ser4vices, connector, connectors (protocols, asynchronous, stream + distributed)
- Can you use JSON with SOAP? **Team 9**
 - Yes, technically can, embed JSON in SOAP
 - JSON – key – value pair – as plain text which describes data
 - Can also adapter between SOAP & JSON

ACTIVITIES

Previous activity



- 10 minutes to revisit reflection questions and prepare to present

Activity – 15 mins



- Work in groups
- Look at the architecture of the following systems and determine what architectural styles they use
- Recreate these architectures with components and connectors labeled and post them to the google slides
- List the NFPs achieved by the architecture style

Activity



The urls below are simply starting points

1. iPhone application: http://elinux.org/Android_Architecture - Dawn
2. Amazon AWS: <https://aws.amazon.com/blogs/architecture/lets-architect-designing-well-architected-systems/> - The Boffins
3. MS Azure: <https://www.interviewbit.com/blog/azure-architecture/> - To be decided
4. Google search engine: <https://krazytech.com/technical-papers/how-does-google-search-engine-work> - Team 4
5. eBay: http://www.cs.mcgill.ca/~mahmed26/eBay_Architecture_Study.pdf - Ludus
6. Twitter:
https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale - The Magratheans
7. YouTube: <http://highscalability.com/youtube-architecture> - Team 9
8. Linked in : https://www.slideshare.net/r39132/linkedin-data-infrastructure-qcon-london-2012/9-LinkedIn_Architecture_A_web_page Open source web services
9. Instagram: <https://scaleyourapp.com/instagram-architecture-how-does-it-store-search-billions-of-images/> -
SQL Server: <https://www.guru99.com/sql-server-architecture.html>



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023



ARCHITECTURES FROM SPECIFIC DOMAINS

Robotics



- Uses
 - Space/underwater exploration
 - Hazardous waste disposal
 - Industrial automation system
- Issues
 - What are some issues with robots?

Robotics



- Current day robotic technologies:
<https://www.youtube.com/watch?v=Jky9I1ihAkg>



Robotics



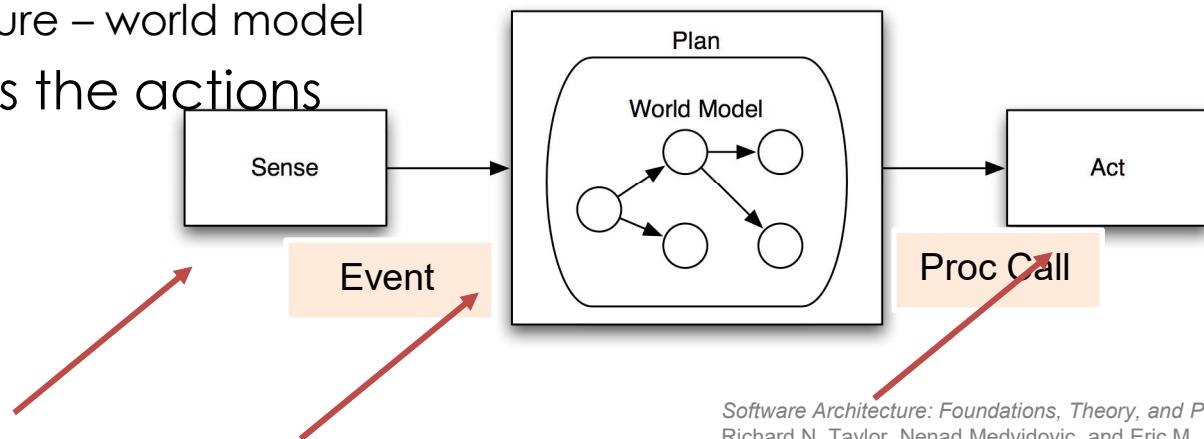
- Uses
 - Space/underwater exploration
 - Hazardous waste disposal
 - Industrial automation system
- Issues
 - Interface with external sensors & actuators
 - Real-time response to stimuli
 - Response to obstacles
 - Sensor input fidelity
 - Power failures
 - Mechanical limitations
 - Unpredictable events



Robotics: Sense-Plan-Act



- Architecture that uses continuous environment feedback as an explicit input to planning actions
- Components
 - Sense – gather sensor information from the environment
 - Plan – sensor data + robot's state + task → determine what to do next
 - Subarchitecture – world model
 - Act – performs the actions
- Connectors?



Software Architecture: Foundations, Theory, and Practice;
Richard N. Taylor, Nenad Medvidovic, and Eric M.
Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with
permission.

Robotics: Sense-Plan-Act (SPA)

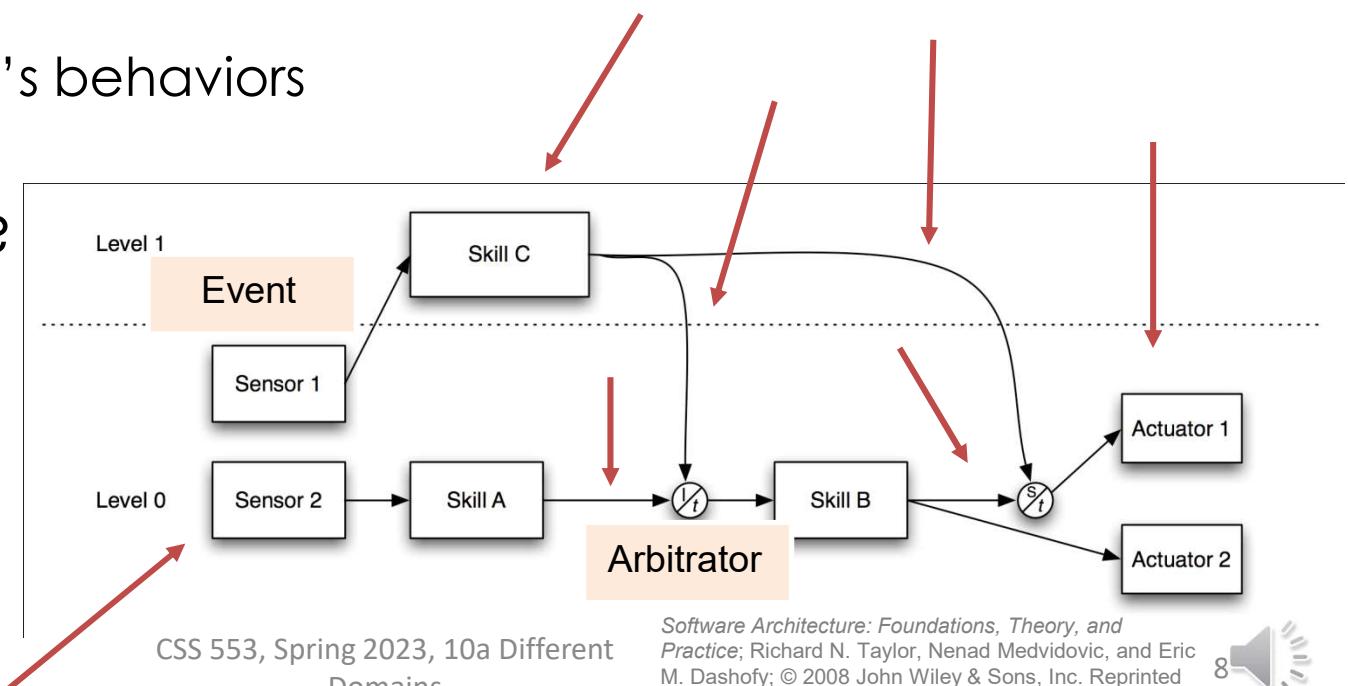


- Similar architectural styles?
 - Sense-compute-control
 - however, this does not maintain a model or perform sophisticated planning
 - Pipe and filter
 - iterative unidirectional flow of data
 - Difference is the sub-architecture in the plan component
- Limitations
 - Performance
 - Prior to each action, time-consuming computations performed
 - Scalability
 - Does not scale well as the robotic system capabilities and goals expand

Robotics: Subsumption Architecture



- Architecture that modularizes a robot's behavior and does not use a central planning
 - Allows for quick responses to stimuli
- Components?
 - Sensors
 - Skills – robot's behaviors
 - Actuators
- Connectors?



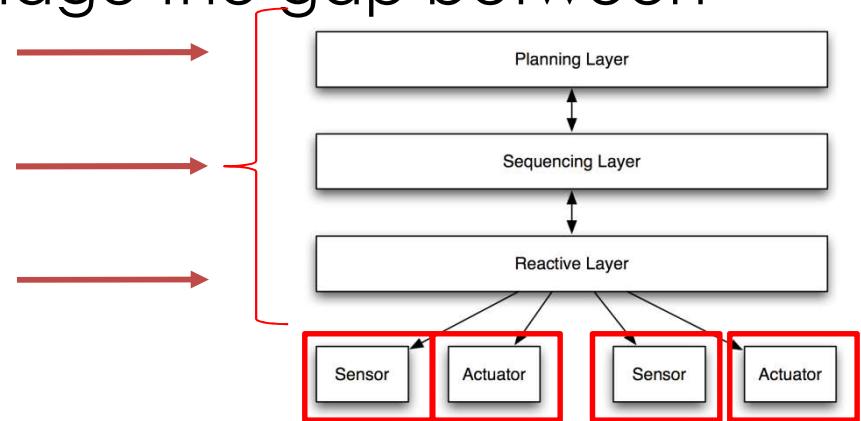
Robotics: Subsumption Architecture



- Overall behavior
 - Based on the topology, or the organization of the components
 - Component-based approach to the data flow between sensors and actuators
- Limitations?
 - Lack of coherent plan
 - No guidance for layering the architecture

Robotics: Three-Layer

- Architecture that aims to bridge the gap between SPA and Subsumption
- Components?
 - Sensors
 - Actuators
 - Those that reside in the layers
 - Planning - performs the slower and long term planning
 - Sequencing – responsible for linking functionalities present in the reactive layer for more complex behaviors; translate high level directions to lower level actions
 - Reactive – quickly responds to events in the environment
- Connectors?
 - Most likely procedure calls



Software Architecture: Foundations, Theory, and Practice; Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy; © 2008 John Wiley & Sons, Inc. Reprinted with permission.

Robotics: Three-Layer



- Similar to which architecture?
 - Layered virtual machine – difference is that layers can call upper and lower layers, instead of simply calling lower layers
- Challenges?
 - How to separate a functionality into three layers?
 - Tendency for one layer to dominate others in importance and complexity
 - Overhead in maintaining and reconciling different world and behavior models between the planning and the reactive layers

Wireless Sensor Networks (WSN)



- Monitor on the environment and reports on its state
- Application domains:
 - medical systems, navigation, industrial automation, and civil engineering
- May be used in the Internet of Things (IoT)
- Videoclip – watch the first minute
 - <https://www.youtube.com/watch?v=W1aMmCZ25fw>

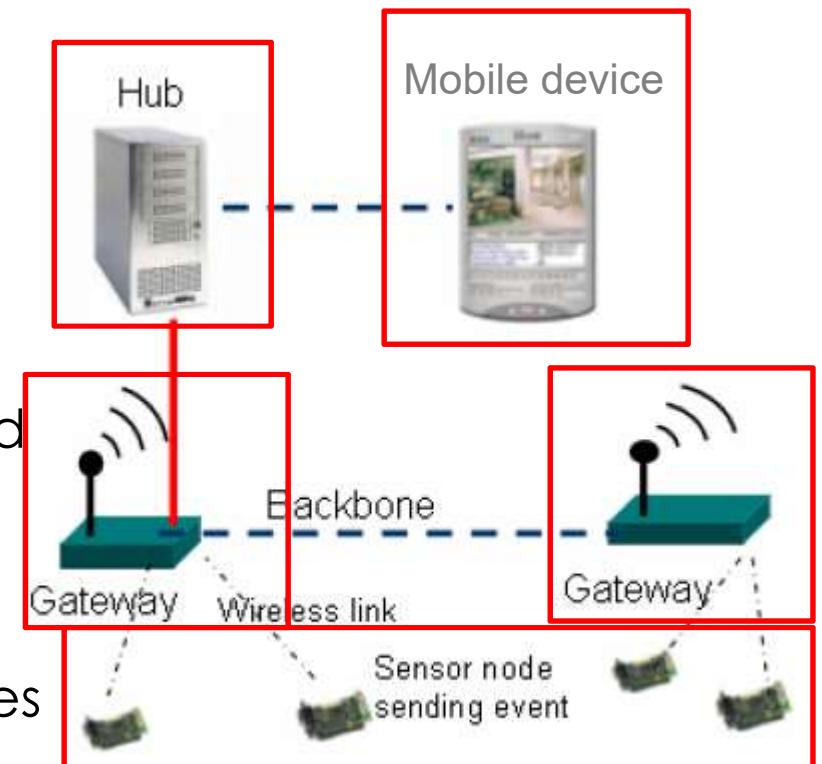
Wireless Sensor Networks (WSN)



- Benefits:
 - Low installation cost, inexpensive maintenance
- Challenges?
 - Integration with legacy wired networks, embedded devices, mobile networks
 - Highly constrained: power, bandwidth, processing capacity
 - Need scalability, fault-tolerance, performance (response time), availability, dependability

WSN: Bosch's MIDAS

- MIDAS – WSN-based family of embedded applications, consisting of
 - Sensors – monitor environment
 - Gateways – manage and coordinate the sensors; collect data from the sensors and forward to the hub
 - Hubs – evaluate and visualize sensor data for people; may propagate data to Mobile Devices
 - Mobile Devices – view sensor data



Adapted from Malek, S., Seo, C., et. al.
Reconceptualizing a Family of heterogeneous Embedded Systems via Explicit Architectural Support , ICSE 2007

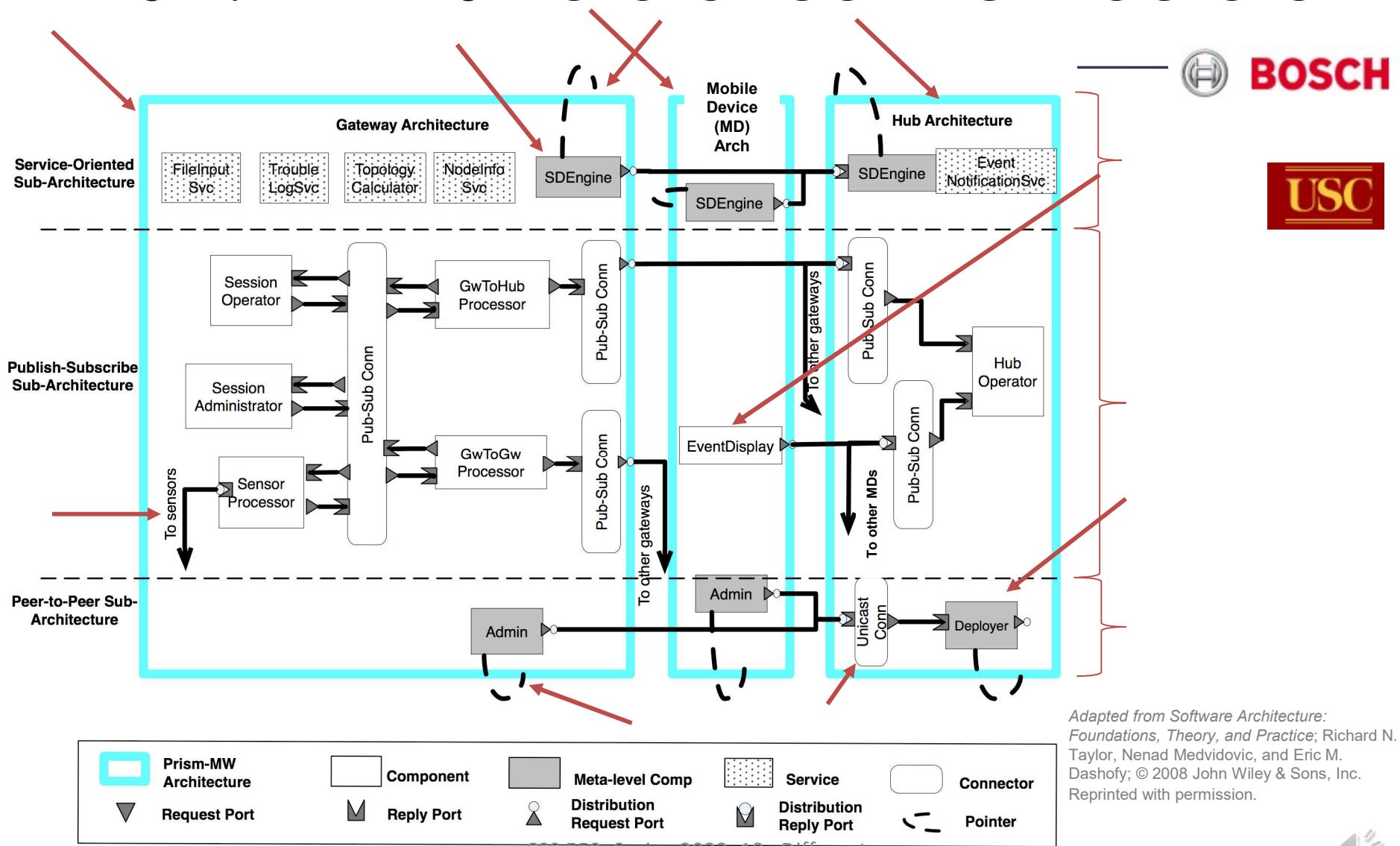
WSN: Bosch's MIDAS



- MIDAS
 - Needs a generic and flexible middleware platform
- NFPs
 - Resource-constrained
 - Performance
 - Scalability
 - Heterogeneity
 - Fault-tolerance

Malek, S., Seo, C., et. al. Reconceptualizing a Family of heterogeneous Embedded Systems via Explicit Architectural Support , ICSE 2007

WSN: MIDAS Reference Architecture



Adapted from Software Architecture:
Foundations, Theory, and Practice; Richard N.
Taylor, Nenad Medvidovic, and Eric M.
Dashofy; © 2008 John Wiley & Sons, Inc.
Reprinted with permission.

Takeaways



- A great architecture is the ticket to runaway success
- A great architecture reflects deep understanding of the problem domain
- A great architecture probably combines aspects of several simpler architectures
- Develop a new architectural style with great care and caution. Most likely you don't need a new style.

Questions



- With the availability powerful hardware and off-the-shelf artificial intelligence software, how do these change the discussed architecture styles in the field of Robotics?
 - Same architecture, faster, perhaps more accurate – AI technologies
- You saw an example of using multiple architecture styles in IoT (MIDAS System). Find a modern-day IoT system and identify at least two architecture styles present in the system.
 - IoT – different appliances, gadgets being connected through some network
- Pick one domain (aside from Robotics and IoT). Identify key requirements in this domain and create an ideal architecture style (or hybrid of styles) to satisfy these requirements.
- Pick **one** question from Chapter 11 (11.8) in the book and answer that question.

Clarifications



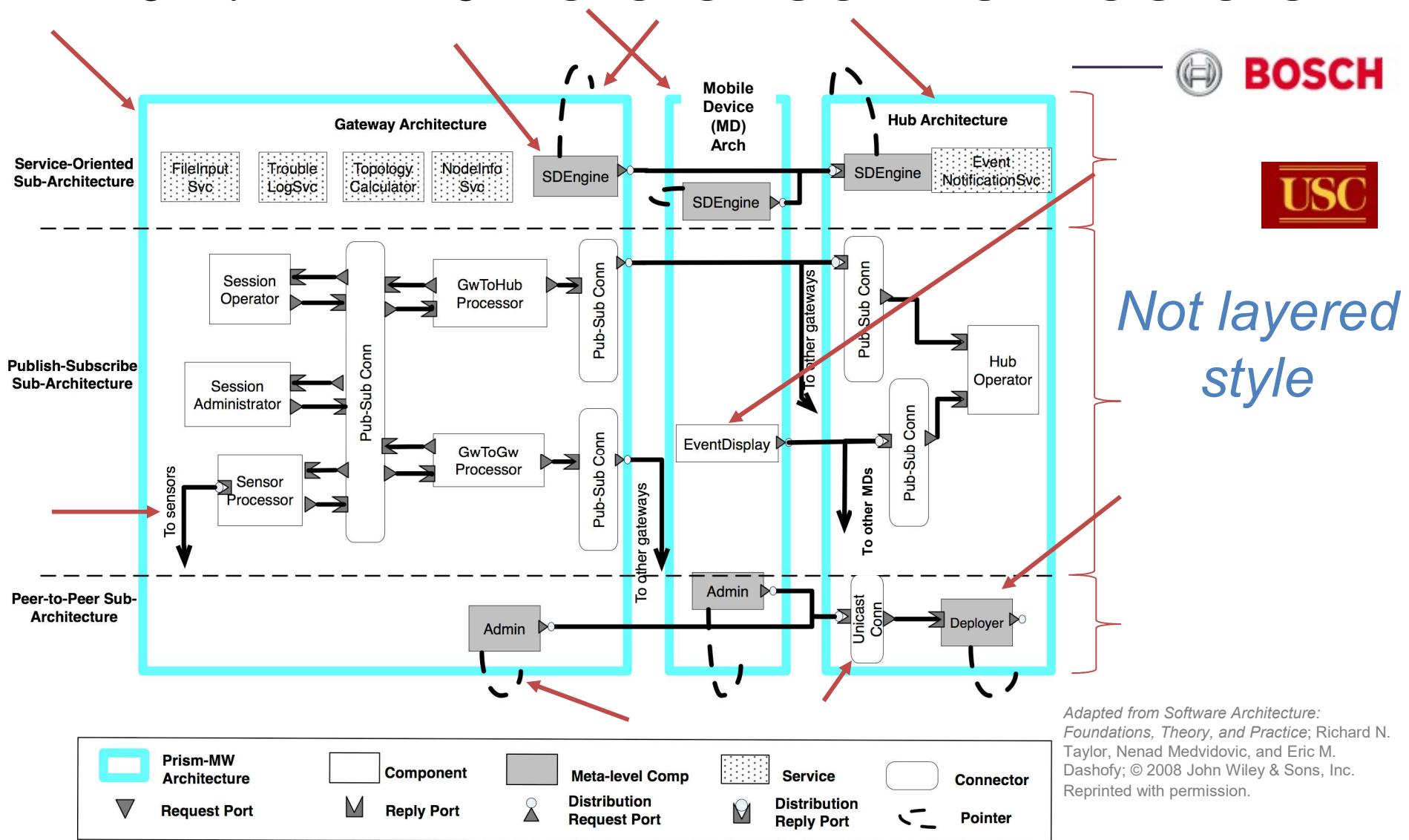
- Direction of arrows
- Component-connector notation
- Legend

Practice designing



- Identify functional and non-functional requirements for the system listed in next slide
- Based on these requirements select your architecture style and draw your diagram
- Compare your style with the architecture style provided in the links

WSN: MIDAS Reference Architecture



Adapted from Software Architecture:
Foundations, Theory, and Practice; Richard N.
Taylor, Nenad Medvidovic, and Eric M.
Dashofy; © 2008 John Wiley & Sons, Inc.
Reprinted with permission.

COURSE OVERVIEW/WRAP-UP

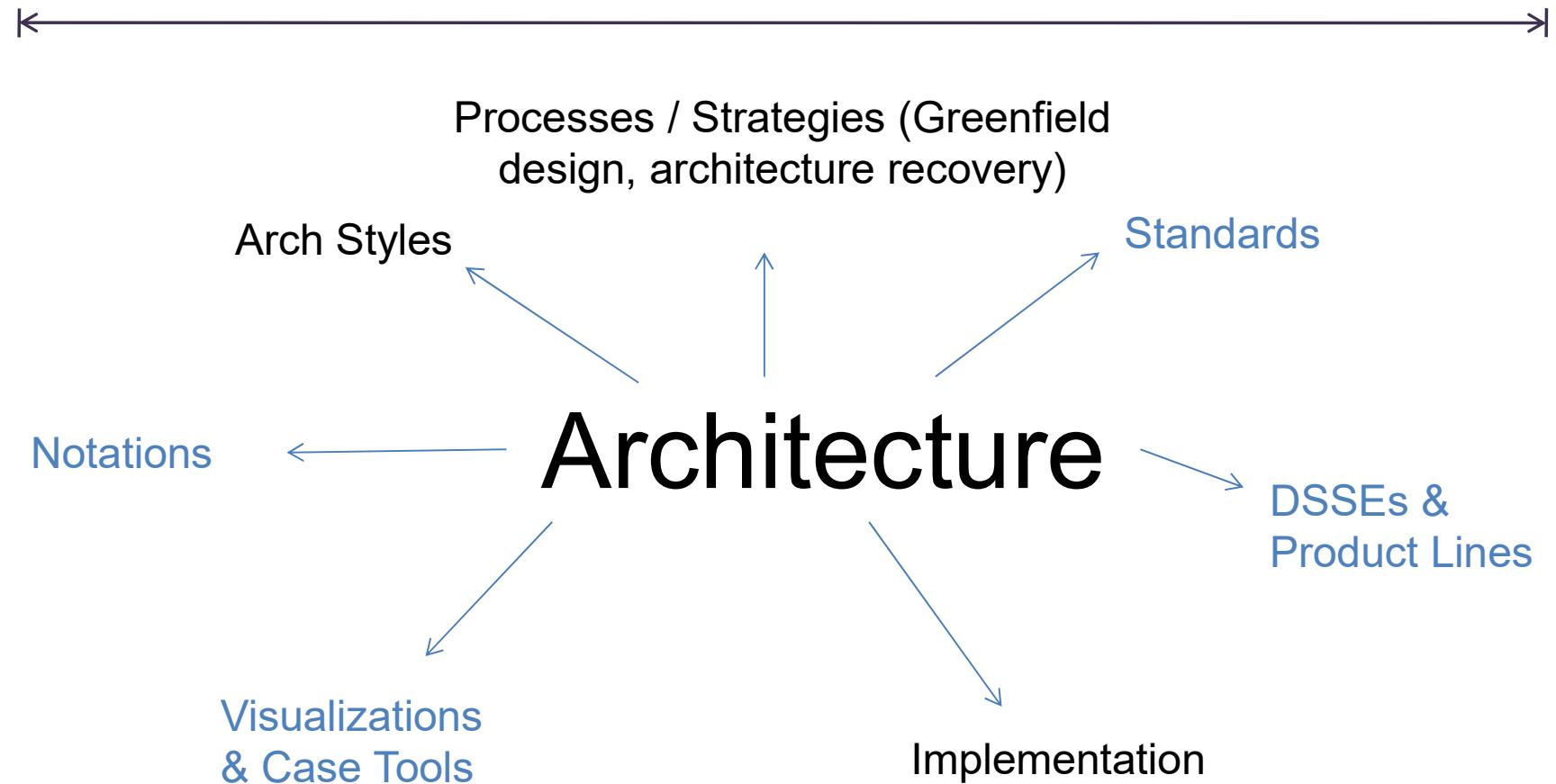
Running Threads in the class



- Software is complex
- Learn from experience

Architecture

Running Threads in the class



Course objectives



- Name and describe software architecture concepts, methods, and best practices
- Select and justify the use of software architecture styles or patterns for a given situation
- Apply software architecture styles or patterns on a small but realistic project
- Use CASE tools in creating, visualizing, and analyzing software architecture artifacts



Discussion points



Is Agile and Architecture an oxymoron?

Discussion points



What arch style to use if...

- You wish to allow multiple organizations to control own resources
- Distributed

Good sources of readings



- Software Architecture conferences
 - <https://icsa-conferences.org/series/history/>
- Software Engineering conferences
 - <http://www.icse-conferences.org/>
 - <https://www.esec-fse.org/>
 - <http://www.ase-conferences.org/>
- Websites
 - <http://www.softwarearchitectureportal.org/>

Good sources of readings



- Library resources – full articles
 - <https://guides.lib.uw.edu/bothell/css>
 - From off-campus, login
 - Select which digital library to access (e.g. IEEE)
 - Enter the article title in the search box

READ THIS FIRST - 15-20 minutes

Topic: Modern Systems

This activity is designed to

- Learn about design of modern systems
- Tie up loose ends regarding upcoming assignment(s)

Roles:

- **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.
- **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.
- **Note Taker** - takes record of the group’s discussion in the Google doc (below).
- **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity or the assignments due tonight, type in Red text

All teams will do the following task:

- Work in groups
- Look at the architecture of the following systems and determine what architectural styles they use
- Recreate these architectures with components and connectors labeled and post them to the google slides
- List the NFPs achieved by the architecture style

Use a Component-Connector notation. Do not specify the architecture style - class will determine

The urls below are simply starting points

1.Android application: http://elinux.org/Android_Architecture - **Dawn**

2.Amazon AWS: <https://aws.amazon.com/blogs/architecture/lets-architect-designing-well-architected-systems/> - **The Boffins**

3.MS Azure: <https://www.interviewbit.com/blog/azure-architecture/> - **To be decided**

4.Google search engine: <https://krazytech.com/technical-papers/how-does-google-search-engine-work> - **Team 4**

5.eBay: http://www.cs.mcgill.ca/~mahmed26/eBay_Architecture_Study.pdf - **Ludus**

6.Twitter: https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale - **The Magratheans**

7.YouTube: <http://highscalability.com/youtube-architecture> - **Team 9**

8.Linkedin :

https://www.slideshare.net/r39132/linkedin-data-infrastructure-qcon-london-2012/9-LinkedIn_Architecture_A_web_page **Open source web services**

Team Name: Dawn

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator : Chloe

TimeKeeper: Abdul

NoteTaker: Luo

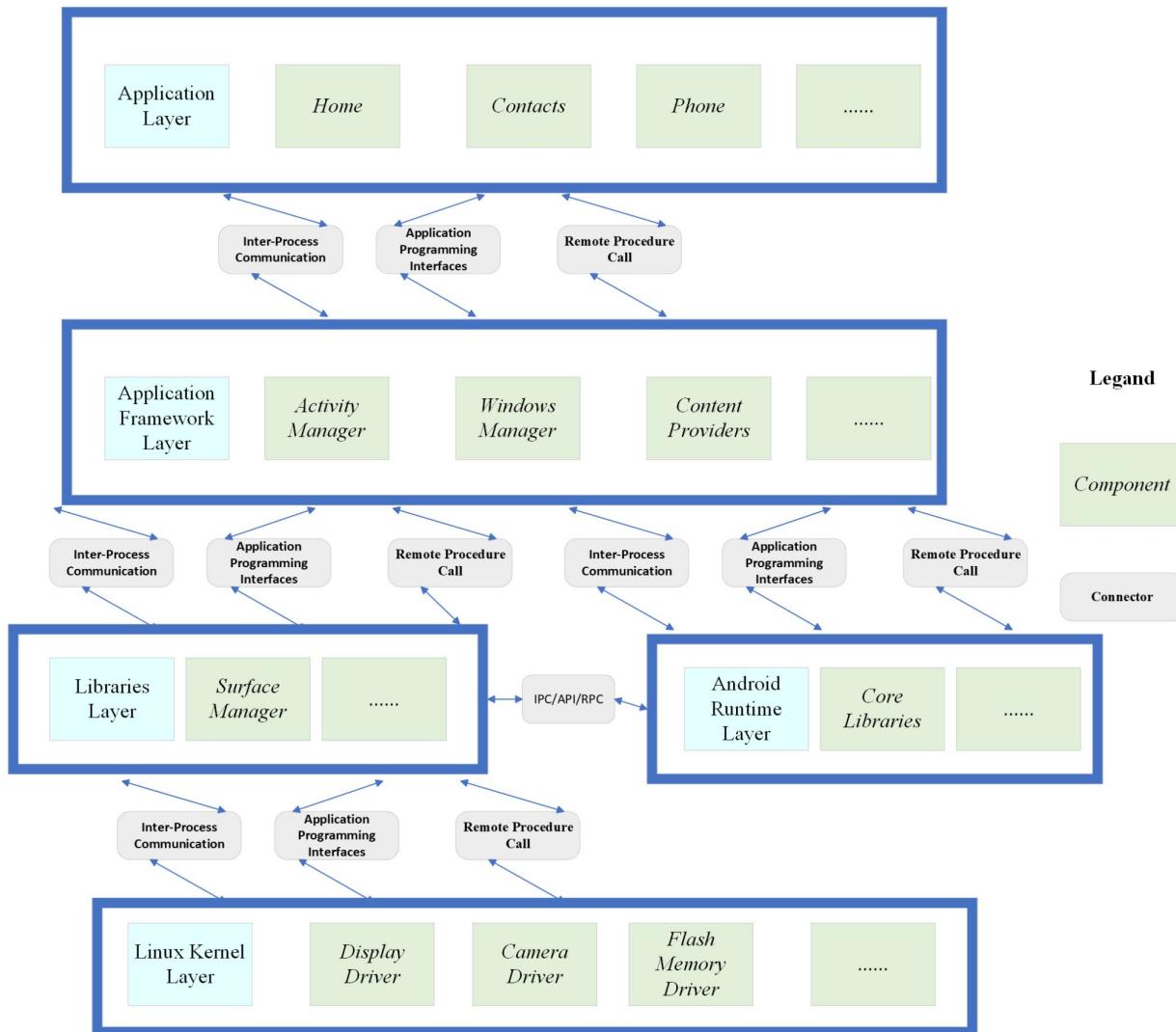
Reporter: Barack

Android architecture is divided into four main parts:

1. **Applications**: This is where all the applications like contacts, browser, games, etc. are located. These applications are written in Java programming language.
2. **Application Framework**: This layer provides higher-level services to applications in the form of Java classes. It includes managers that manage various types of data and services like resource manager, notification manager, activity manager, content providers, etc.
3. **Libraries and Android Runtime**: This layer is a set of C/C++ libraries used by various components of the Android system. It provides a wide range of libraries to perform different tasks. The Android Runtime includes the Dalvik Virtual Machine and Core Java libraries.
4. **Linux Kernel**: This is the foundation of the Android platform. It provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display, etc.

Question: List NFPs for architecture style

1. **Reliability:** The layered architecture style contributes to the reliability of the system. If one layer fails, the impact on other layers can be minimized.
2. **Scalability:** The component-based architecture of the application layer allows for easy scaling. New applications can be added without affecting existing ones.
3. **Reusability:** In the application layer, each application is developed as a separate component that can function independently. This allows for the reuse of applications across different devices or contexts without the need for significant changes.
4. **Security:** The Microkernel architecture of the Linux Kernel layer helps enhance security. The separation of user-space processes from kernel functionalities reduces the risk of system crashes and security breaches.
5. **Portability:** The use of the Java programming language for applications and most framework code, along with the Dalvik VM, contributes to the portability of Android. Applications can run on any device that has a Dalvik VM, regardless of the underlying hardware.
6. **Maintainability:** The clear separation of concerns in the Layered architecture and the Component-based architecture simplifies maintenance. Issues can be isolated to specific layers or components within that layer, making it easier to identify and fix problems.
7. **Performance:** The use of a Virtual Machine (VM) allows for efficient use of system resources, which can enhance performance. Also, the Native Libraries, Daemons, and Services layer, written in C/C++, can offer high performance for lower-level functionalities.



Team Name: The Boffins

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

NFPs:

- Scalability: Elastic Load Balancing allows requests to scale to large volumes
- Security: Each request passes through an EC2-driven security group and outputs all logs to an S3 bucket for auditing purposes
- Availability: Customers can leverage CloudFront distributions to reach wide availability zone coverages
- Performance: AWS utilizes Elastic Block Storage (EBS) to process data efficiently

List Roles and Student names

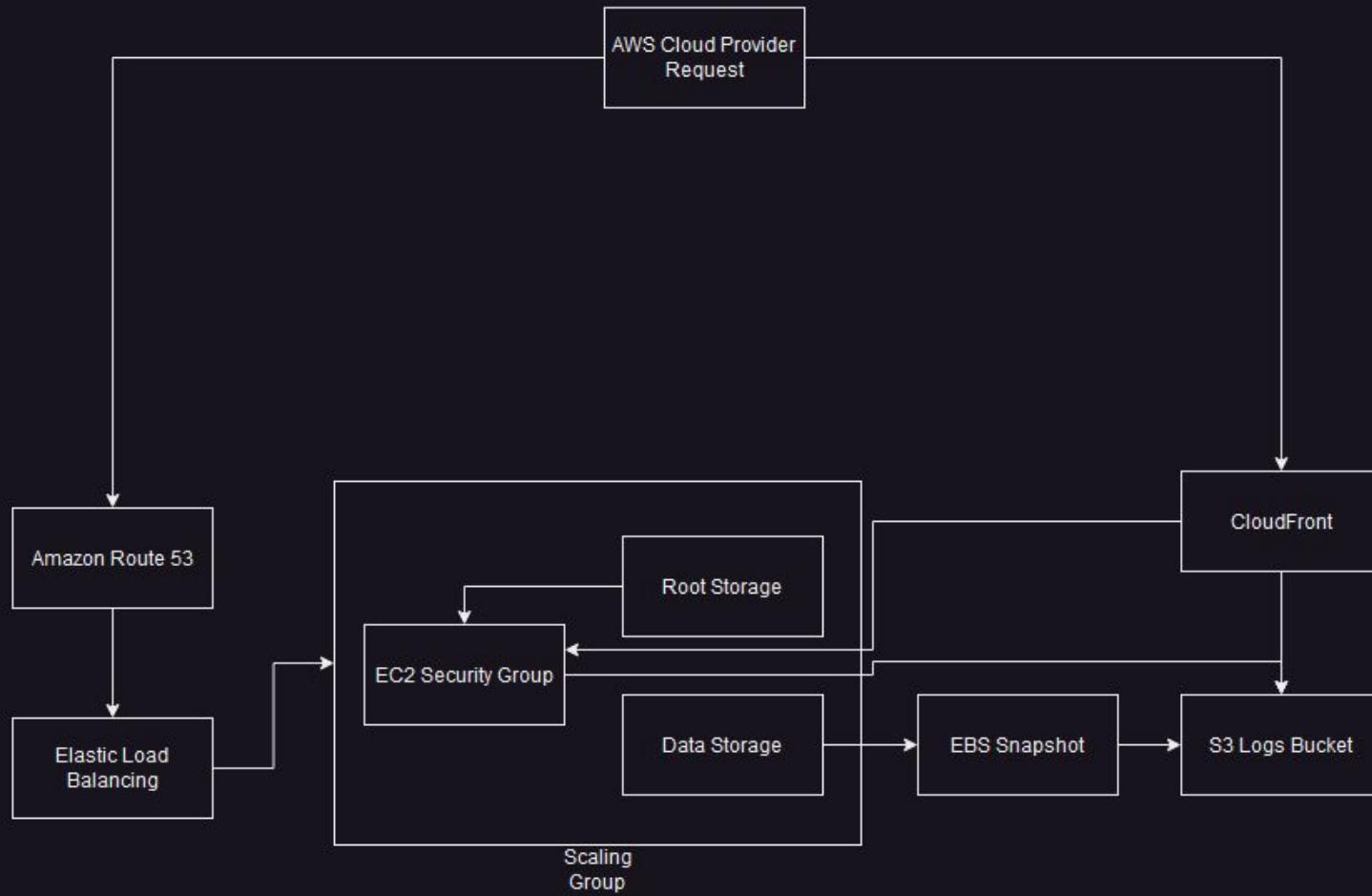
Moderator :

TimeKeeper:

NoteTaker:

Reporter:

AWS



Team Name: To be decided

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

“...is a major technique for reducing the amount of physical hardware required for a server farm”

“Each application is discrete, but the subsequent sources can assist”

everything is automated

each rack has a bunch of different servers

control portal (manage stuff)

Azure Resource Manager (talks to the service provider) - similar to GFS

portal talks to ARM, the expert / controller is the portal that talks to all of the other services

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

“...is a major technique for reducing the amount of physical hardware required for a server farm”

“Each application is discrete, but the subsequent sources can assist”

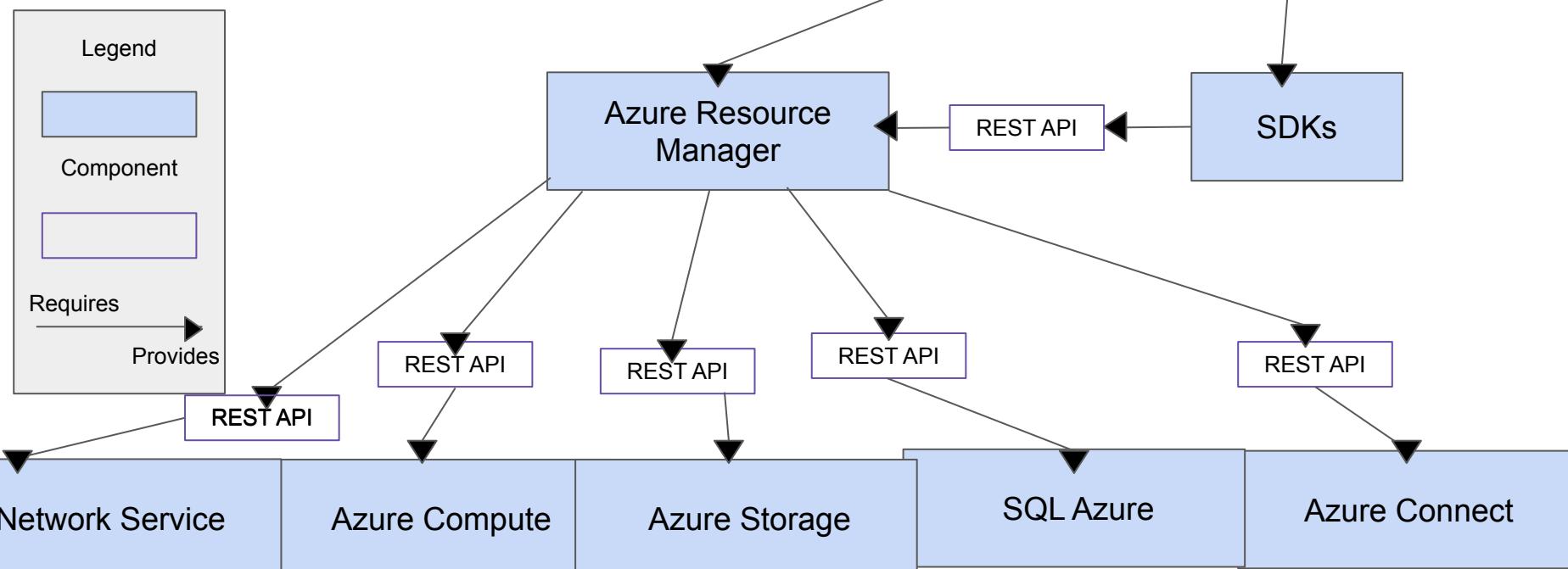
everything is automated

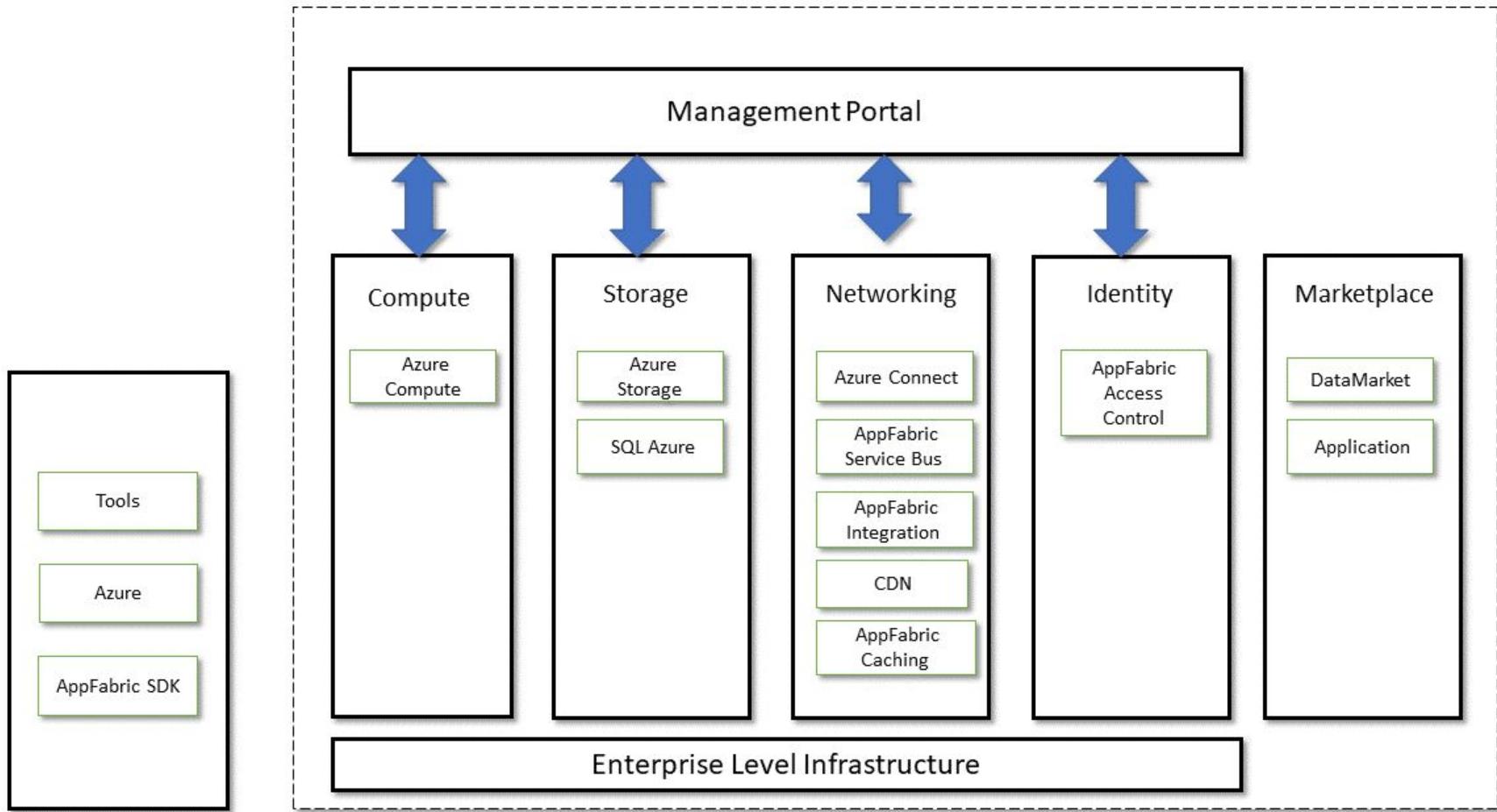
each rack has a bunch of different servers

control portal (manage stuff)

Azure Resource Manager (talks to the service provider) - similar to GFS

portal talks to ARM, the expert / controller is the portal that talks to all of the other services





Team Name: Team 4

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

4. Google search engine: <https://krazytech.com/technical-papers/how-does-google-search-engine-work> - Team 4

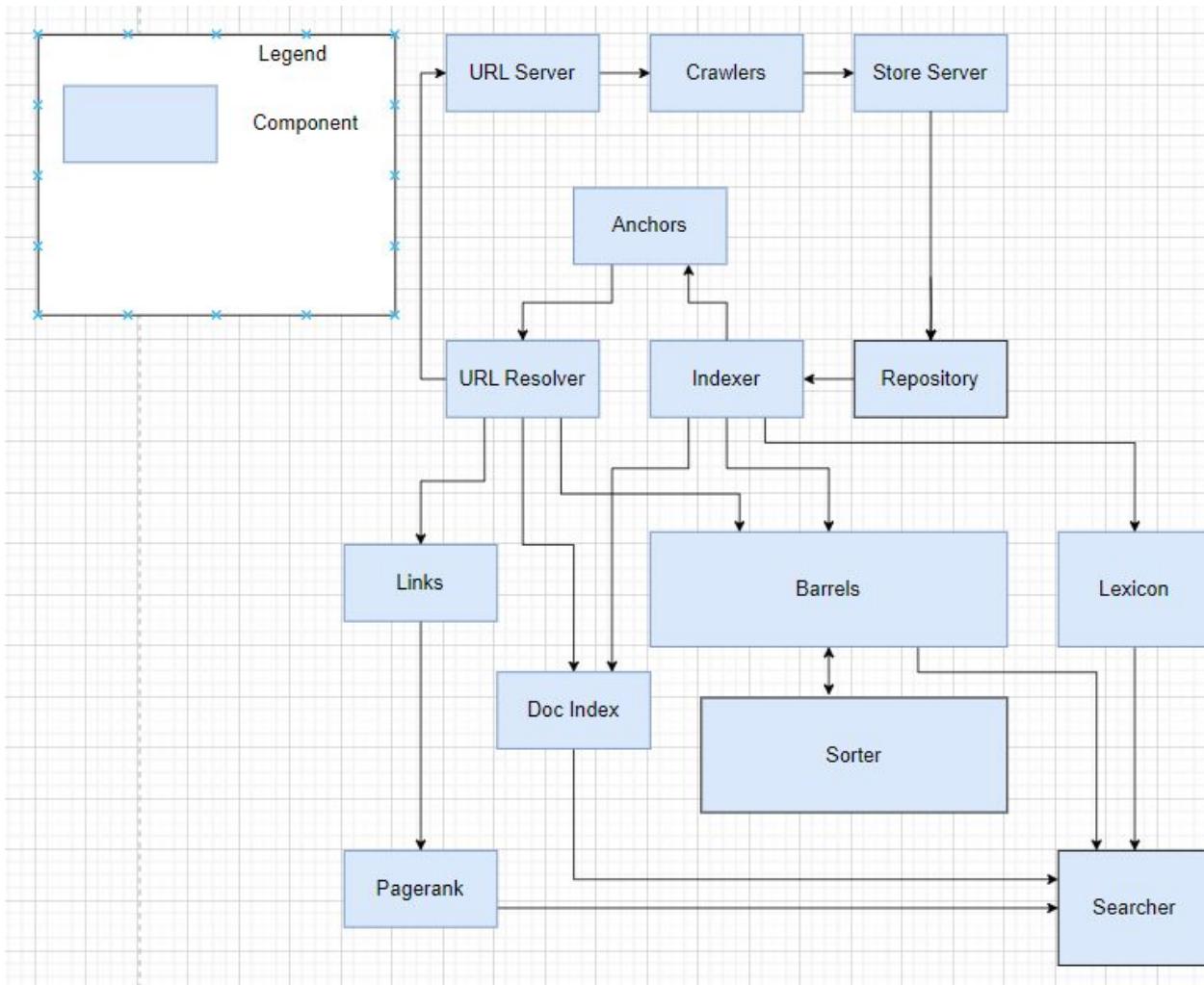
List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:



Team Name: Ludus

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

eBay: http://www.cs.mcgill.ca/~mahmed26/eBay_Architecture_Study.pdf - *Ludus*

- Work in groups
- Look at the architecture of the following systems and determine what architectural styles they use
 - 3 tiered client server
- Recreate these architectures with components and connectors labeled and post them to the google slides
- List the NFPs achieved by the architecture style
 - Availability/reliability
 - Security
 - Quality
 - Modifiability
 - High performance
 - Scalability

Legends

Component

Conector

Data Acess

Data Base

Client

HTTPs Request

DNS Load Balancer

Proxy Server

Load Balancer

HTTPs Request

Web Servers

RPC / REST

Application
Servers

Data Acess

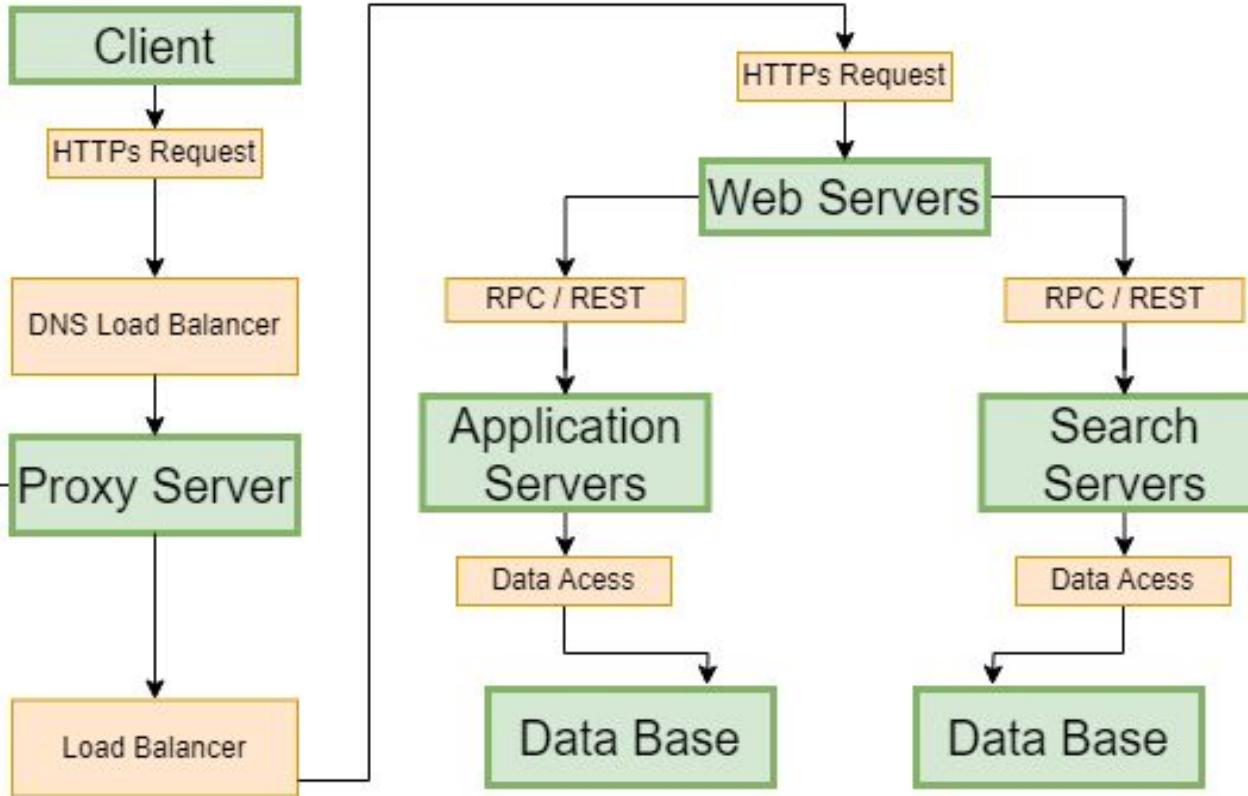
Data Base

RPC / REST

Search
Servers

Data Acess

Data Base



Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: MAGRATHEANS

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

- Look at the architecture of the following system and determine what architectural styles they use
- Recreate these architectures with components and connectors labeled and post them to the google slides
- List the Non Functional Properties achieved by the architecture style
 - Scalability
 - Reliability
 - Availability
 - Speed
 - Security
 - Efficiency

https://blog.twitter.com/engineering/en_us/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale

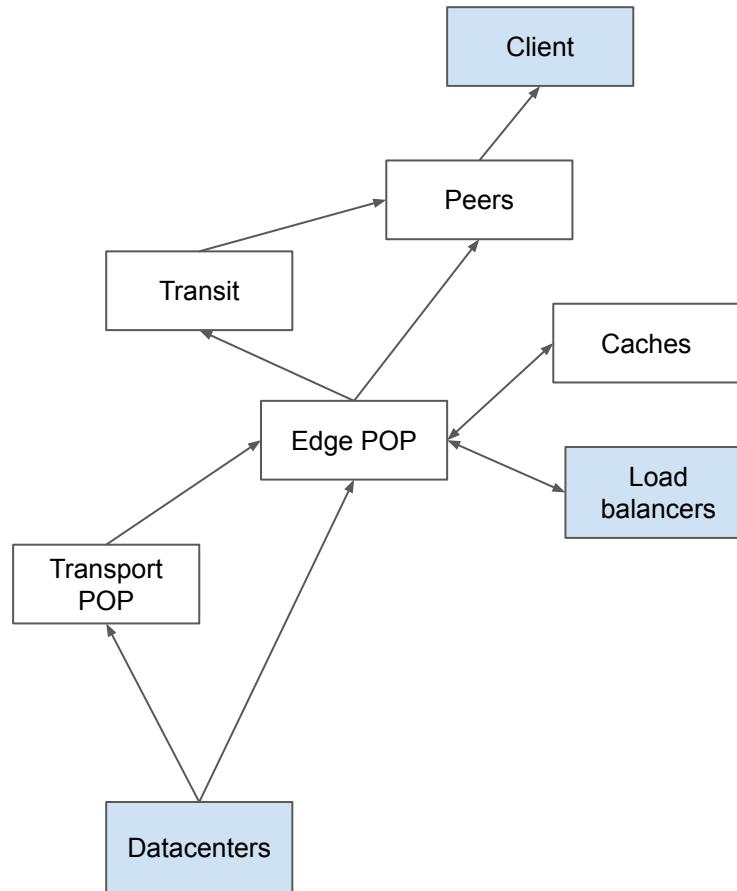
Legend

Component

Connector



Interface connection



Team Name: Team 9

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

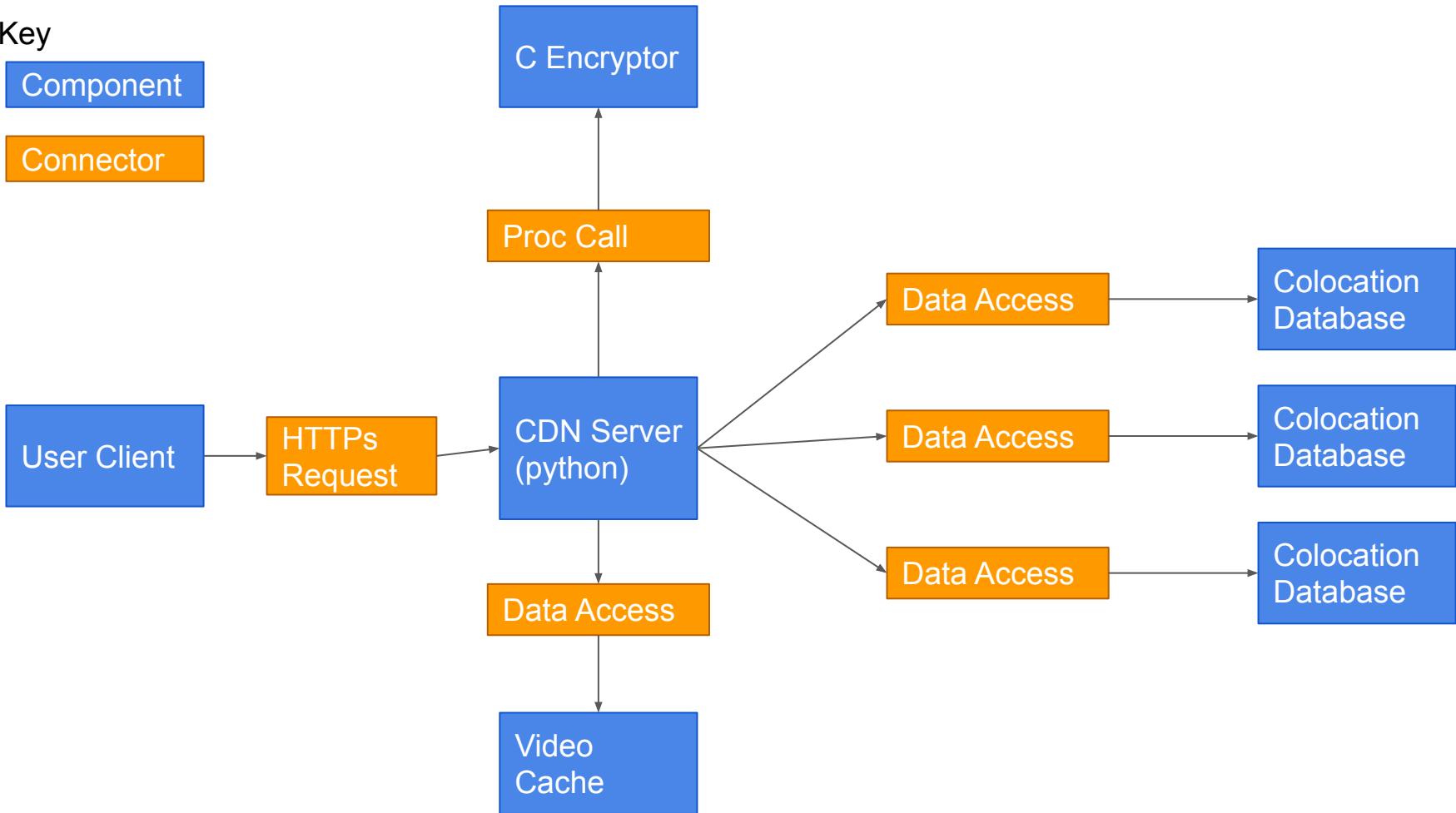
NoteTaker:

Reporter:

Key

Component

Connector



Team Name: Open Source Web Services

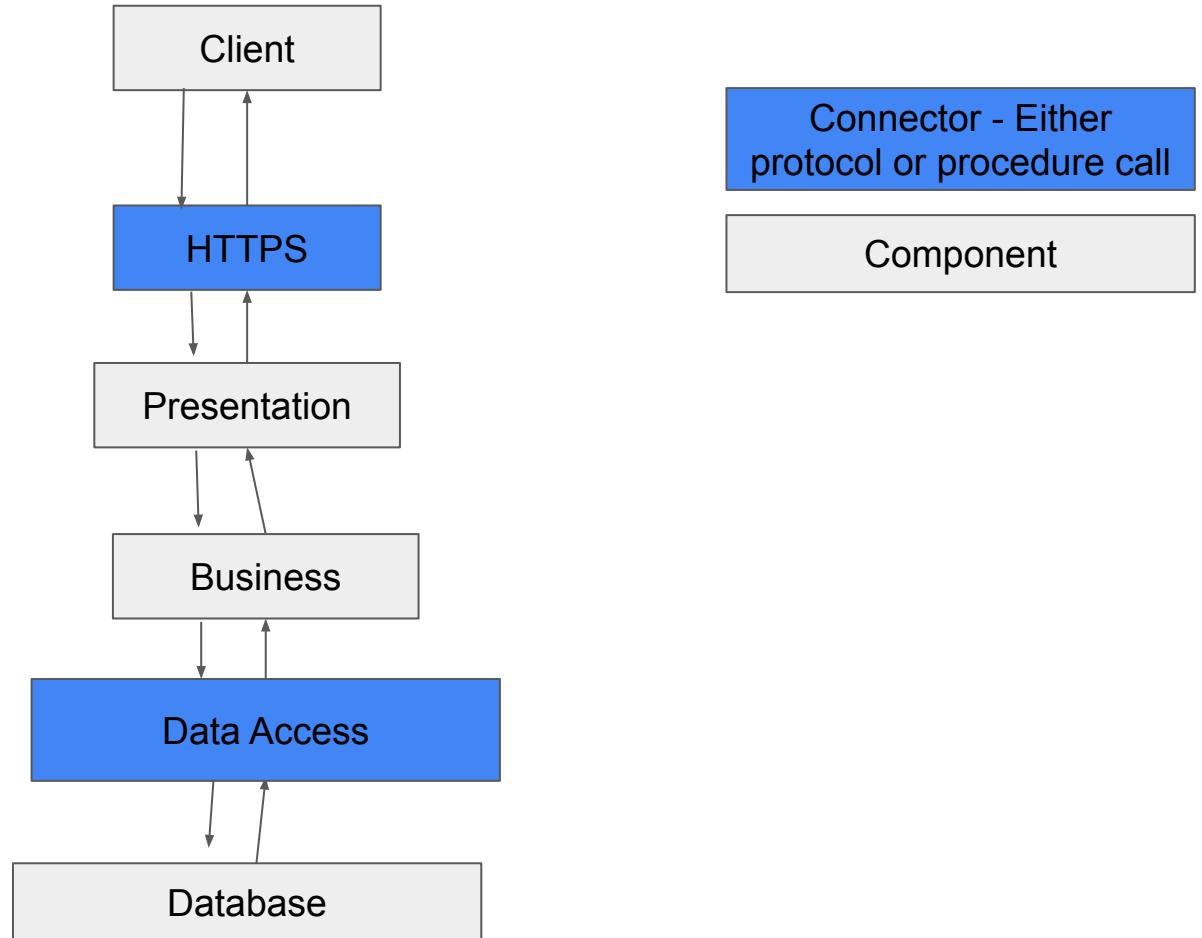
Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

- Look at the architecture of the following systems and determine what architectural styles they use
- Layered Architecture style**
- Recreate these architectures with components and connectors labeled and post them to the google slides
 - List the NFPs achieved by the architecture style}
 - **Maintainability**
 - **Modularity**
 - **Portability**
 - **Reusability**
 - **Interoperability**
 - **Scalability**
 - **Security**



READ THIS FIRST - 15-20 minutes

Topic: Review

This activity is designed to

- Review arch styles learned throughout the quarter
- Tie up loose ends regarding upcoming assignment(s)

Roles:

- **Moderator** - helps facilitate the conversation; a “project manager” who helps the group come up with a plan for using time wisely, and to participate equitably. Makes sure each team member has a chance to speak.
- **Timekeeper** - keeps the group on track by being mindful of the time; lets everyone know when they need to wrap up or move on to another task.
- **Note Taker** - takes record of the group’s discussion in the Google doc (below).
- **Reporter** - after we regroup as a class, the reporter summarizes to the whole class the group’s discussion (will need to have video and audio turned on). Reporter will defend the group's answer in class.

Before you begin your tasks:

1. Please assign roles and record them in your group’s section. Try to take a different role each time.
2. Then respond to the prompts, taking notes on your discussion

If you have any questions about this activity or the assignments due tonight, type in Red text

All teams will do the following task:

- Work in groups
- Create an architecture for allowing your different home appliances to communicate with each other.
What architectural style would you use? Draw your architecture.
-

Use a Component-Connector notation. Do not specify the architecture style - class will determine

Team Name: Dawn

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Are Agile SDLCs and architecture an oxymoron?

List Roles and Student names

Moderator : Chloe

TimeKeeper: Luo

NoteTaker: Abdul

Reporter: Barack

Not oxymorons

1. They're actually different concepts focusing on different topics. Agile focus on how to manage the development flow while architecture focuses on what the construction of the software should look like. If we give an analogy, Agile tells us how to mange the task, including defining reqruiemtns, mange human resources, monitor the project progress while architecture is about how the house should look like.
2. - Agile Software Development Life Cycles (SDLCs) and architecture are not oxymorons. They can work well together.
3. - Agile methodologies prioritize working software over comprehensive documentation. This might appear to contradict the traditional role of software architecture, which involves a significant amount of planning and documentation.
4. - Agile methodologies develop software in small, incremental updates, allowing more flexibility and responsiveness to changes.
5. - In Agile, architecture is not ignored. Instead, it evolves alongside the software instead of being fully defined upfront.
6. - The balance is crucial between an architectural framework that supports system requirements (performance, security, scalability) and the flexibility to adapt the architecture as requirements change.
7. - This approach of allowing the architecture to evolve is often referred to as "evolutionary architecture."
8. - Agile practices supporting a flexible, evolutionary approach to architecture include refactoring, continuous integration, automated testing, and iterative development.
9. - While Agile and architecture might seem to be at odds superficially, they can be effectively integrated with the right practices and mindset.

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name: Team 4

Discussion points

Task 1: List questions about this activity or any in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide

Is Agile SDLCs and Architecture an oxymoron?

What arch style to use if...

- You wish to allow multiple organizations to control own resources
- Distributed

tion here, come to your room, or answer

No, it is not. Agile defines the products to be iterative and undetermined at early stages, but the development can still follow architectures. Iterative development can follow architectures through considerations of architecture requirements in each iteration. Iterative development is not contradictory with architectures, though it may be difficult to have a full view of the architecture from the beginning since the initial design is just for one iteration instead of the whole system. In short, compliance to an architecture in each step can lead to compliance to an architecture as an entire system since compliance in each step is in fact stricter.

Distributed - Mobile Code (Remote Evaluation)

The team could also refactor some code to achieve certain architecture styles.

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:

Team Name:

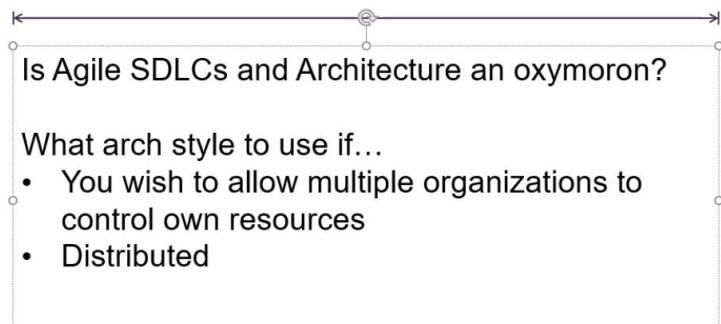
Task 1: List questions about this activity or any of the upcoming assignments. I may answer your question here, come to your room, or answer in the main room (3 mins)

Ques #1:

Ques #2:

Task 2: see assigned task and instructions on slide 2. You may add another slide if needed (15 minutes)

Discussion points



List Roles and Student names

Moderator :

TimeKeeper:

NoteTaker:

Reporter:



Computing and Software Systems
<http://www.uwb.edu/css>

CSS 553: Software Architectures



Hazeline Asuncion
Spring 2023

Outline



- Review
- Course overview/wrap-up
- Activity
- Course Survey

Review



- Robotics
 - Different arch styles?
- Wireless sensor networks

Reflection Questions



Question: When going over week 10's lecture, I kept in mind to wonder about the fundamental structures I could find. For example WSN I think if it's a fundamental Event Based C2? not sure if I'm correct or not. But with p2p elements in MIDAS, I think it will not qualify as C2 anymore. It would be nice if someone can answer me here.

Questions – 10 mins breakout room



- With the availability powerful hardware and off-the-shelf artificial intelligence software, how do these change the discussed architecture styles in the field of Robotics?
 - Dawn, the Boffins, Team 4
 - Dawn: Sense-Plan-Act – more powerful hardware – higher resolution(Sensor) video, can do more complex operations (Act), better algorithms (Plan is more efficient)
 - Subsumption – highly modularized – divided into different levels, more powerful hardware can accommodate more modules, better AI (need powerful hardware) – assist with complexity
 - Boffins – 3 layer architecture – different layers use cloud computing – hardware – high internet speed (latency)
 - Team 4 – bandwidth is a constraint of WSN – better hardware assist with this. AI assist with recognize patterns in data from sensors

Questions – 10 mins breakout room



- You saw an example of using multiple architecture styles in IoT (MIDAS System). Find a modern-day IoT system and identify at least two architecture styles present in the system.
- -Ludus, Open Source Web services
 - Ring – client-server, event-based (notification when detect video)
- Pick one domain (aside from Robotics and IoT). Identify key requirements in this domain and create an ideal architecture style (or hybrid of styles) to satisfy these requirements.
 - The Magratheans, Team 9
 - Public Libraries (books, media) – requirements: control – digital representations, non-repudiability. Arch style: REST – interactions between clients (patrons), good for scalability, reliability; P2P – different libraries connect with each other (more reliable)
 - E-Commerce (Amazon) – requirements: scalability, availability, security, extensible; Arch style: layered (scalability, availability, security), client-server (availability, extensibility), each layer has redundant components (hardware)

Clarifications



- Direction of arrows

Explicit
invocation

Required
interface

Provided
interface

Implicit
invocation



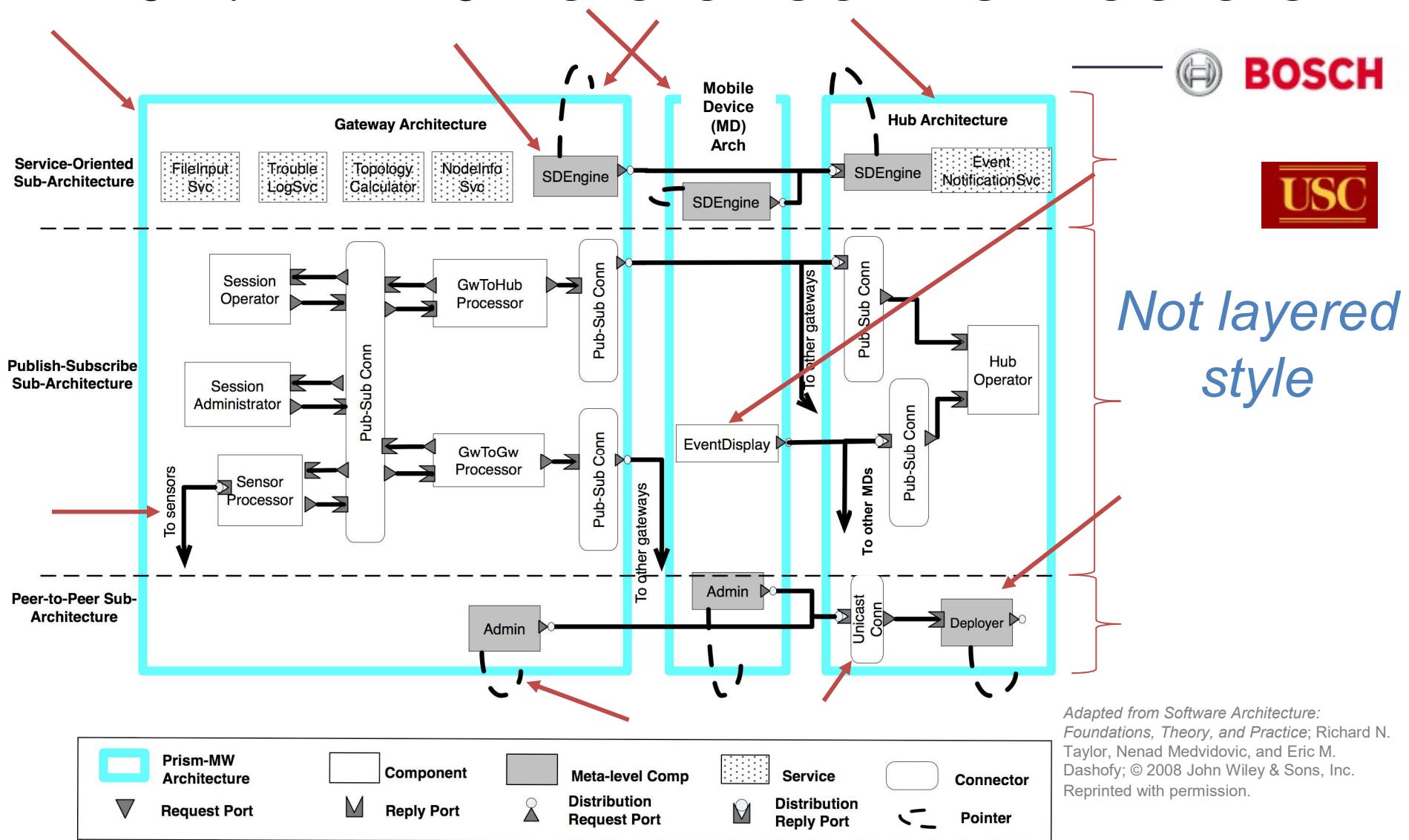
- Component-connector notation
- Legend

Practice designing



- Identify functional and non-functional requirements for the system listed in next slide
- Based on these requirements select your architecture style and draw your diagram
- Compare your style with the architecture style provided in the links

WSN: MIDAS Reference Architecture



Adapted from Software Architecture:
Foundations, Theory, and Practice; Richard N.
Taylor, Nenad Medvidovic, and Eric M.
Dashofy; © 2008 John Wiley & Sons, Inc.
Reprinted with permission.

COURSE OVERVIEW/WRAP-UP

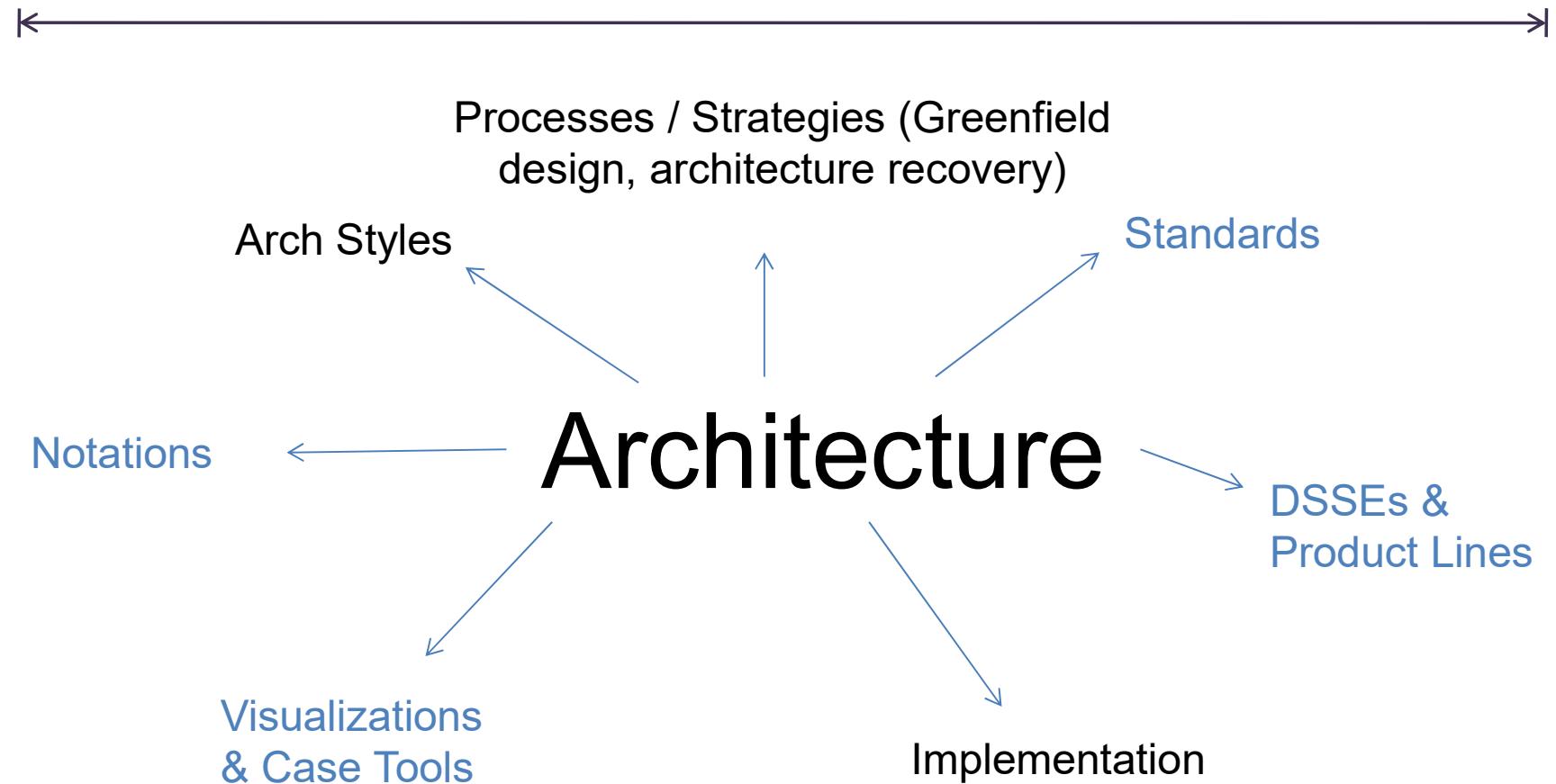
Running Threads in the class



- Software is complex
- Learn from experience

Architecture

Running Threads in the class



Course objectives



- Name and describe software architecture concepts, methods, and best practices
- Select and justify the use of software architecture styles or patterns for a given situation
- Apply software architecture styles or patterns on a small but realistic project
- Use CASE tools in creating, visualizing, and analyzing software architecture artifacts



Discussion points



Is Agile SDLCs and Architecture an oxymoron?

- No, focus on different topics Agile is on full development process, architecture – design
- Architecture does exist in agile – watch it drift, then bring it back (monitor it) – refactoring
- Architecture can be done iteratively (twin peaks) – along with agile

What arch style to use if...

- You wish to allow multiple organizations to control own resources
- Distributed
 - SOA – allow diff org to control own resources

Good sources of readings



- Software Architecture conferences
 - <https://icsa-conferences.org/series/history/>
- Software Engineering conferences
 - <http://www.icse-conferences.org/>
 - <https://www.esec-fse.org/>
 - <http://www.ase-conferences.org/>
- Websites
 - <http://www.softwarearchitectureportal.org/>

Good sources of readings



- Library resources – full articles
 - <https://guides.lib.uw.edu/bothell/css>
 - From off-campus, login
 - Select which digital library to access (e.g. IEEE)
 - Enter the article title in the search box

Final Exam Procedure



- Same as Midterms
- Choose architecture style
- Draw diagram
- How to prepare?
 - Look at reflection questions,
 - Open-ended questions
 - Questions at the end of chapters in book
- Jun 9 – Friday

ACTIVITIES

G4 Questions



- Need more time?
 - Submit by 11:59pm on Monday, Jun 5th
 - Submit all questions by 11:59, Tue, Jun 6
 - 1 team asks 2 questions total (1 question for 2 teams)
 - Submit all answers by 11:59, Wed, Jun 7

Activities



1. Modern Architecture diagrams
2. Find the connectors

Activity – 15 mins



- Work in groups
- Look at the architecture of the following systems and determine what architectural styles they use
- Recreate these architectures with components and connectors labeled and post them to the discussion board (Architecture of Modern Systems)
- List the NFPs achieved by the architecture style

Activities



- Evaluate the different architectural styles and specify
 - Advantages
 - Disadvantages
 - The context to apply it
- Create an architecture for allowing your different home appliances to communicate with each other. What architectural style would you use? Draw your architecture.

Scenario



- Smart-home System
 - What architecture style?

Activities



- Silver bullet
 - See Brooks-NoSilverBullet.pdf
 - Do you think automatic code generation (e.g., 1.x way mapping) is a silver bullet?

COURSE SURVEY

Course Evaluation



Please do