Software Architecture

- Definitions
  - http://www.sei.cmu.edu/architecture/published_definitions.html
  - ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems
    - Architecture is defined by the recommended practice as *the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.*

Architecture and Design

- Software design is often divided into two categories:
  - – Software architecture design
    - Top-level design, high-level software structure and organization of components
  - – Software detailed design
    - Describing each component sufficiently to allow for its construction

Testing

- Requirements -> Acceptance Testing
- Architecture -> System Testing
- Design -> Integration Testing
- Construction -> Unit Testing

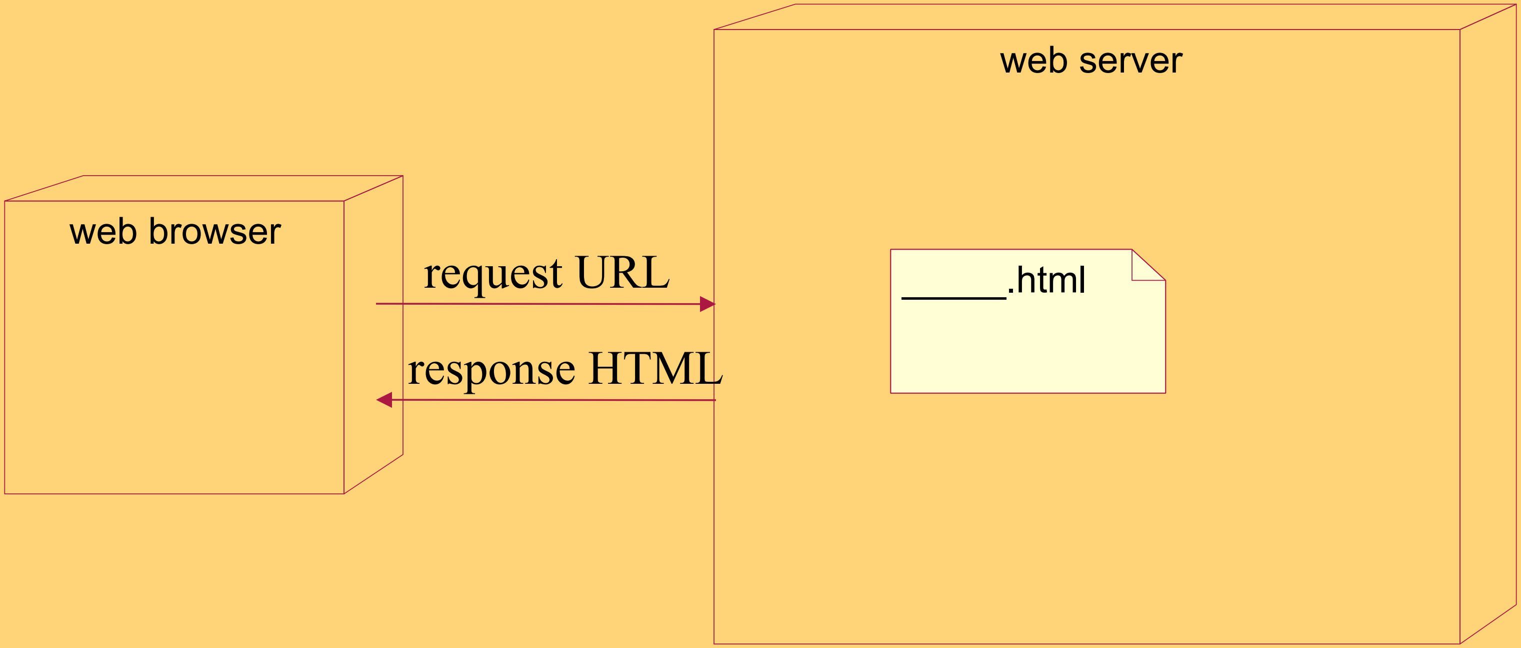
Design Strategies

- Divide-and-conquer/stepwise refinement
- Top-down vs. bottom-up
- Data abstraction and information hiding
- CRC Cards (Class-Responsibilities-Collaborators)
- Use of heuristics
- Use of patterns and pattern languages
- Iterative and incremental approach

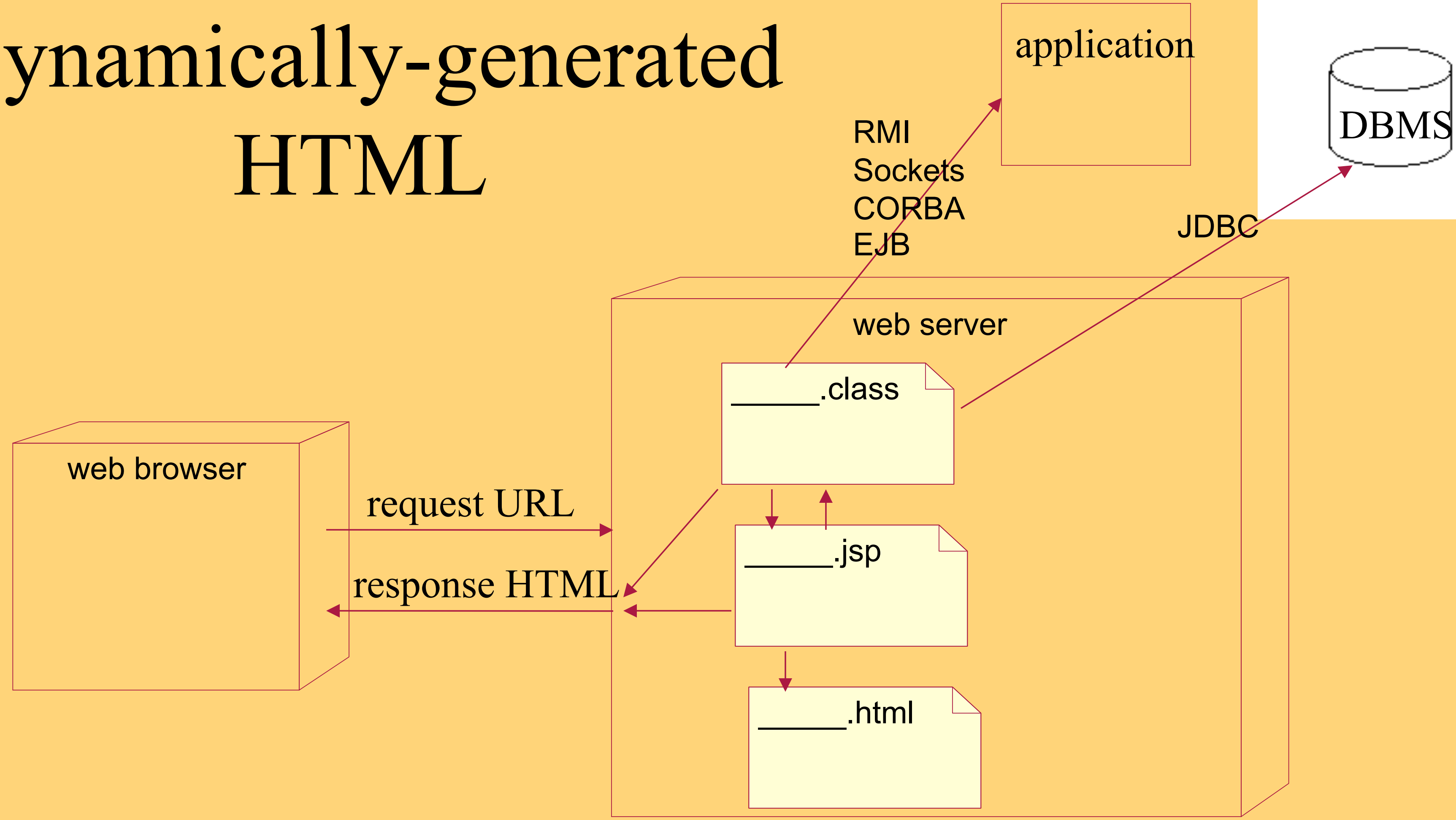
Web Architecture Overview

- Static html
  - text file containing html tags created manually
  - may include some client-side scripts (e.g. JavaScript)
- Dynamically generated html
  - html file produced at time of request
  - cgi, php, asp, jsp, Servlets
- Html with active content
  - html contains a program that runs at the client inside a web browser
  - Java applets, javascript

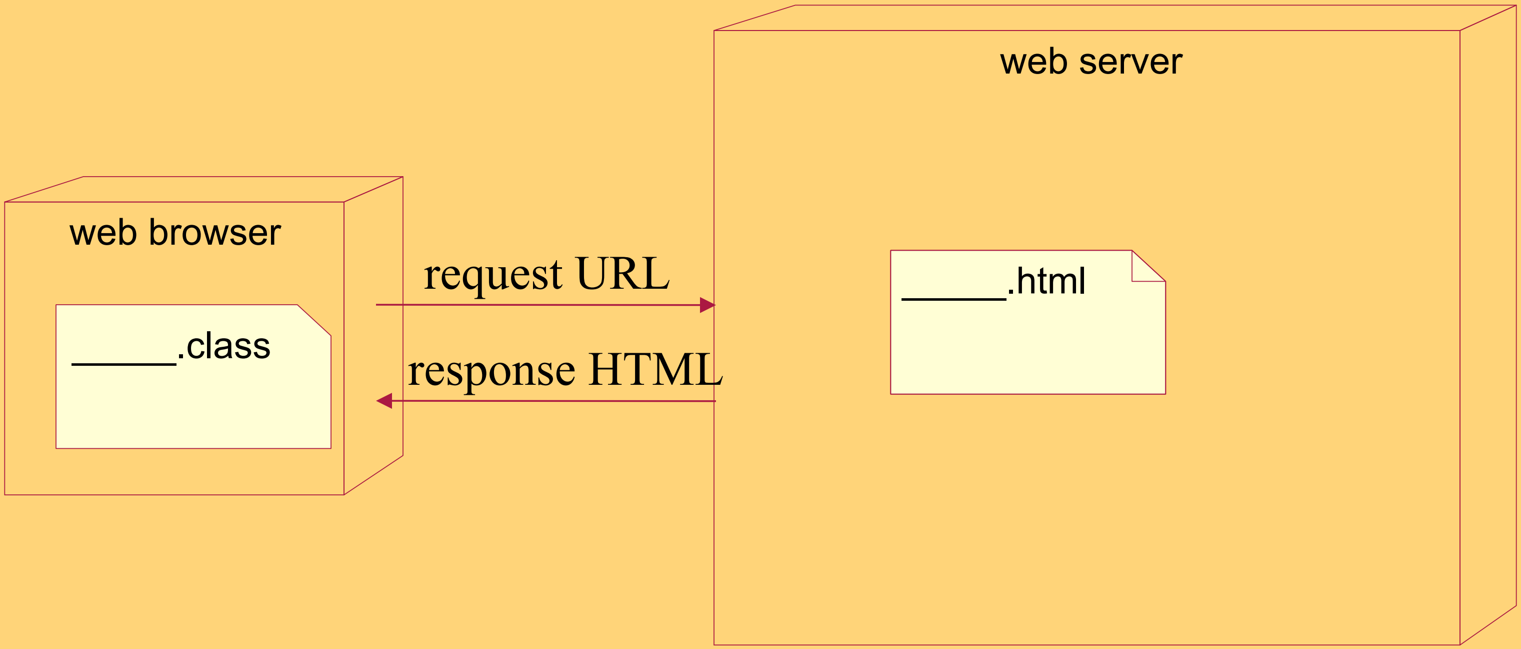
Static HTML



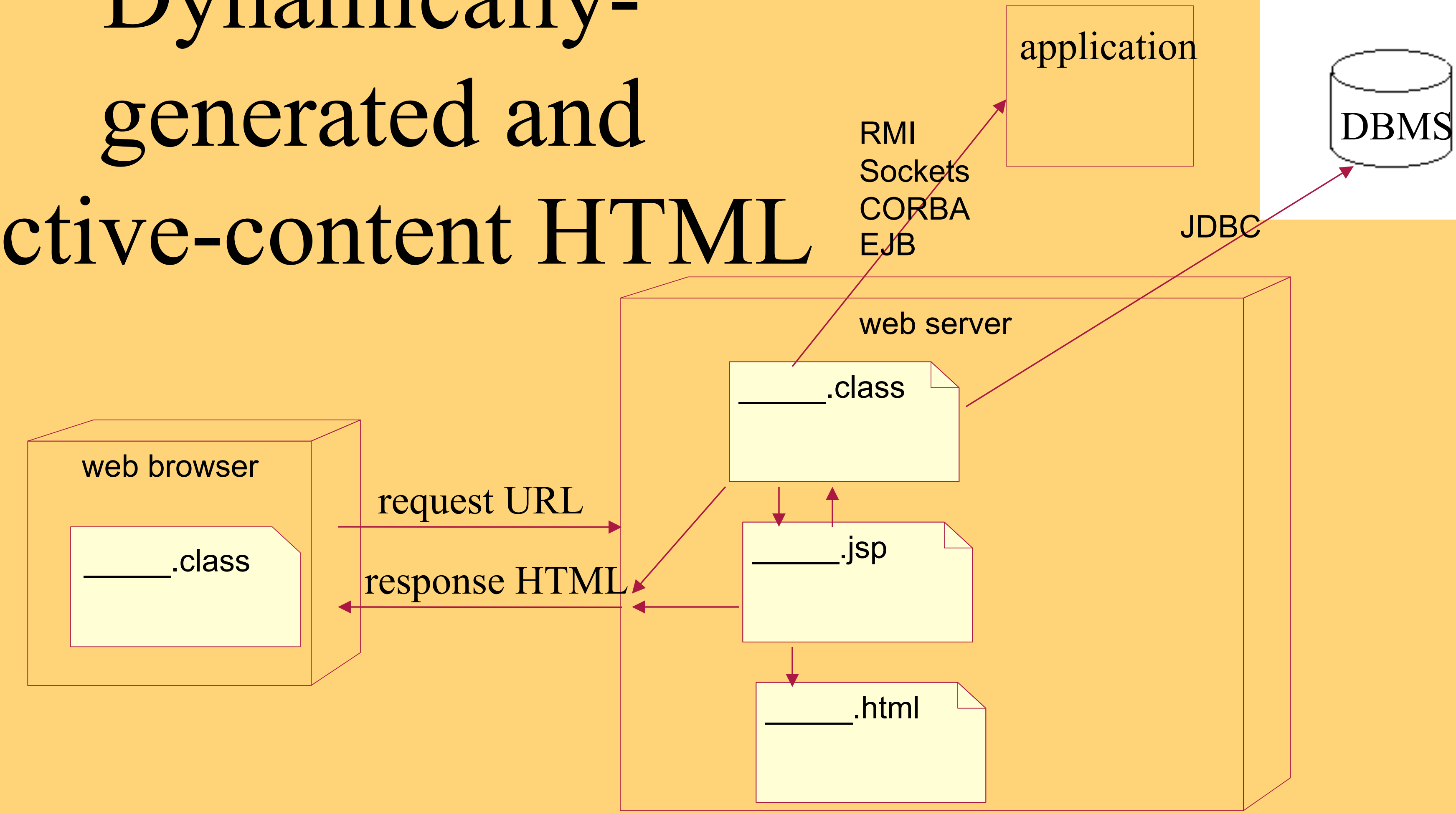
Dynamically-generated HTML



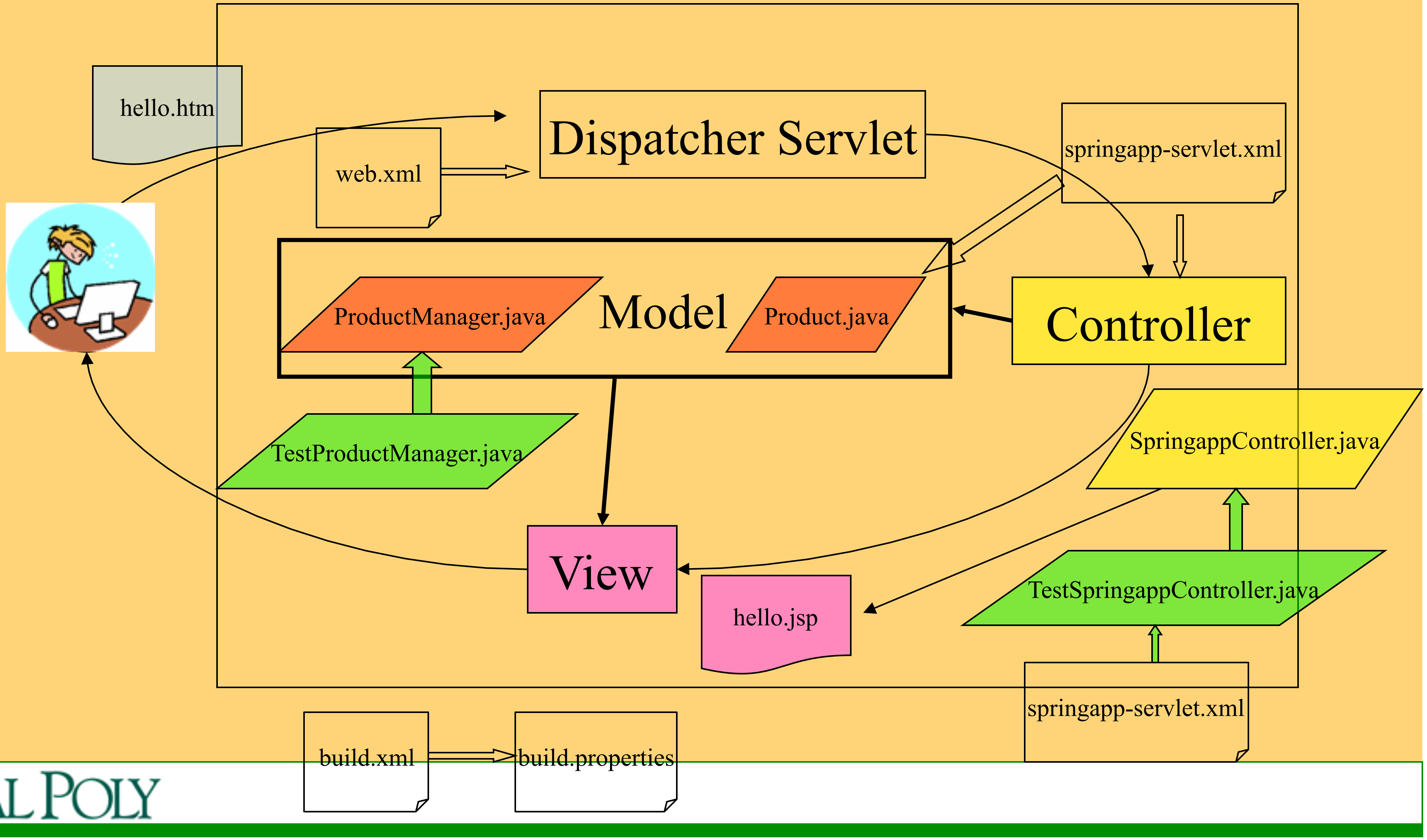
HTML with Active Content



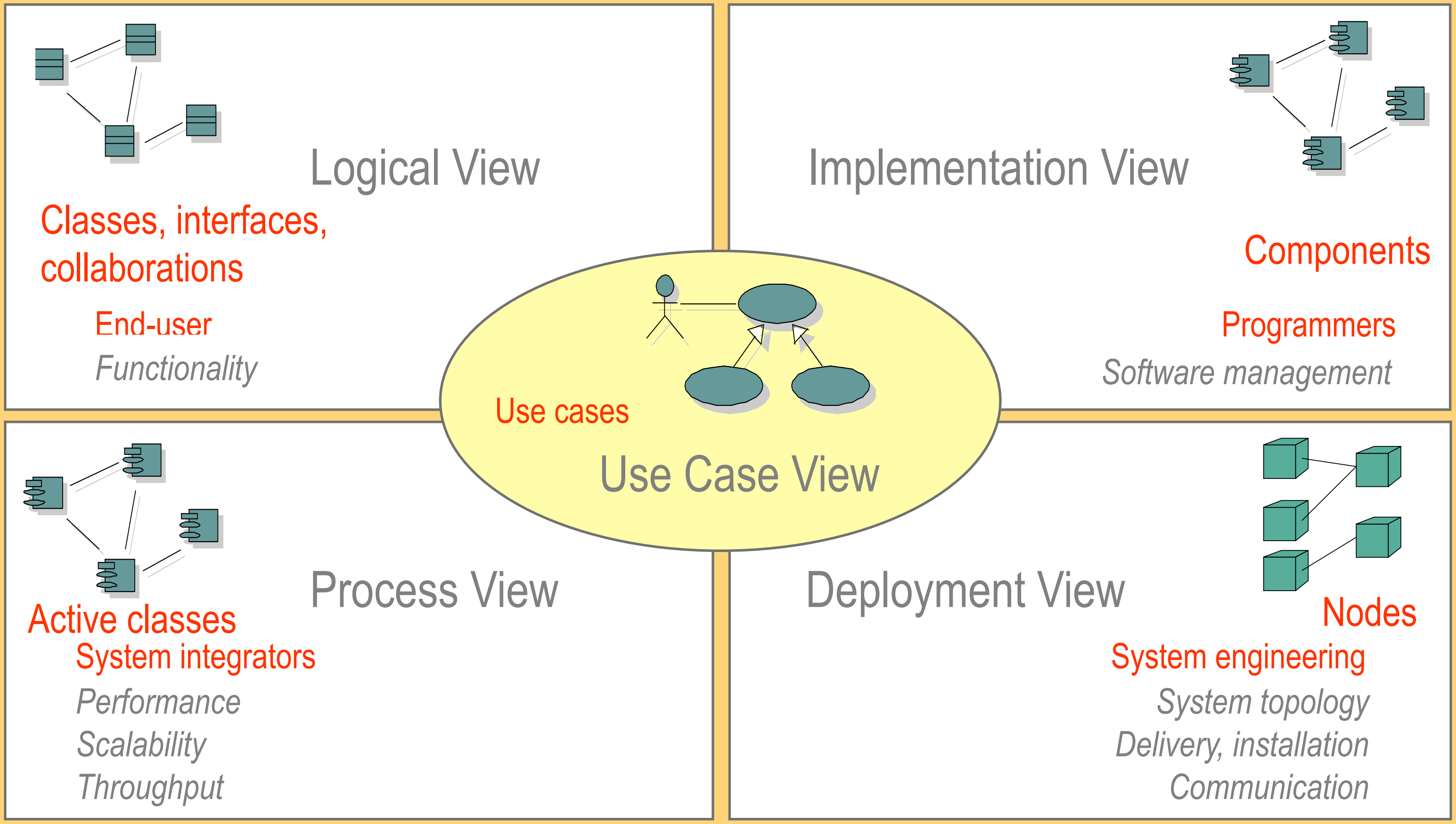
Dynamically-generated and Active-content HTML



Sample Web Architecture with Spring Framework



4+1 Architecture Views from the Unified Process



Logical View

Classes, interfaces, collaborations

End-user

Functionality

Implementation View

Components

Programmers

Software management

Use cases

Use Case View

Process View

Active classes

System integrators

Performance

Scalability

Throughput

Deployment View

Nodes

System engineering

System topology

Delivery, installation

Communication

Conceptual

Physical

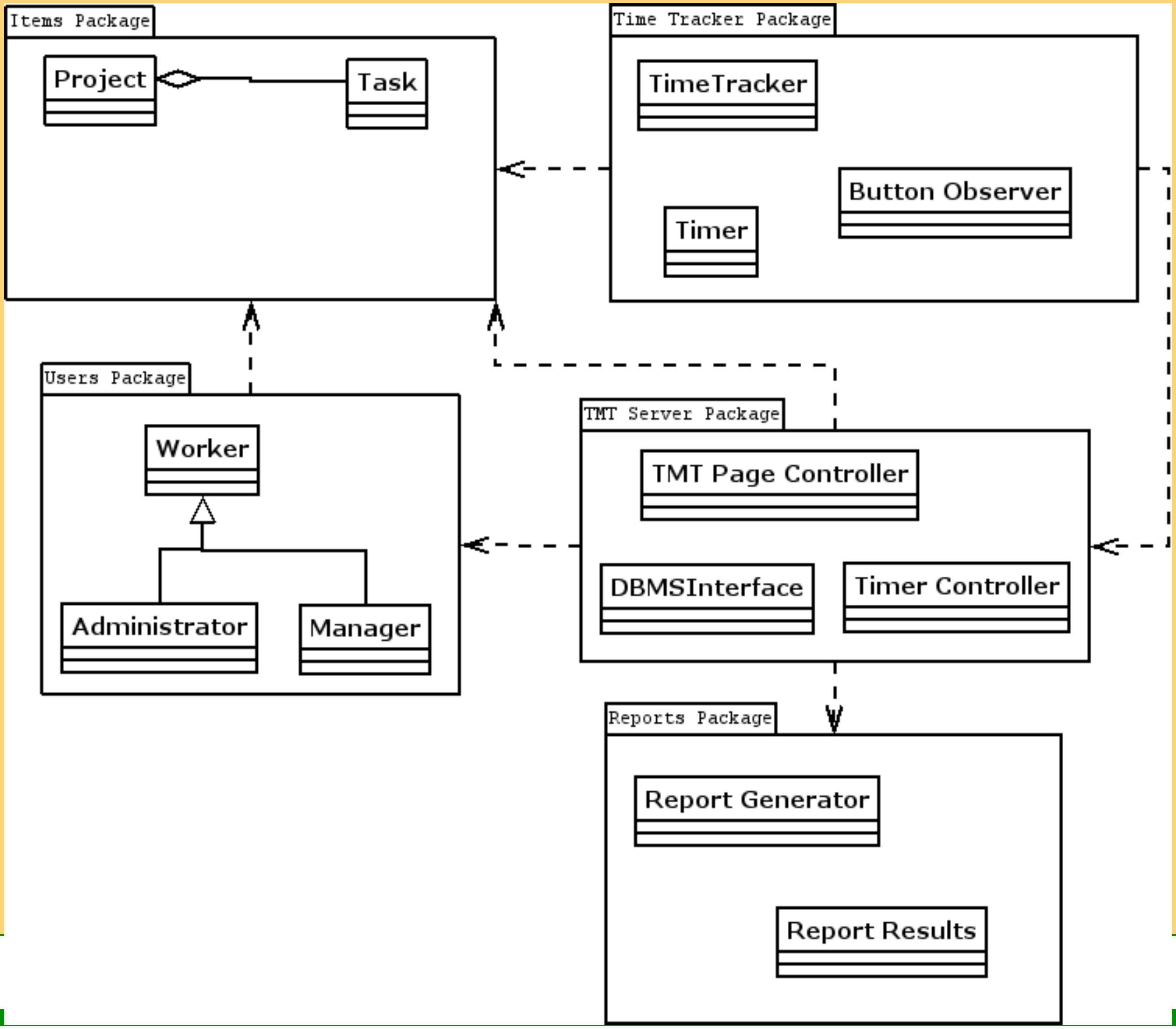
Modeling

- CASE
- Model->code-generation->source code
- Source code->object code
- Reverse engineering

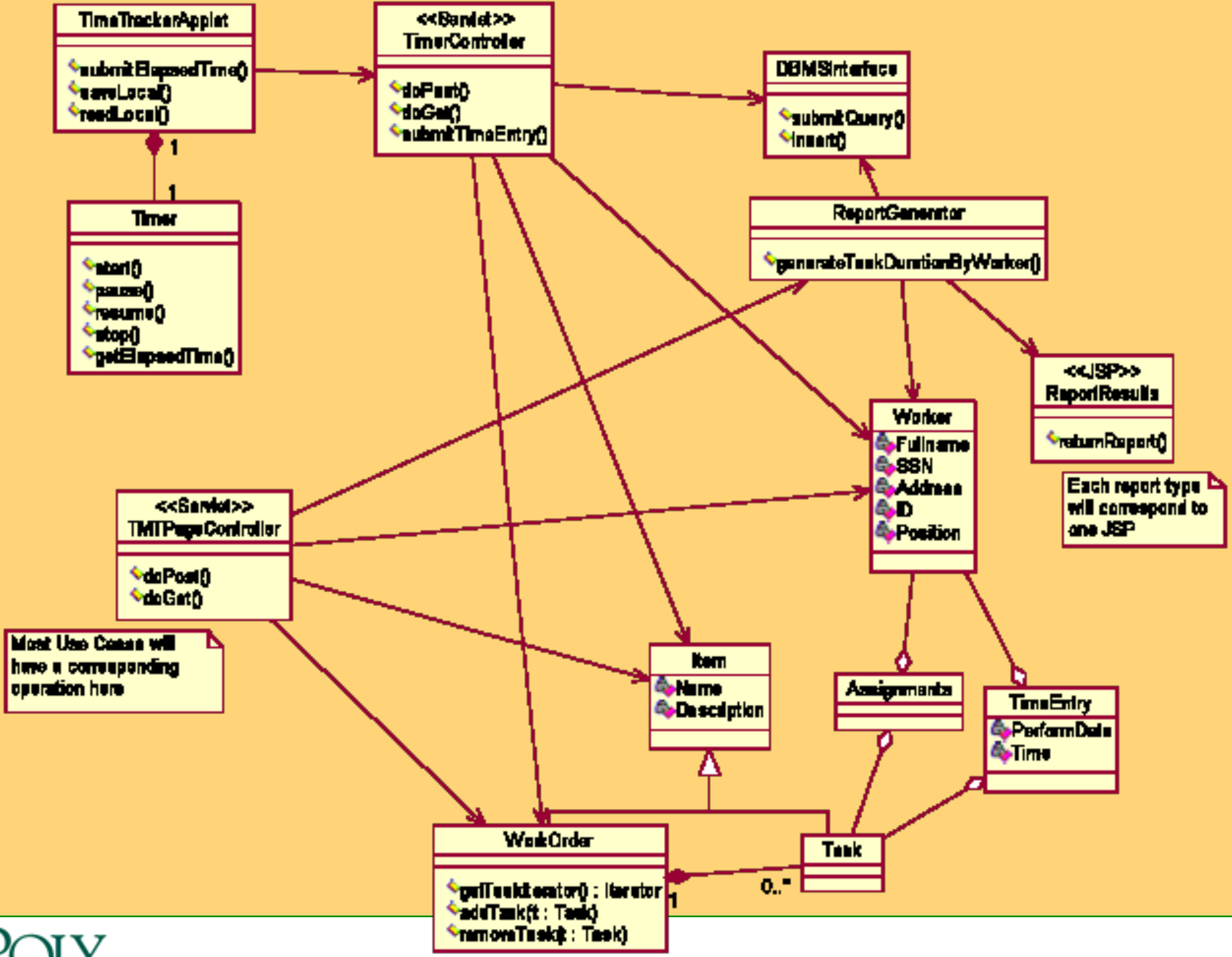
UML Diagrams

- Structural
  - Class, Object, Component, Deployment Diagrams
- Behavioral
  - Use-Case, Activity, Sequence, Communication/ Collaboration, Statechart Diagrams

Architecturally Significant Classes Organized by Package



Architecturally Significant Classes



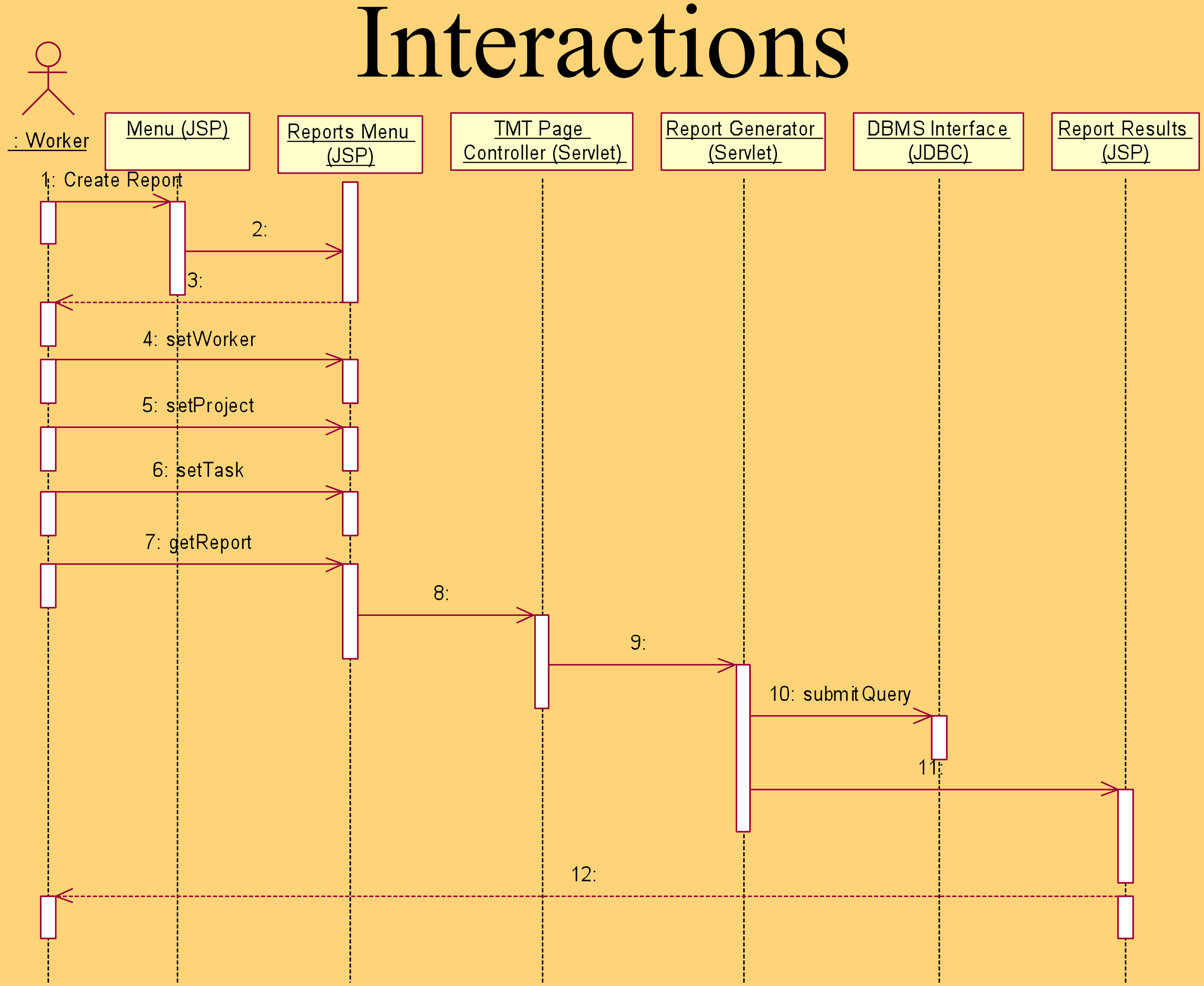
Each report type will correspond to one JSP

Most Use Cases will have a corresponding operation here

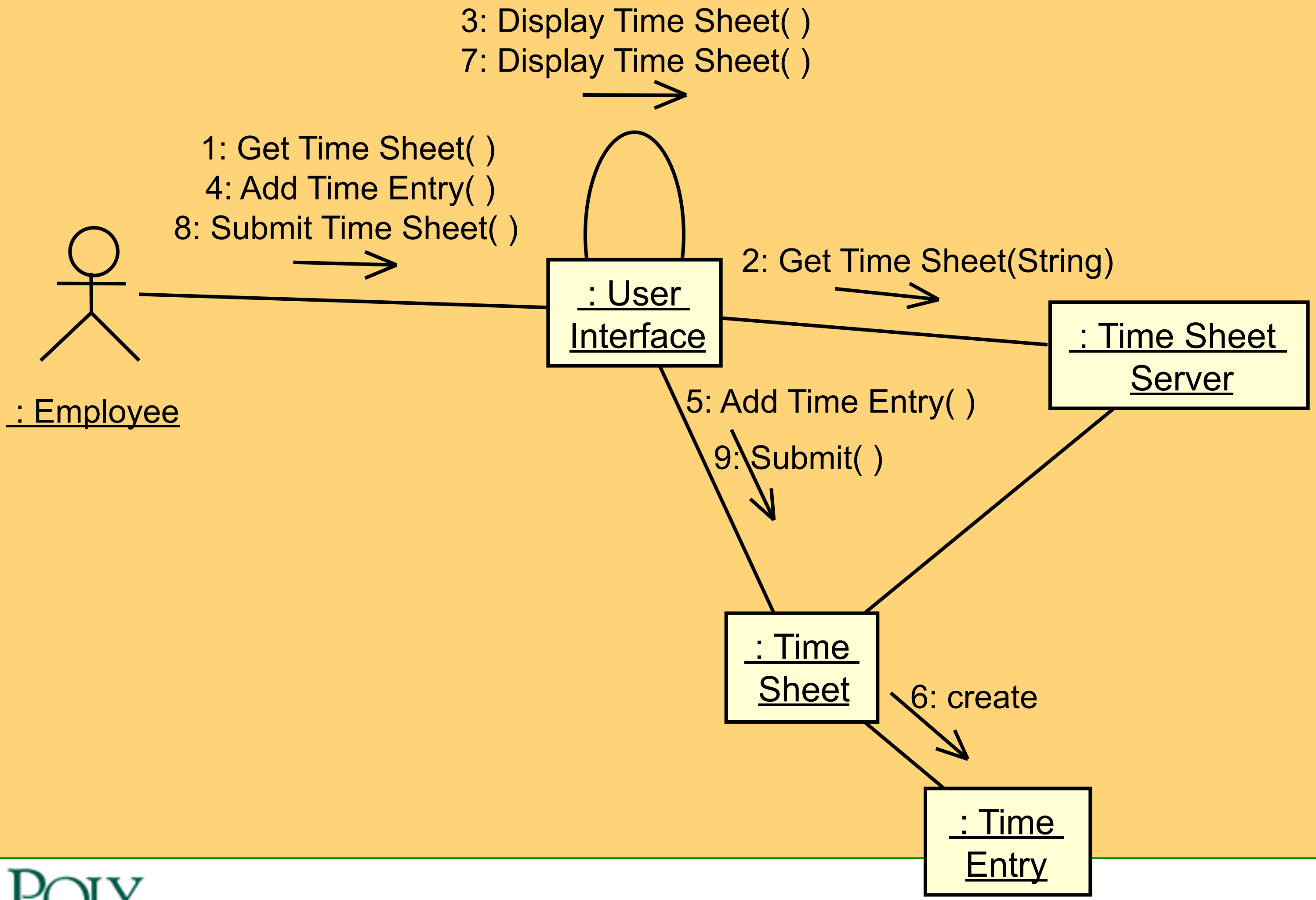
Class Diagrams

- Classes, attributes, operations
- Associations, aggregation, composition
- Inheritance/generalization

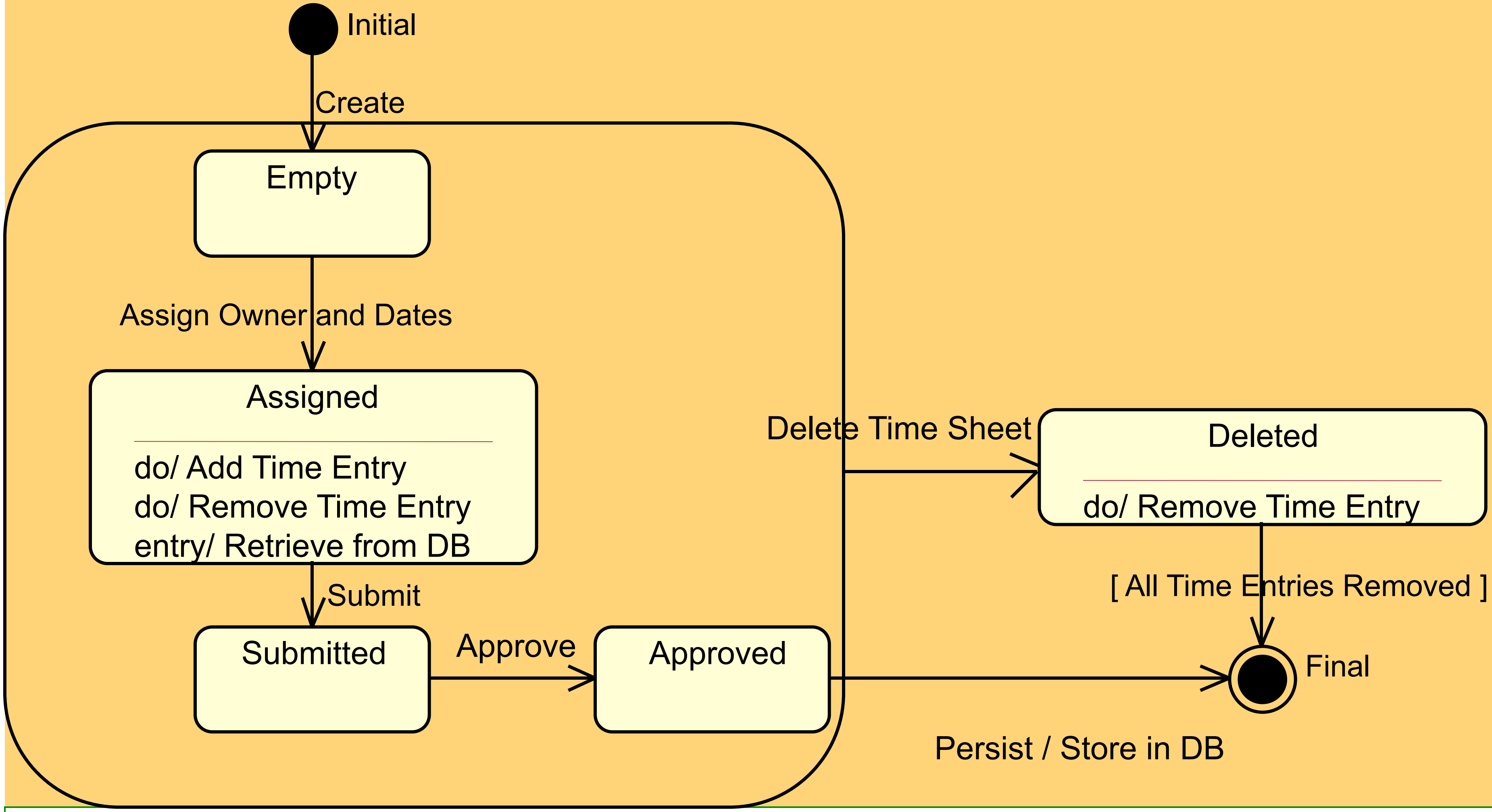
Architecturally Significant Interactions



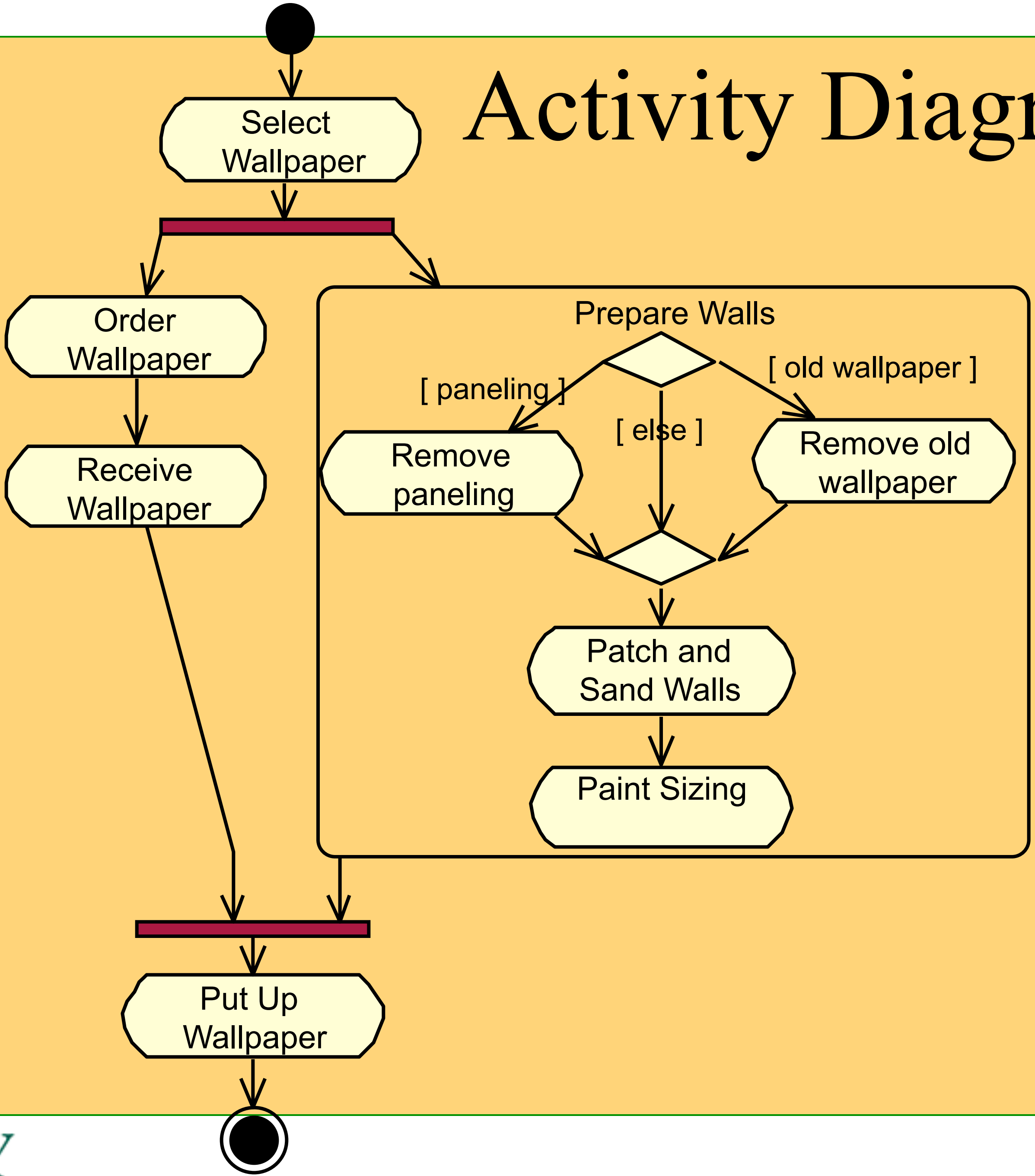
Communication Diagrams



State Diagrams



Activity Diagrams

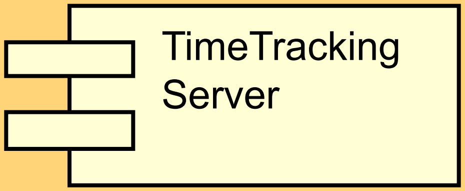


Logical View

- Describes an architecturally significant subset of the design model
- Contains a subset of classes, packages, and use-case realizations
- Concerns the functionality, behavior, use of frameworks and patterns
- Uses Class, Interaction, and State Diagrams

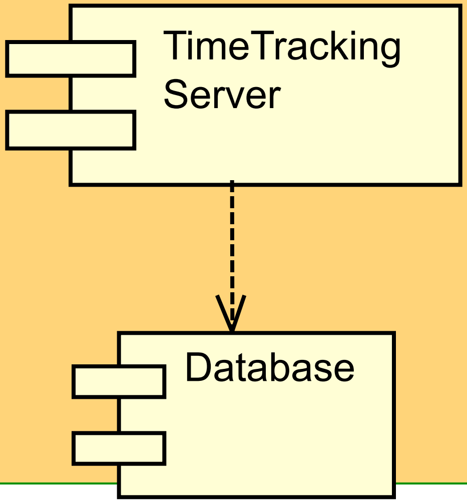
Process View

- Describes threads of control and communication between them
- Concerns the availability, reliability, scalability, performance, synchronization
- Uses Component, Class, and Collaboration Diagrams



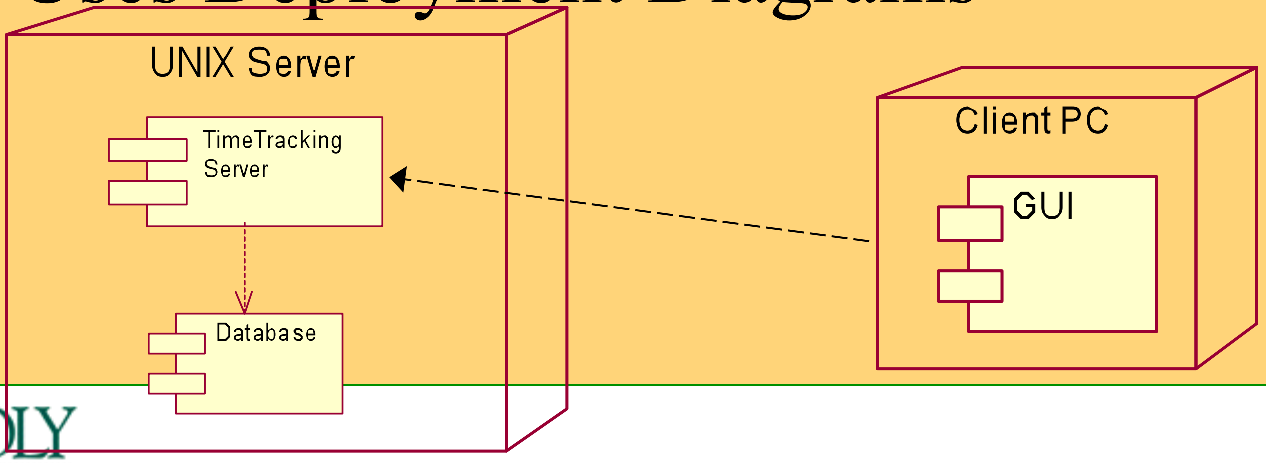
Implementation View

- Describes software component organization
- Concerns team organization and configuration management
- Uses Component Diagrams



Deployment View

- Describes physical network configurations
- Concerns the performance, throughput, fault-tolerance, availability, installation, and maintenance
- Uses Deployment Diagrams


UNIX Server
TimeTracking Server
Database
Client PC
GUI