



Software Engineering
Software Architecting
Elegant tools for a more civilized age.

Dr. Hale
University of Nebraska at Omaha

Continued from last time.

Part 3: Software Architectures

- Architectures 101

- Architectural patterns

- Benefits

- Examples

- Adding Security Components

- Asking the right questions

Hands-on

- Build an architecture diagram of your app idea

- What security components should be added?

Part 3: Software architecture

Software Architecture: Origins

Software Engineers have always employed software architectures
Very often without realizing it!

A “field” was needed to address issues identified by researchers and practitioners

- Essential software engineering difficulties
- Need for software reuse

Christopher Alexander – Architect and professor

Notes on the Synthesis of Form

- required reading for researchers in computer science throughout the 1960s.
- Influenced programming language design, modular programming, object-oriented programming, software engineering and other design methodologies

Edsger Dijkstra

A Discipline of Programming

- Emphasized mathematical concepts, orientation, and system structure

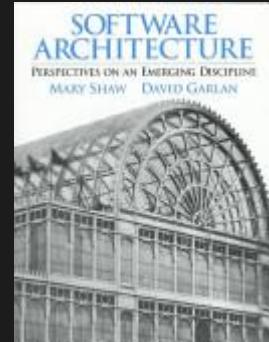
Mary Shaw and David Garlan

Software Architecture: Perspectives on an Emerging Discipline

- Codified standard terms like component, connector, and design patterns

NOTES ON THE
SYNTHESIS
OF FORM

CHRISTOPHER ALEXANDER



Software Architecting

- An architecture-centric approach to software engineering **places an emphasis on design**
- Design **pervades the engineering activity** from the very beginning
- This occurs in three phases (usually iterative)
 - **Analysis**
 - Determining Requirements from the customer
 - Analyzing requirements to determine what components are needed (what tools?).
 - **Design (modeling)**
 - Designing and diagramming components and there interactions.
 - Refining the design to improve efficiency, usefulness, security, etc.
 - **Implementation**
 - Coding components and integrating them together
 - Testing to make sure the implementation meets the design

What is an Architecture?

A software architecture defines four Cs

Components: computational elements

clients, servers, databases, filters, layers

Connectors: interactions among components

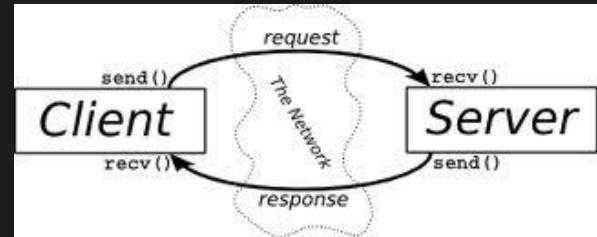
procedure call, shared variable, protocols, events

Configuration: patterns that guide composition of system elements

throughput, capacity, performance, scalability

Control Flow: how information passes through the elements

component capability, consistency with requirements



Ex. Client-server architecture

- Should be familiar, server provides a service, client requests and uses the service
- Client browser (at least) provides the UI
- Server provides the application and database

Components

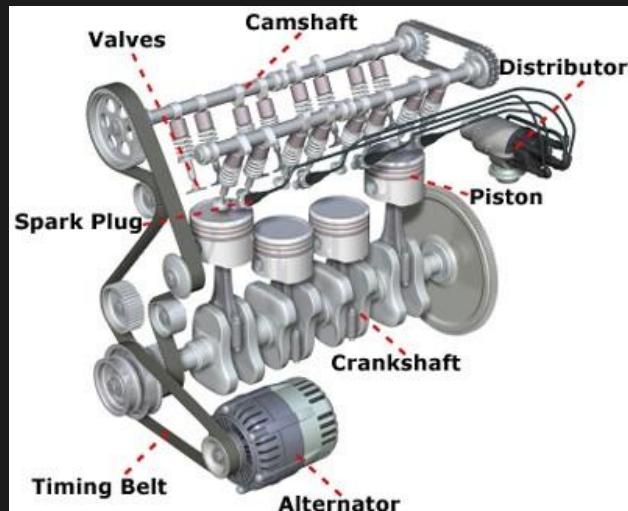
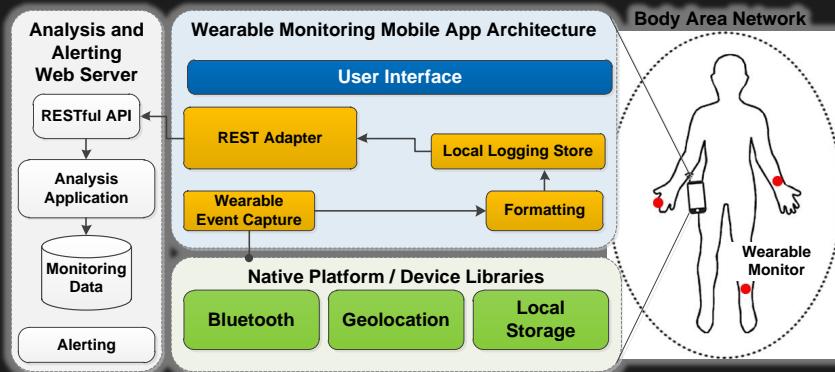
Elements that **encapsulate processing and data** in a system's architecture are referred to as software components

Definition

A *software component* is 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

Components are separable chunks much like car parts



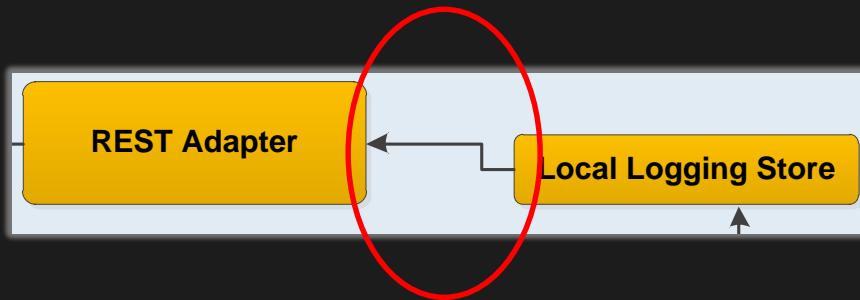
Connectors

In complex systems interaction may become more important and challenging than the functionality of the individual components

Definition

A *software connector* is an architectural building block tasked with effecting and regulating interactions among components. They define *interfaces*.

Connectors allow concerns to be separated into chunks

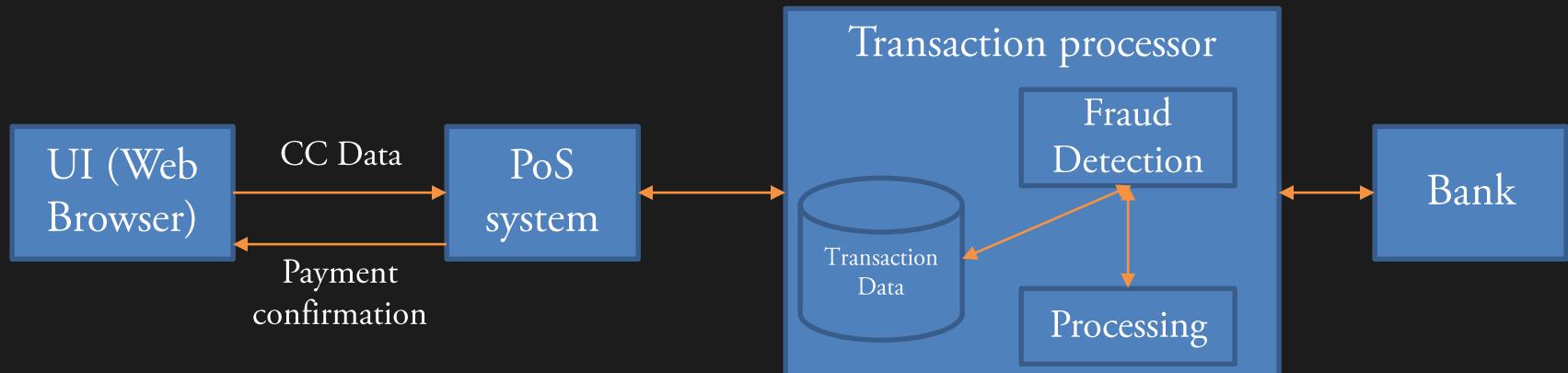


Configuration: Architectural models

Components and connectors are composed in a specific way in a given system's architecture to accomplish that system's objective

Definition

An *architectural configuration*, or topology, is a set of specific associations between the components and connectors of a software system's architecture. It can be high or low level.



a high level architecture

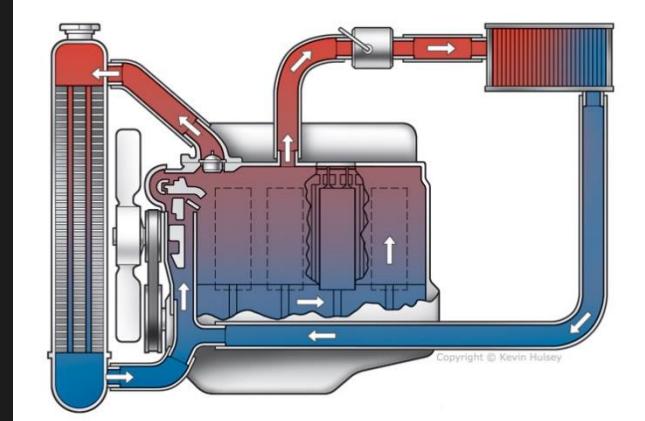
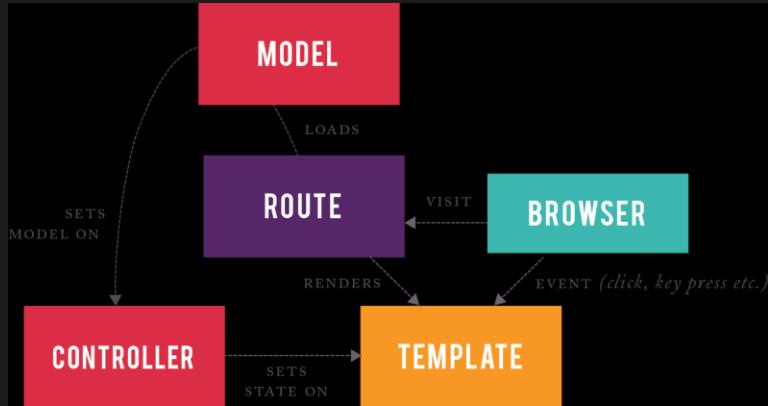
Control Flow

Control flow dictates how execution proceeds logically

Definition

A *control flow*, is a logical progression between components and across connectors that characterizes how an application works.

Although not always present on an architecture, control flows can be very beneficial for descriptive purposes, especially with abstract architectures



Another great tool for thinking about what a product should do

You should ALWAYS have a high level architecture diagram before you create anything.

It represents your single or collective vision.

It is useful when analyzing or evaluating a product too – since it tells you how the product
SHOULD function.

Coupling vs Cohesion

Building blocks (component AKA module)



Good architectures:

Minimize coupling

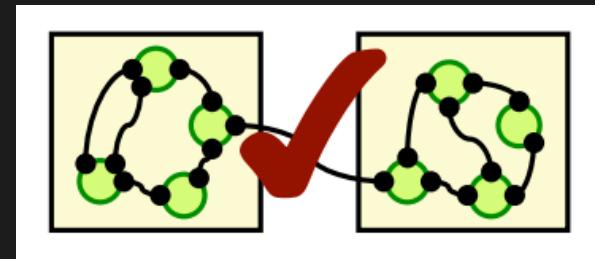
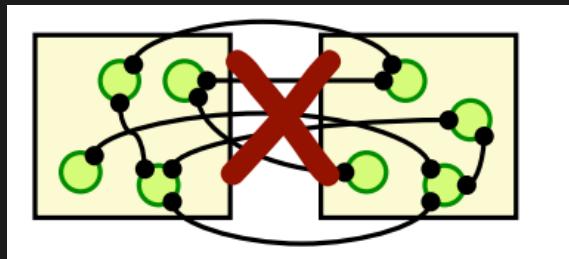
Goal: Modules don't need to know much about each other to interact

Improves: resiliency, extensibility, and minimizes vulnerability surface

Maximize cohesion

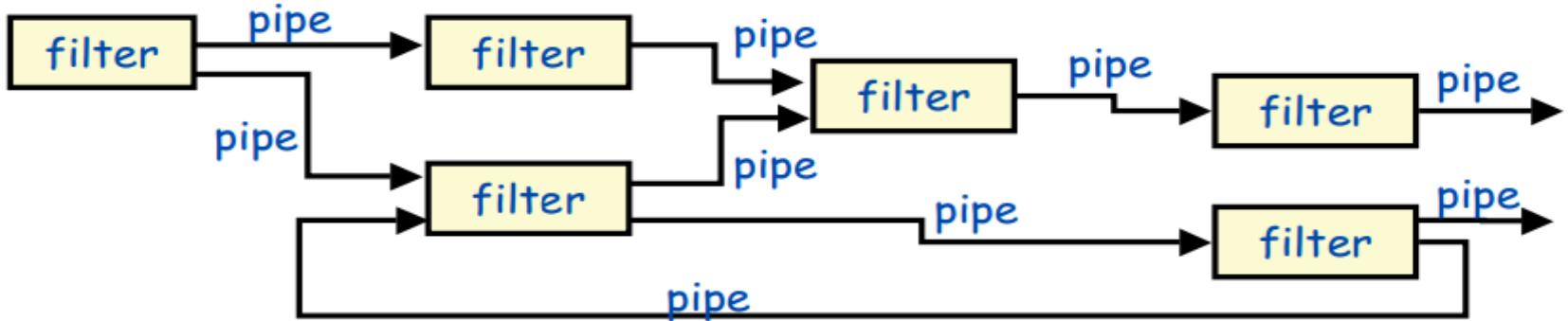
Goal: the contents of a single module are strongly inter-related

Improves: separation of concerns, extensibility, and reuse



Pipe and Filter

Source: Adapted from Shaw & Garlan 1996, p21-2. See also van Vliet, 1999 Pp266-7 and p279



Examples:

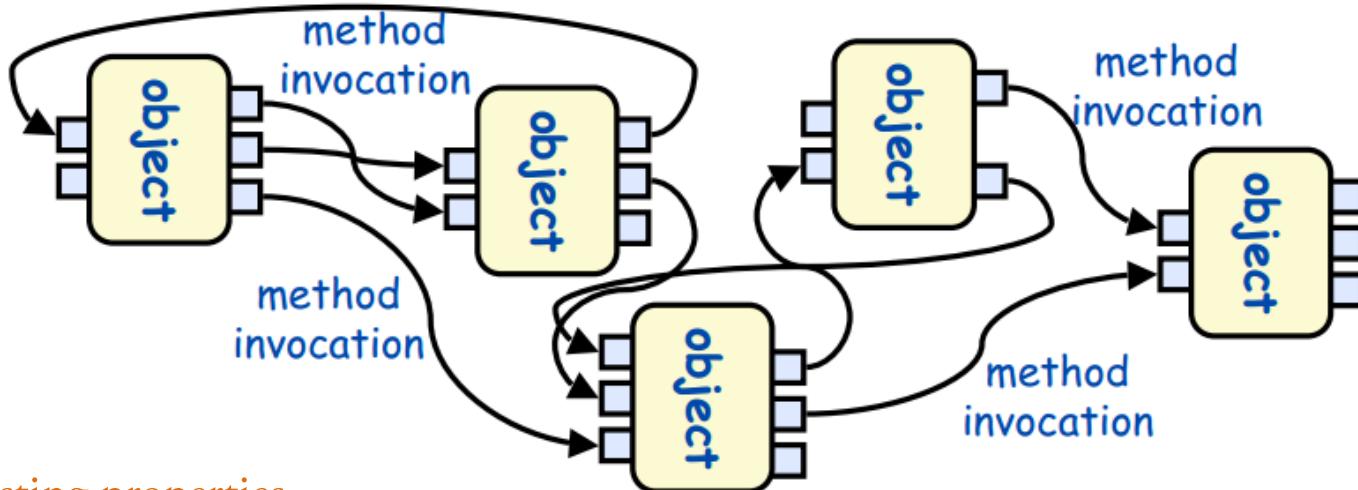
- Unix shell commands
- Compilers
 - Lexical analysis -> parsing -> semantic analysis -> code generation
- Signal processing

Interesting properties:

- Filters don't need to know anything about what they are connected to
- Can be parallelized
- System behavior is defined by the composition and behavior of filters

Object Oriented

Source: Adapted from Shaw & Garlan 1996, p22-3.



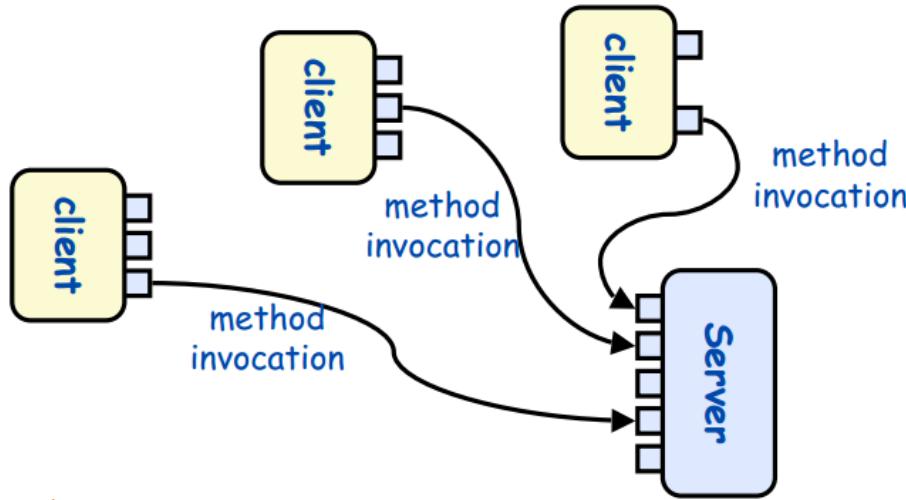
Interesting properties

- Data hiding (e.g. private, public, protected)
- Can be multi-threaded or single threaded
- Can decompose problems into sets of interacting agents

Disadvantages

Objects must know the identity of other objects they need to interact with

Object Oriented: Variant 1 – Client Server



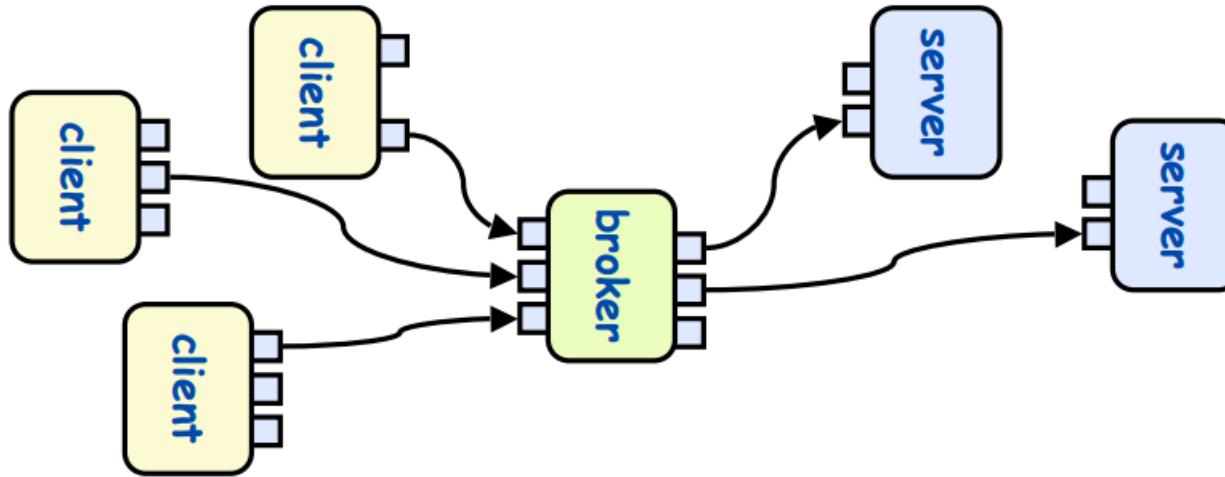
Interesting properties

- Object oriented properties
- Clients do not need to know about one another

Disadvantages

Objects must know the identity of other objects they need to interact with

Object Oriented: Variant 2 – Object Brokers



Example:

Load balancing, cloud services

Interesting properties:

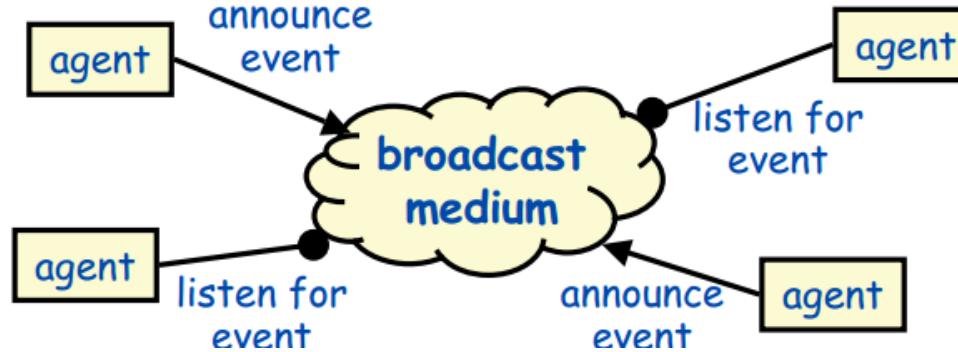
- Clients no longer need to know which server they are using
- Can have many servers
- Clients can move between servers (e.g. mobile location change)

Disadvantages

- Broker can be a bottleneck
- Degraded performance (e.g. response time)

Event Based

Source: Adapted from Shaw & Garlan 1996, p23-4. See also van Vliet, 1999 Pp264-5 and p278



Example:

- Debugging systems (listen for breakpoints)
- Dbms (data integrity checking)
- GUIs – listen for on-click or on-key events and handle them somehow

Interesting properties:

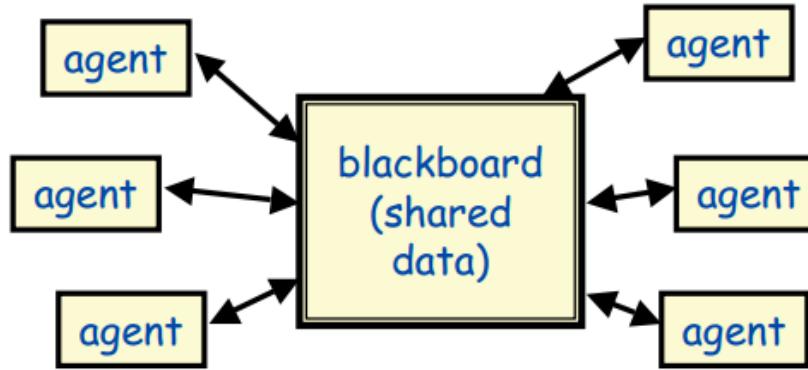
- Announcers don't need to know who will handle the event
- Ease of re-use and emergent evolution (add new agents)
- Asynchronous

Disadvantages

- Components have little-to-no control over temporal execution flows
- Asynchronous systems can be harder to code

Repositories

Source: Adapted from Shaw & Garlan 1996, p26-7. See also van Vliet, 1999, p280



Example:

- Databases
- Blackboard expert systems
- MVC frameworks

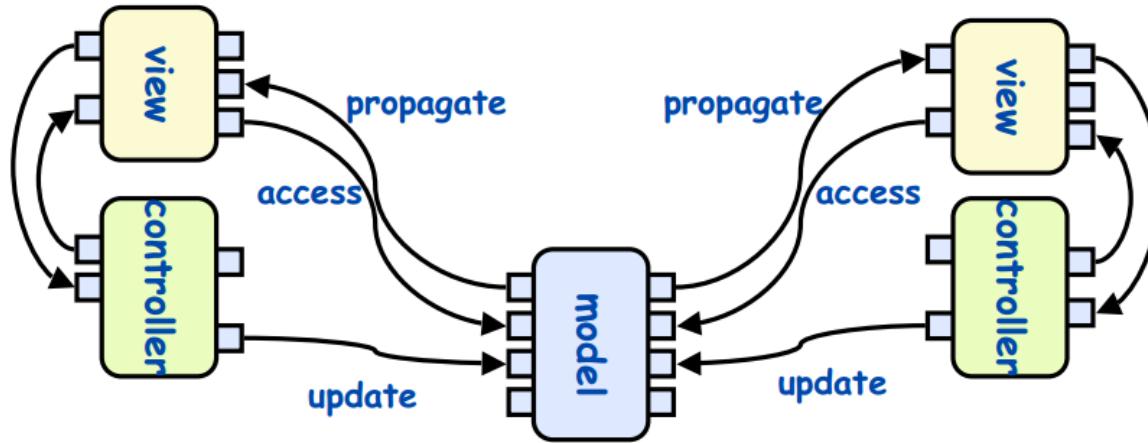
Interesting properties:

- Can choose where control is (agents, blackboard, both)
- Reduces the need to duplicate complex data

Disadvantages

- Blackboard can become a bottleneck (e.g. SQLite)

Repositories: Most important variant



Interesting properties:

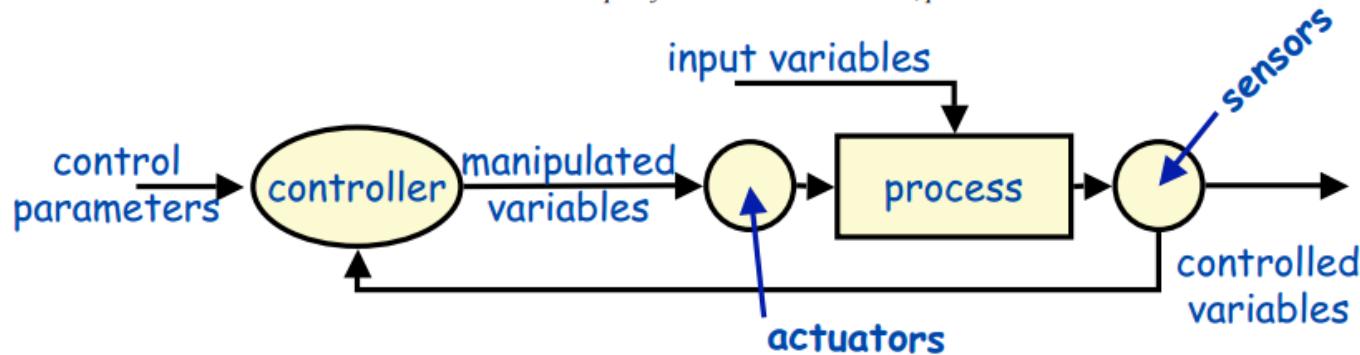
- One central model, many viewers
Changes to the model are propagated to all viewers
- Each view has a controller
 - Handles all user updates to model
- Separation of concerns much like a 3-layer architecture
 - Business logic in the controller
 - Database logic in the model
 - UI logic in the View

Challenges

- Ensuring separation of concerns (e.g. making sure UI logic stays out of the controller)
- Static vs asynchronous loading

Process Control

Source: Adapted from Shaw & Garlan 1996, p27-31.



Example:

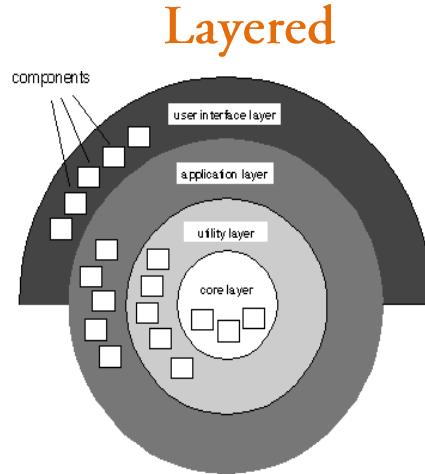
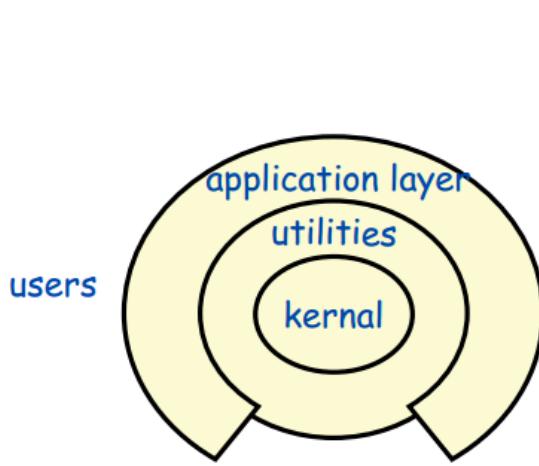
- Aircraft/spacecraft flight control systems
- Controllers for industrial production lines, power stations, etc
- Chemical engineering

Interesting properties:

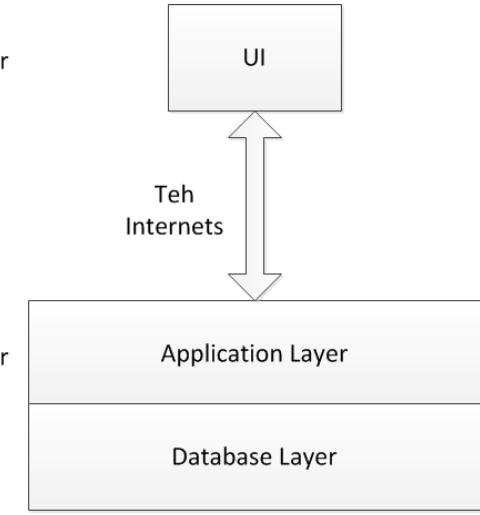
- Separates control policy from the controlled process
- Handles real-time, reactive computations

Disadvantages

- Difficult to specify timing characteristics and disturbance responses



Client Layer



Example:

- Operating systems, django, clouds (e.g. AWS), VMs

Interesting properties:

- Support increasing levels of abstraction during design
- Support enhancement and re-use
- Allows for separation of concerns using standardized layer interfaces

Disadvantages

- May not be able to cleanly separate layers

Layered: Open vs Closed

Closed architecture

Each layer only uses services of the layer below it

Advantages:

- Minimizes dependencies between layers
- Reduces the impact of layer change cascading

Open architecture

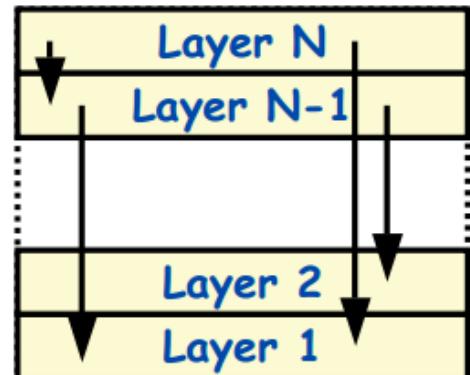
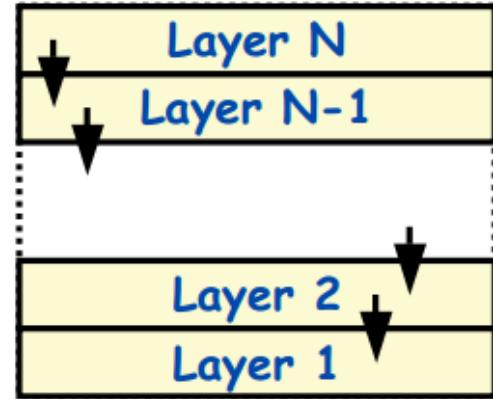
Each layer can use services from any lower layer

Advantages:

- More compact code (less middleware function calls)

Disadvantages:

- Breaks layer encapsulation increasing dependencies between layers
- makes layer change impact higher

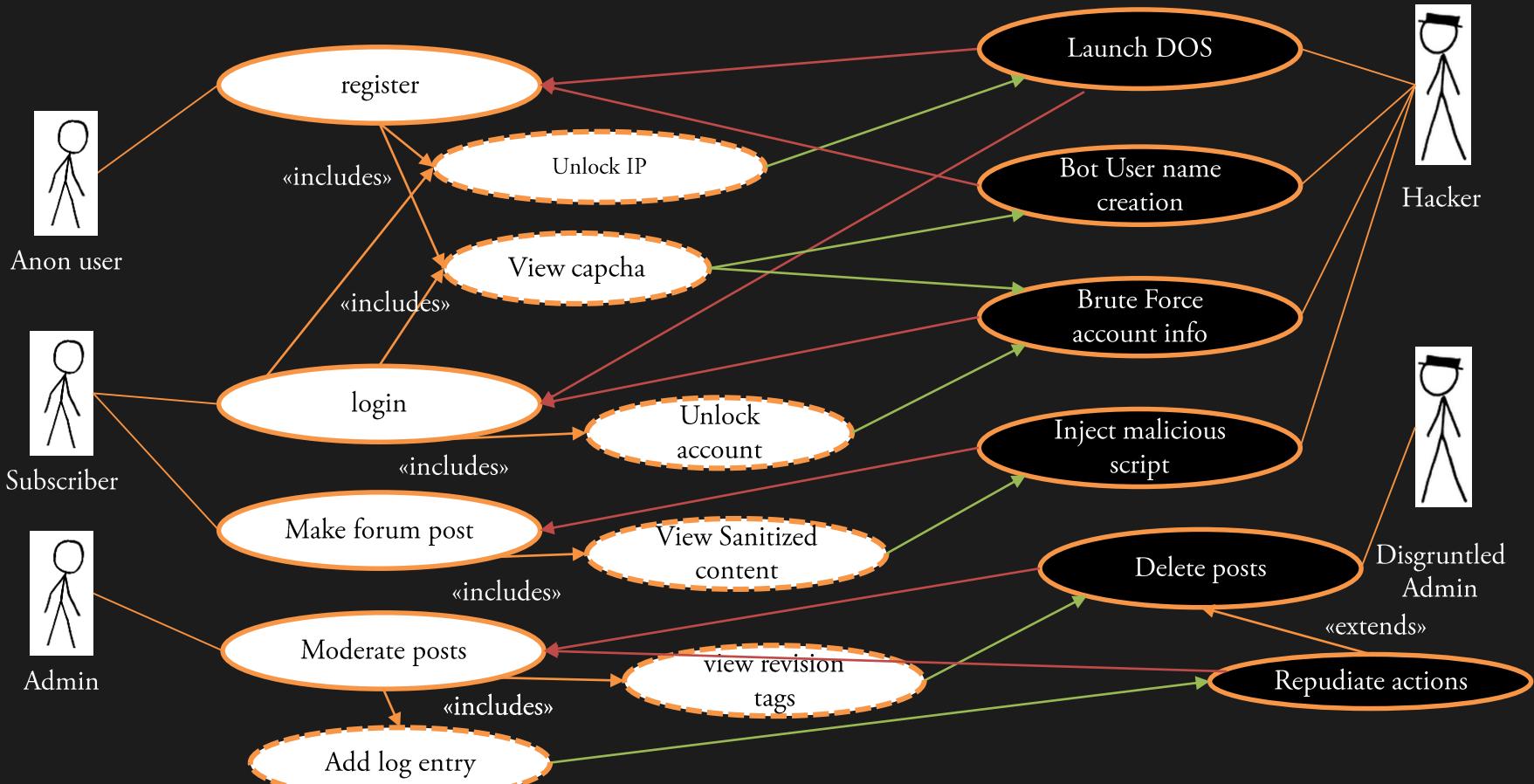


Ok... how do developers make architecture diagrams?

- They use requirements, use cases, and user stories to drive component formation and connection

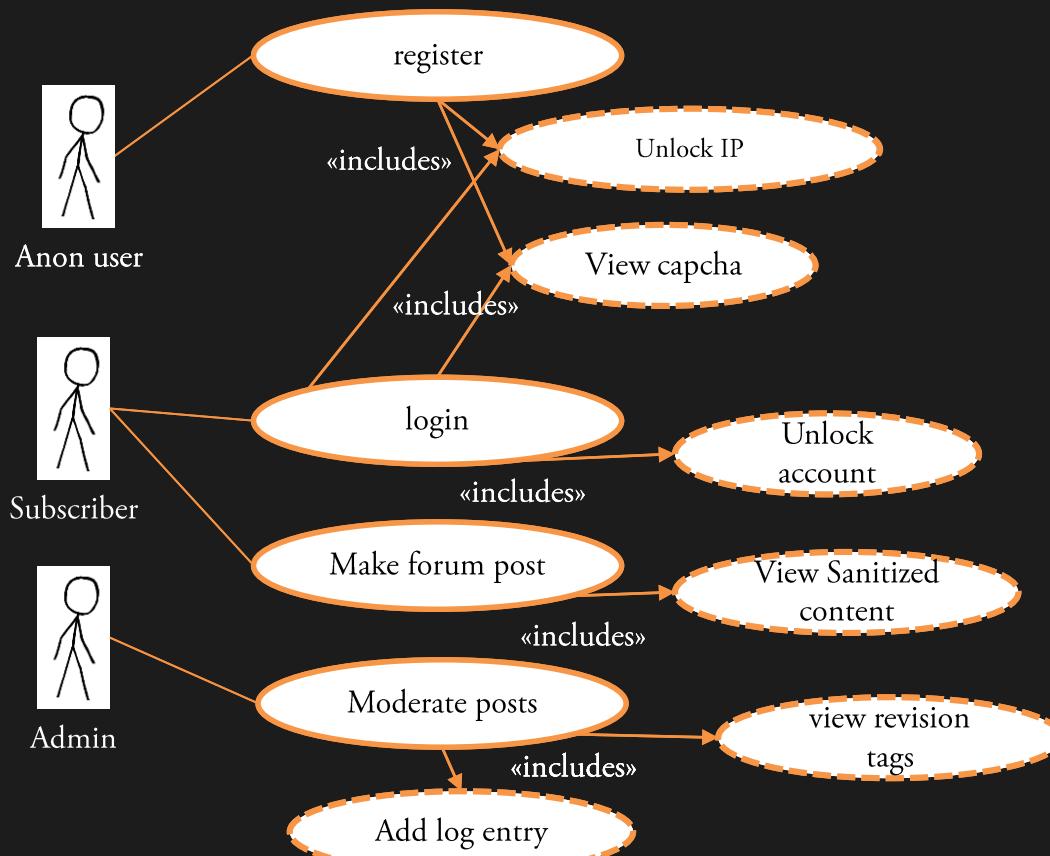
Returning to our system example (a forum)

Translate use cases to components



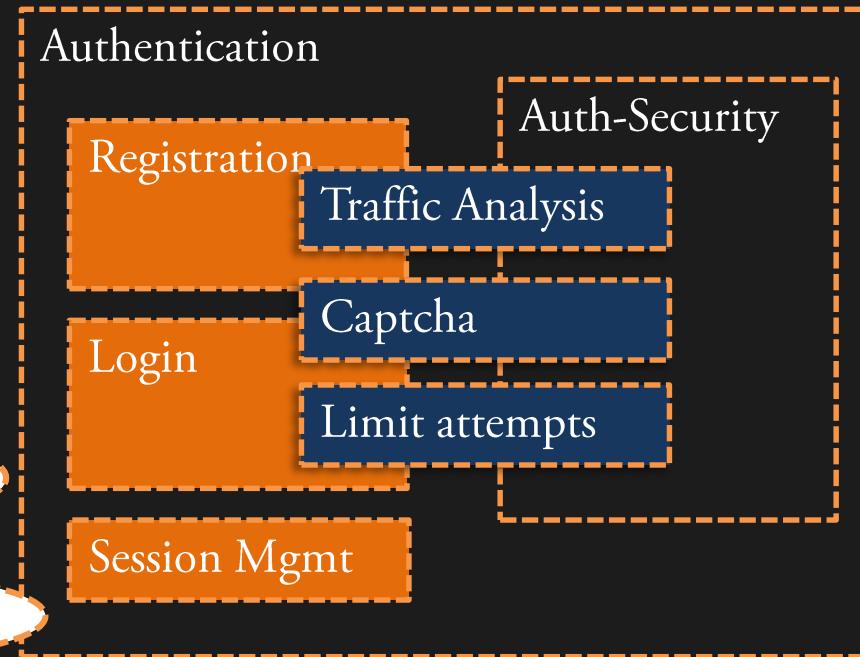
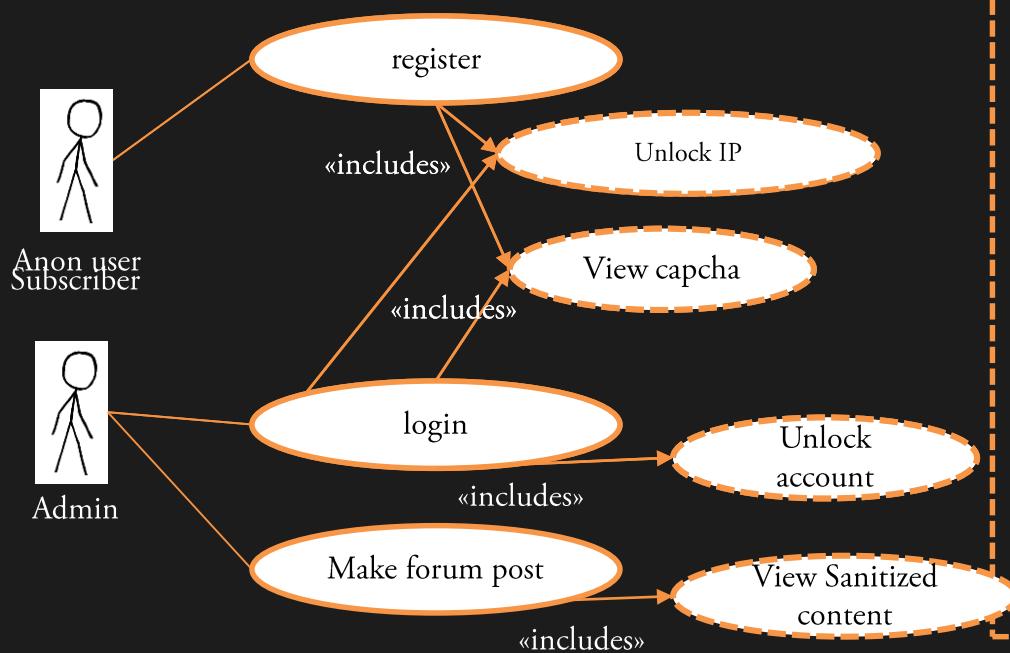
Returning to our system example (a forum)

Translate use cases to components

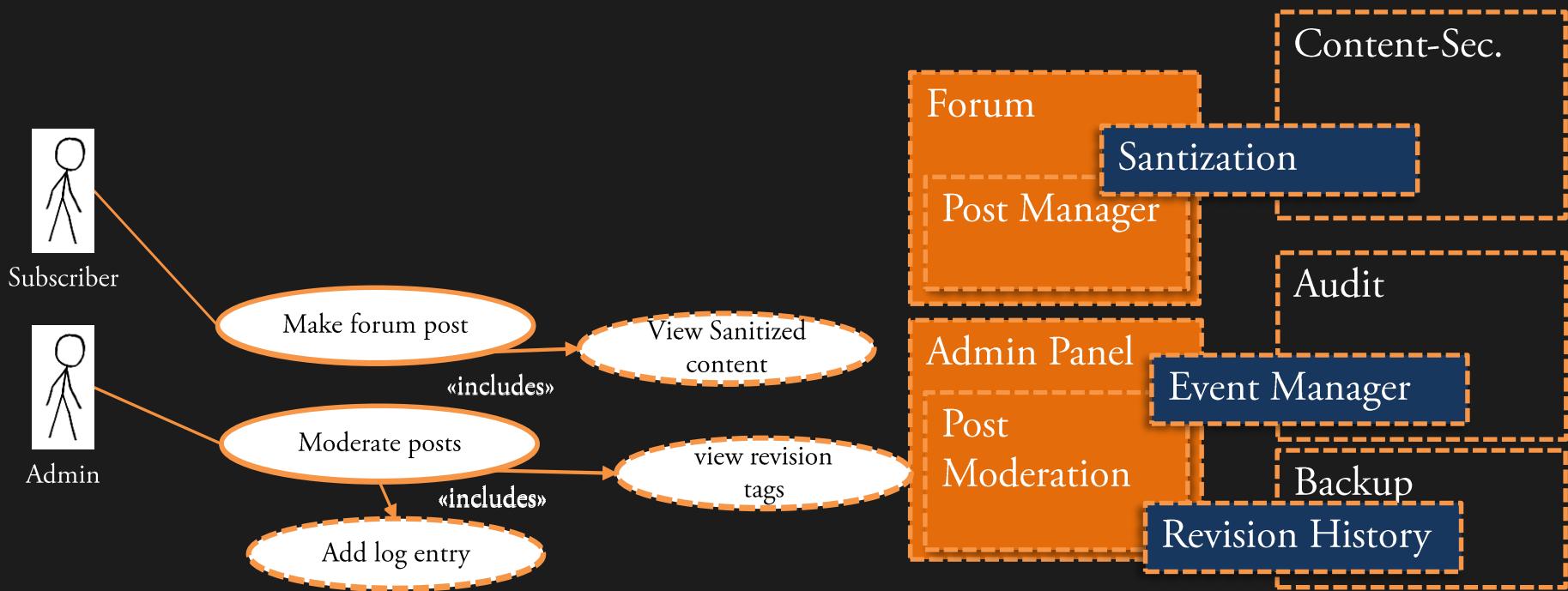


Functional security requirements are realized by building components

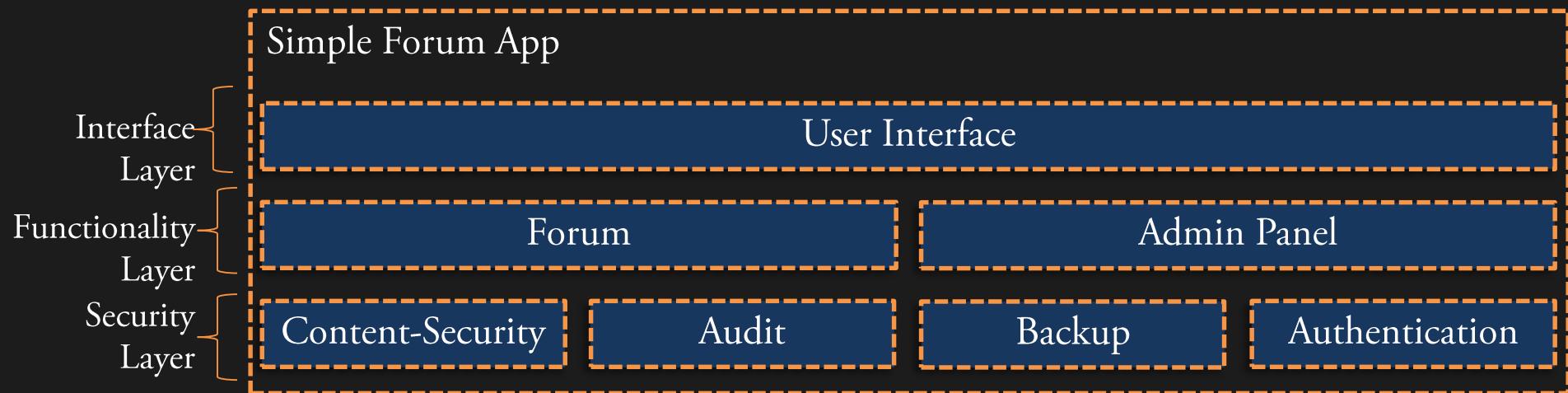
Mentioned this before, now it makes more sense.



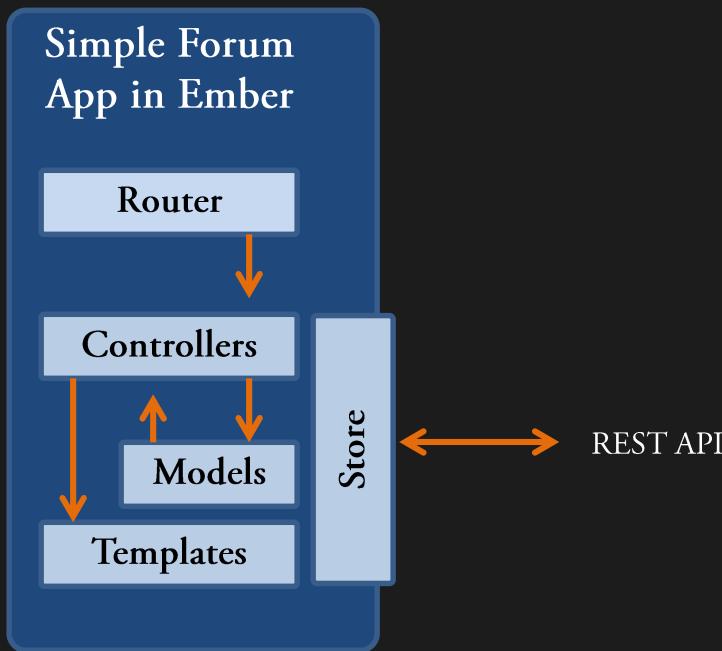
Functional security requirements are realized by building components

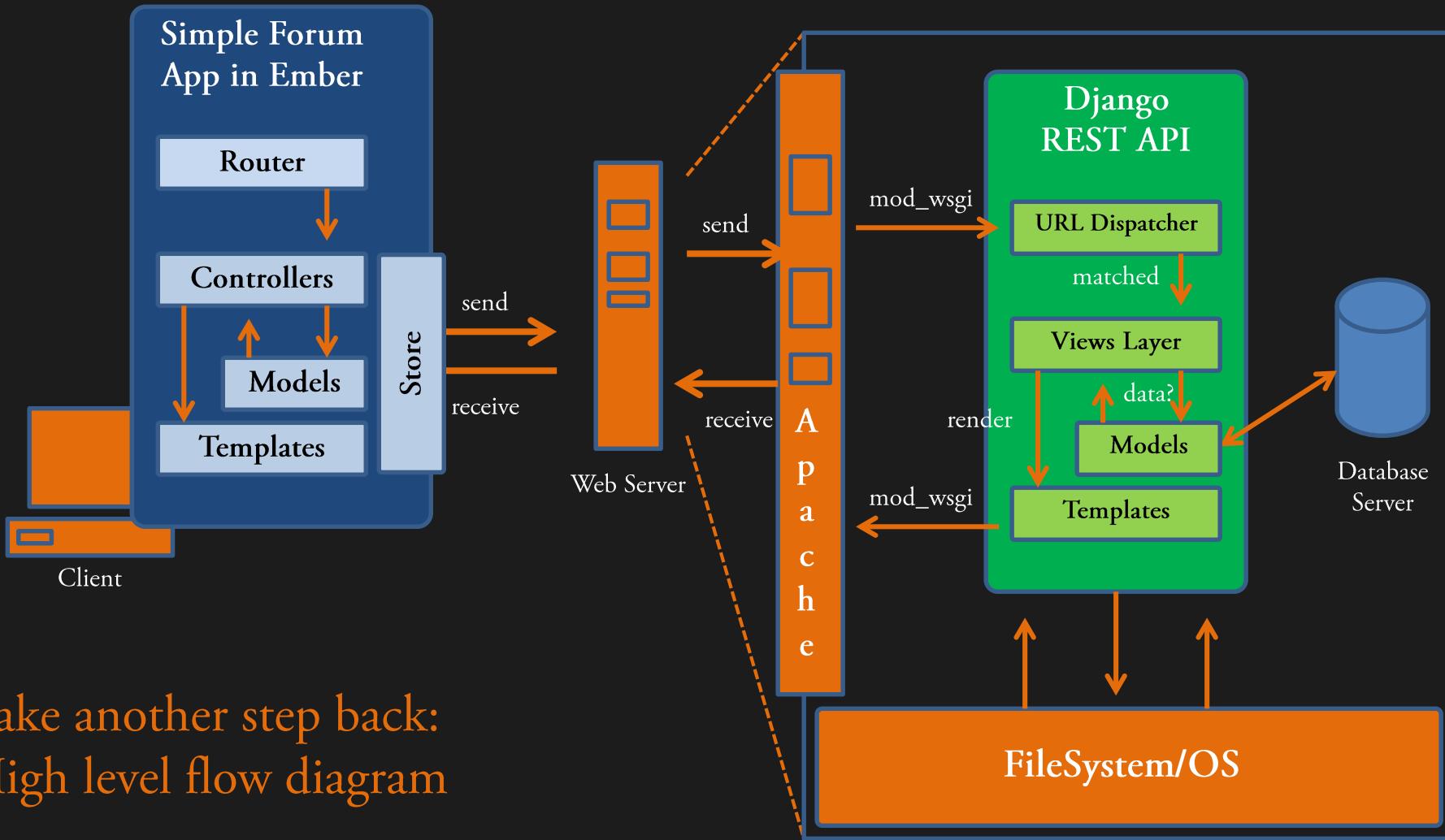


Take a step back: Application level (layered) diagram



Take another step back: Framework Level diagram





Take another step back:
High level flow diagram

The point

An architecture isn't limited to one diagram.

Each (use cases, application level, high level, etc) has its purposes.

Types of Diagrams

Informal Graphical Modeling

- Easy to represent ideas and structure conceptually
- Low learning curve, easy to use
- Most architectures are this

Formal

UML

- Easy-moderate to represent ideas and structure
- Moderate learning curve, harder to use (correctly)
- Provides standard parsability and automatic processing
- A lot of architectures are this

Others (e.g. Wright, Cross-UNITY, etc)

- moderate-hard based on language type
- Huge learning curves
- Potential for formal proofs of properties on systems
- Few architectures are this

Types of Diagrams: Informal Graphical Modeling

Generally produced in tools like Visio, PowerPoint, OmniGraffle, Lucidchart, etc

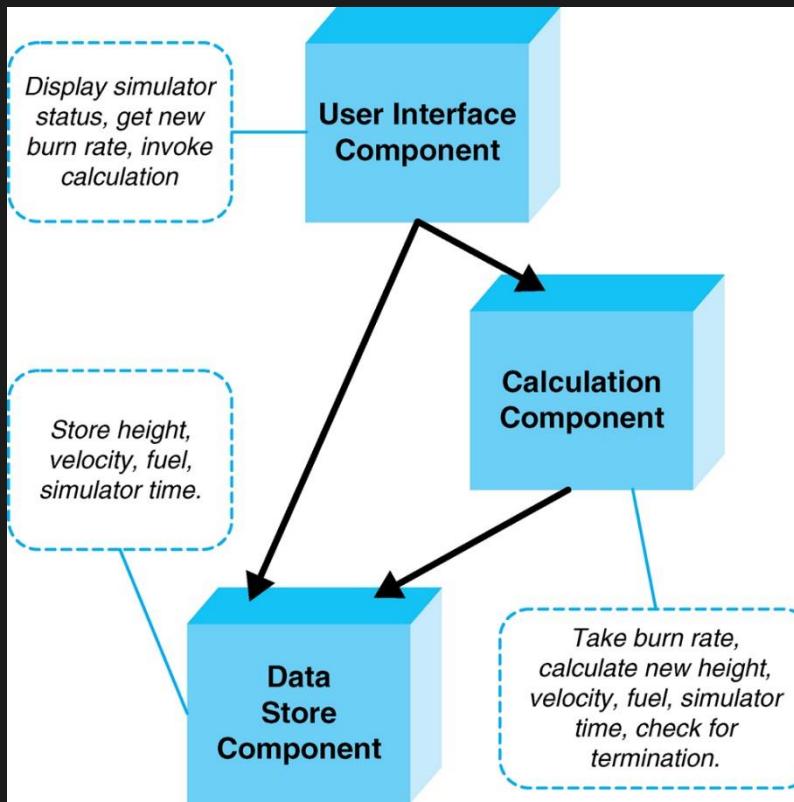
Advantages

- Can be aesthetically pleasing
- Size limitations (e.g., one slide, one page) generally constrain complexity of diagrams
- Extremely flexible due to large symbolic vocabulary
- Low to no learning curve

Disadvantages

- Ambiguous, non-rigorous, informal
 - But often treated otherwise
- Cannot be effectively processed or analyzed by machines/software

Types of Diagrams: Informal Example



Types of Diagrams: UML (Unified Modeling Language)

13 loosely-interconnected notations called diagrams that capture static and dynamic aspects of software-intensive systems

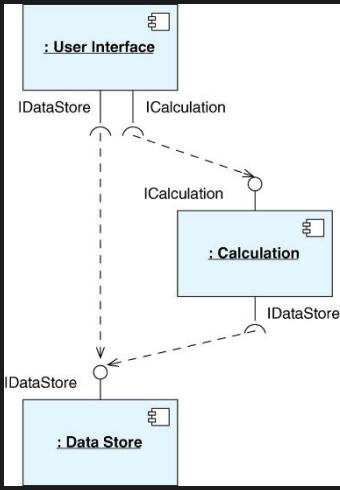
Advantages

- Support for a diverse array of viewpoints focused on many common software engineering concerns
- Ubiquity improves comprehensibility
- Extensive documentation and tool support from many vendors
- High precision and accuracy, minimal ambiguity

Disadvantages

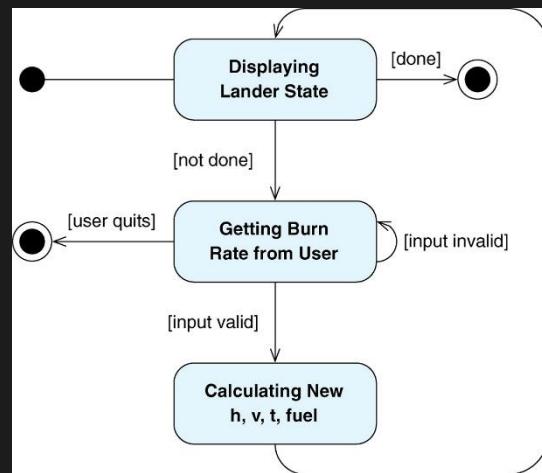
- Needs customization through profiles to reduce ambiguity
- Difficult to assess consistency among views
- Difficult to capture foreign concepts or views
- Medium-high learning curve

Types of Diagrams



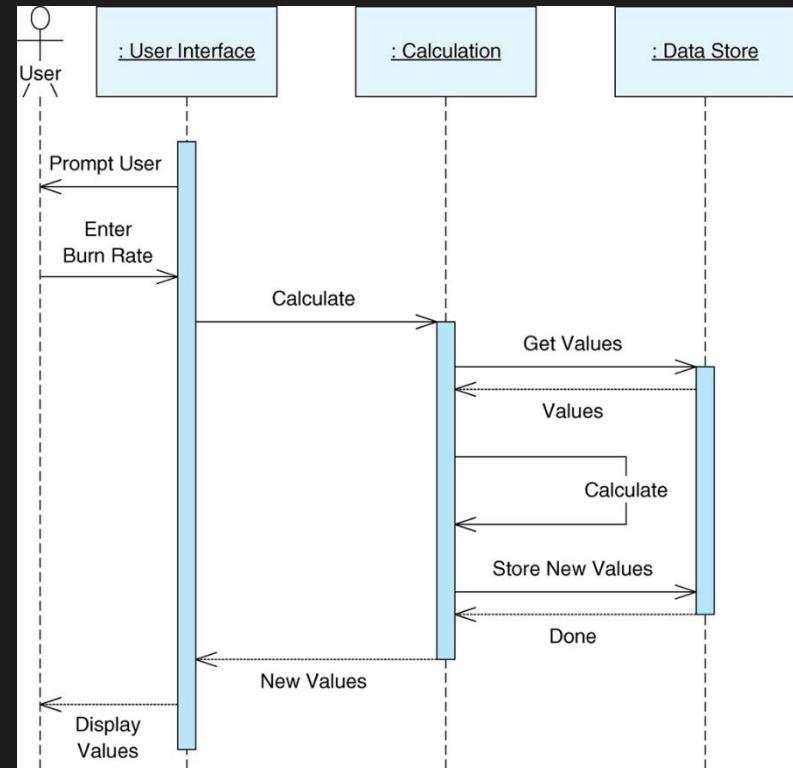
Component Diagram

Activity Diagram



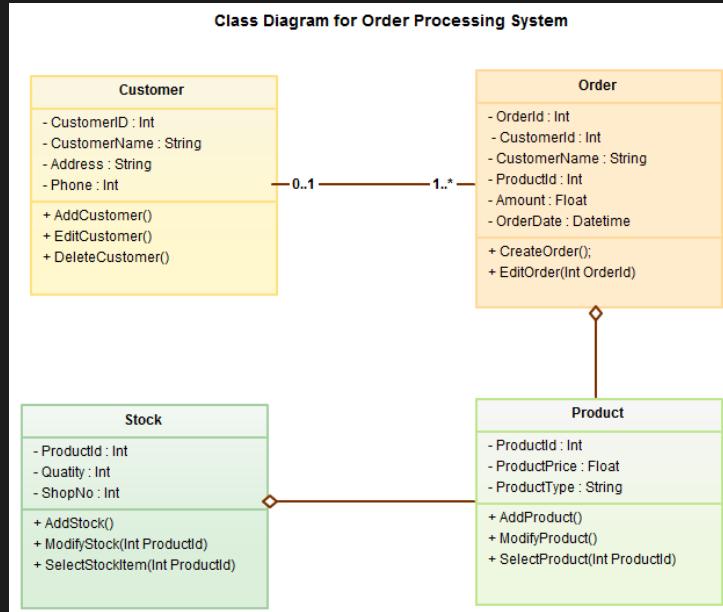
UML Examples

Sequence Diagram



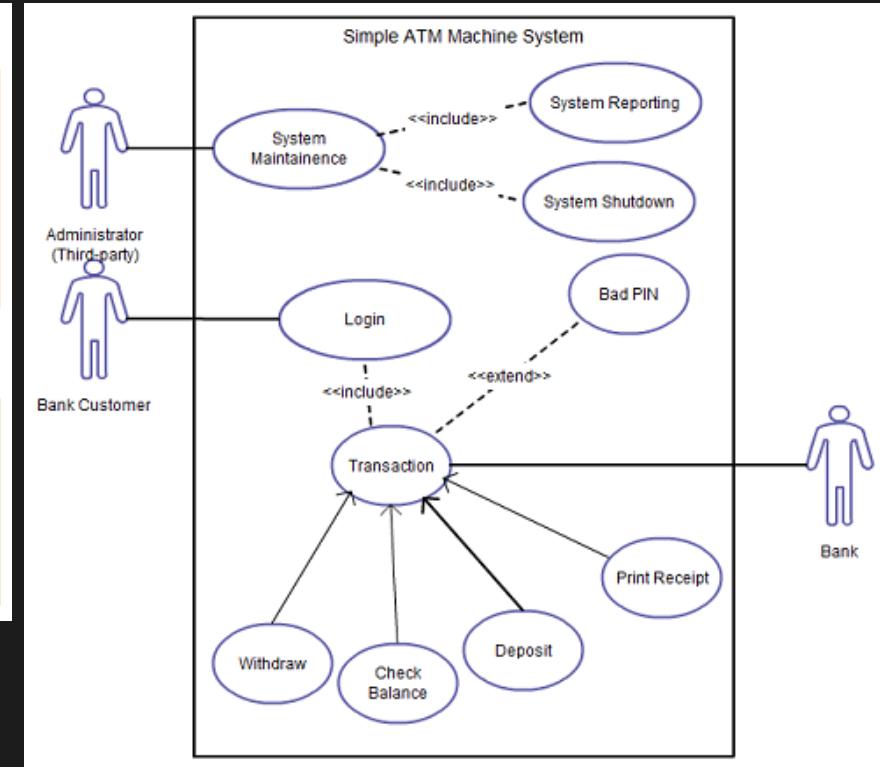
Types of Diagrams

Class Diagram



UML Examples

Use Case Diagram



Other types

Some of My Diagrams

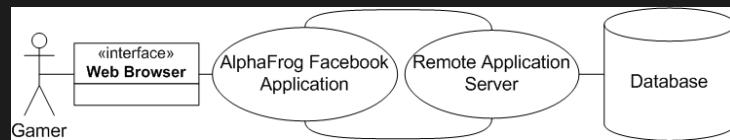
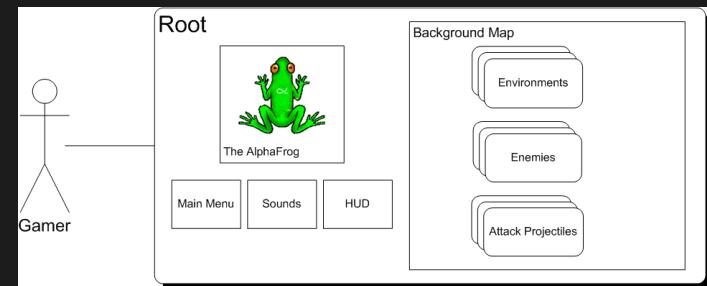
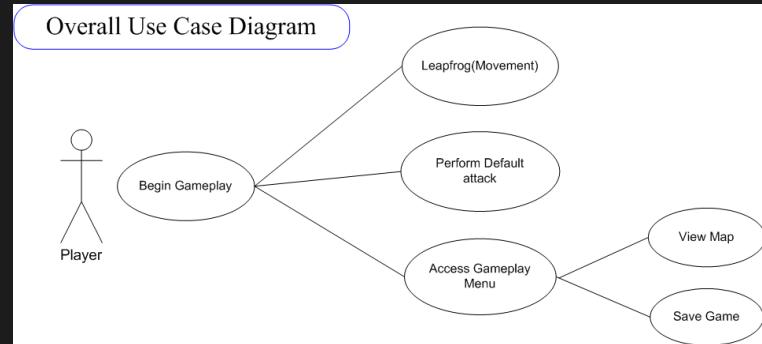
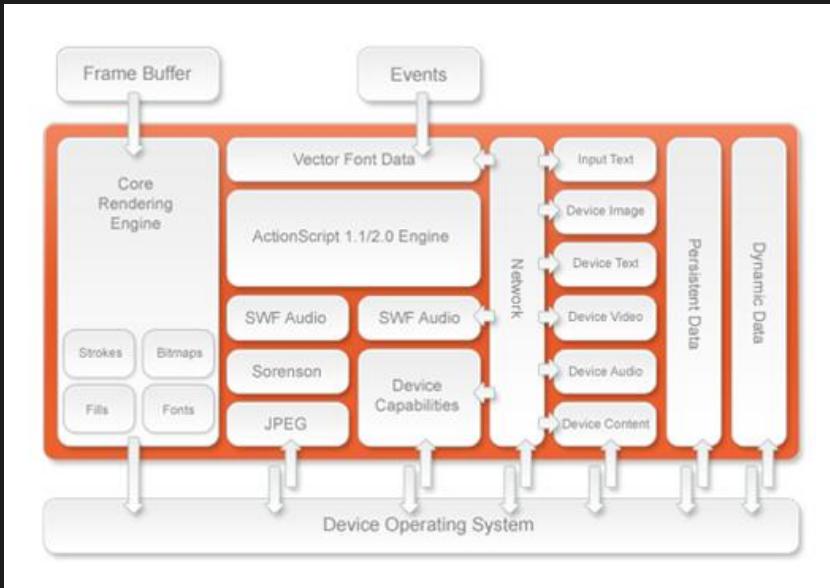
Example 1 – AlphaFrog (facebook connect)

Rad bar to
level up



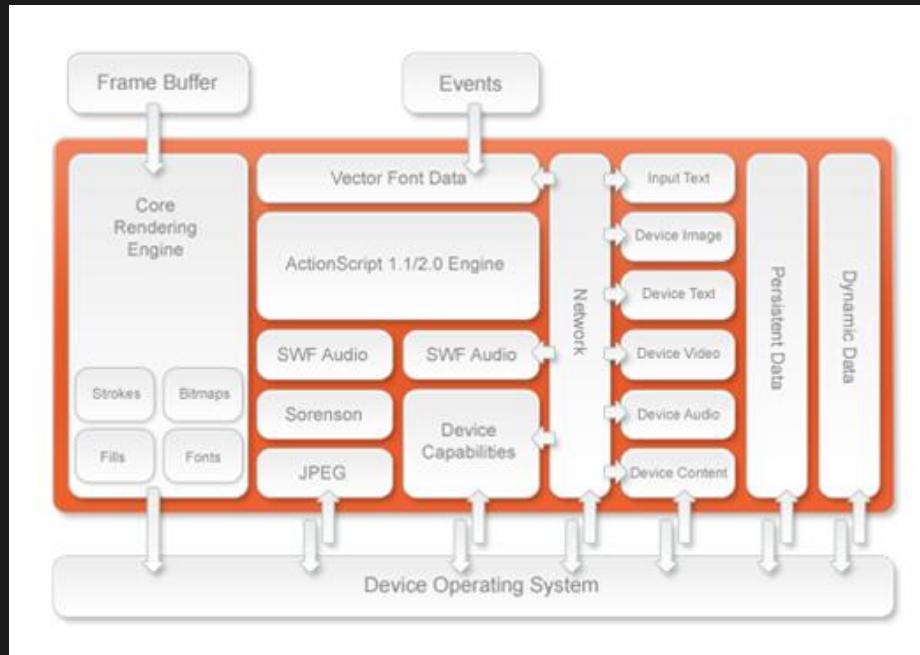
Real Diagrams

Example 1 – AlphaFrog (facebook connect)



Real Diagrams

Example 2 – VisGA (Genetic algorithm in flash)



<http://www.personal.utulsa.edu/~matt-hale/vis-ga/GA.swf>

HUD

Text Feedback

GA algorithms

Crossover
Operators

Mutation
Operators

Population
Management

Selection
Methods

Pathfinding

Line
Generation

Line Drawing

Source and
Obstacle
Generation

Real Diagrams

Example 3 - Cybertrust



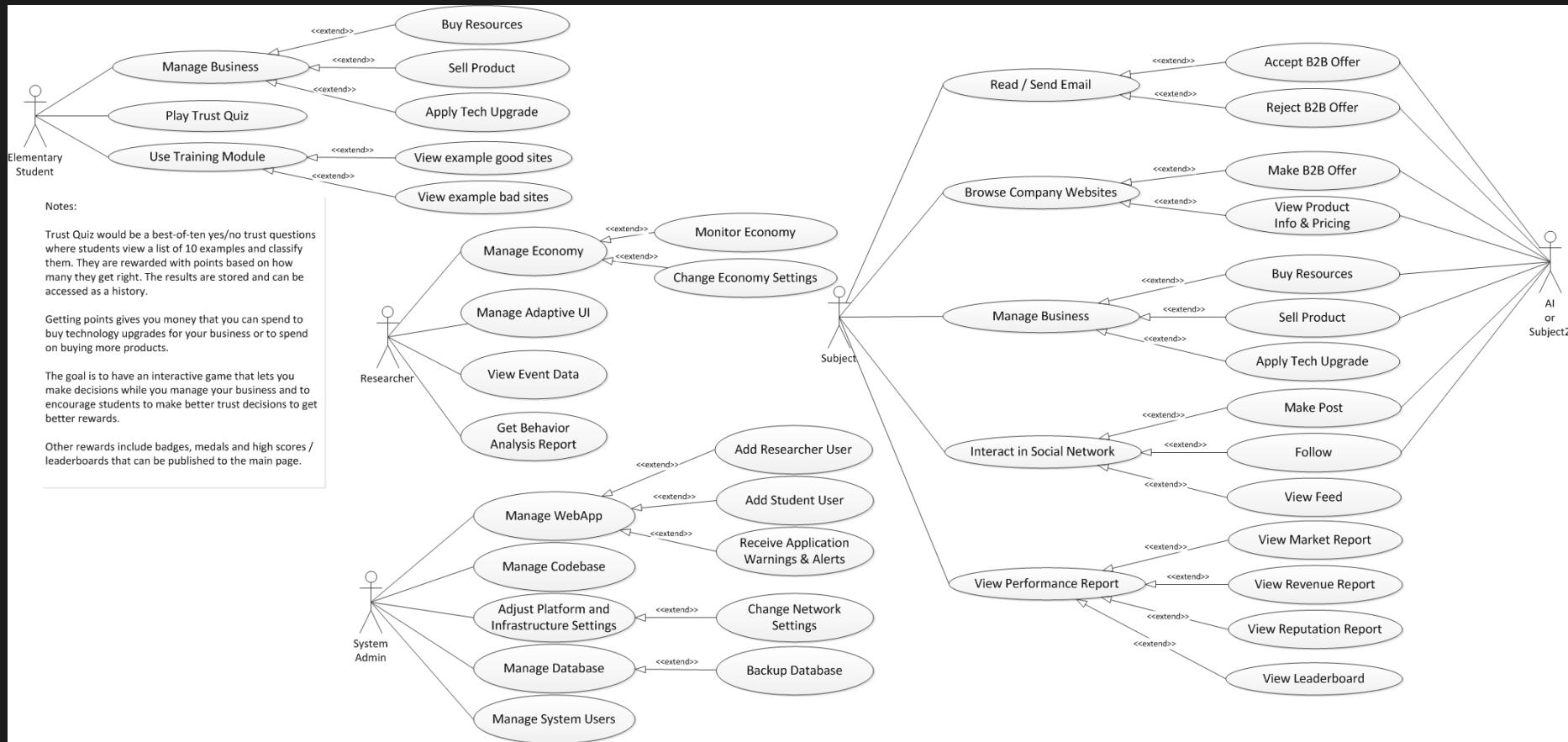
The screenshot shows a web-based email inbox interface for 'admin' (admin). The top navigation bar includes links for Home, Email (28), Web (3), bVerse Social Network (3), and Your status. On the right, there are options to Log out, Switch to Admin View, and Switch to Student View.

The left sidebar features a 'Inbox' section with a list of messages. The messages are listed in a table format:

	From	Subject	Date
<input type="checkbox"/>	bVerse	Account Compromised - Your bVerse account has been compromised. Please following the link below to upd...	10/17/13
<input type="checkbox"/>	Jimmy Smith	Join My Network - Jimmy Smith wants to add you as a friend on your social network. To accept his i...	12/16/13
<input type="checkbox"/>	Paul Bunyan Personnel Service	Temporary Lumberjack Agency - I would love to sell you some lumber. Sincerely, Paul Bunyan	Jan 1
<input type="checkbox"/>	Ann Howard	Inventory Control - You are running low on nails. Would you like me to purchase more from the same v...	Jan 1
<input type="checkbox"/>	PaperMe	Corporate Friend Request - I'd like to invite you to join my social network.-Paul BunyanAccept invitation...	Jan 7
<input type="checkbox"/>	bVerse	Account Compromised - Your bVerse account has been compromised. Please following the link below to upd...	Jan 12
<input type="checkbox"/>	Jimmy Smith	Join My Network - Jimmy Smith wants to add you as a friend on your social network. To accept his i...	Jan 12
<input type="checkbox"/>	test	test - test	Jan 12
<input type="checkbox"/>	asdf	asdf - asdfdsafasfasdfasfasdfasfasdf	Jan 20
<input type="checkbox"/>	asdf	asdf - asdfdsafasfasdfasfasdfasfasdf	Jan 20
<input type="checkbox"/>	asdf	asdf - asdfdsafasfasdfasfasdfasfasdf	Jan 20
<input type="checkbox"/>	asdf	asdf - asdfdsafasfasdfasfasdfasfasdf	Jan 20
<input type="checkbox"/>	bVerse	Account Compromised - Your bVerse account has been compromised. Please following the link below to upd...	Jan 20
<input checked="" type="checkbox"/>	me, asdf (2)	asdf - asdfdsafasfasdfasfasdfasfasdf	Jan 20
<input type="checkbox"/>	asdf	asdf - asdfdsafasfasdfasfasdfasfasdf	Jan 21
<input type="checkbox"/>	Woodchuck Tool and test	Elevate Your Cutting - Woodchuck, Timberjack, and Groundhog are offering great savings on all products ...	Jan 1
<input type="checkbox"/>	Bob Smith	Check out our new online catalog - Hope you had a great holiday, I know I did. We would like you to check out our i...	12/27/13
<input type="checkbox"/>	Doug Banks	Deal of the month - Congratulations! Wally's is offering you only a deal of 30% off all orders over ...	12/30/13
<input type="checkbox"/>	Your friend	Left your credit card - You left your credit card at Wal-Mart. Go to my website http://helpyouforreal.c...	Jan 13
<input checked="" type="checkbox"/>	Your friend	Left your credit card - You left your credit card at Wal-Mart. Go to my website http://helpyouforreal.c...	Jan 12
<input type="checkbox"/>	bVerse	Account Compromised - Your bVerse account has been compromised. Please following the link below to upd...	Jan 12
<input checked="" type="checkbox"/>	me, Jimmy Smith (2)	Join My Network - Jimmy Smith wants to add you as a friend on your social network. To accept his i...	Jan 12
<input type="checkbox"/>	bVerse	Account Compromised - Your bVerse account has been compromised. Please following the link below to upd...	Jan 20
<input type="checkbox"/>	bVerse	Account Compromised - Your bVerse account has been compromised. Please following the link below to upd...	Jan 20
<input type="checkbox"/>	Jimmy Smith	Join My Network - Jimmy Smith wants to add you as a friend on your social network. To accept his i...	Jan 20

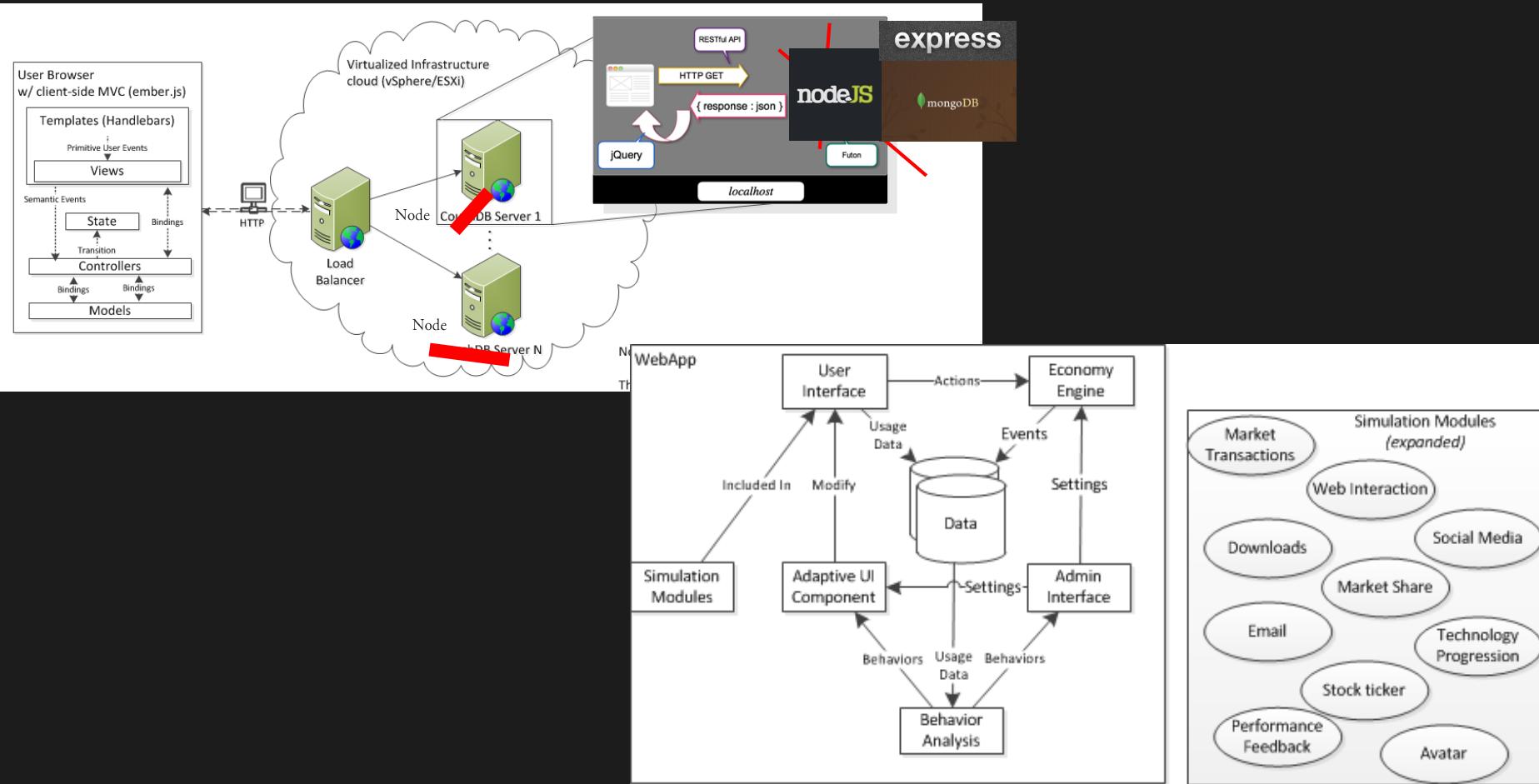
Real Diagrams

Example 3 - Cybertrust



Real Diagrams

Example 3 - Cybertrust





Questions?

Matt Hale, PhD

University of Nebraska at Omaha

Interdisciplinary Informatics

mlhale@unomaha.edu

Twitter: [@mlhale_](https://twitter.com/mlhale_)

