Recitation 6

Software Architecture

Software Architecture

The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

[Source: Software Architecture in Practice]

Architectural Drivers

You make design decisions based on / to meet requirements for the following drivers:

- Functional requirements
- Quality attribute requirements
  - Performance
  - Availability
  - Scalability
  - Availability
  - Security
  - ...
- Technical & business constraints

Design Decisions

Driver	Requirement	Decision (what) & Rationale (why, how)
Availability	System should be up and responding to requests 99% of the time	Have redundant copies of the service behind a load balancer. <How does this help meet the requirement?>
Scalability	System should be able to scale with the number of requests up to X requests per second while meeting latency requirements	Have redundant copies of the service behind a load balancer. <How does this help meet the requirement?>
Performance	System should respond to a request within X seconds, or System should respond to N requests per second	Optimize model storage and loading / Optimize the way predictions are stored and retrieved, etc.

Architectural Diagrams

- Used to document and communicate design decisions and rationale for making these decision (architecture of the system)
- Should be able to easily understandable by others (depending on the type of audience)
- Notations (Informal - whiteboard, UML, etc.)
- In this course:
  - We do not insist on a particular notation
  - BUT, you should make sure that your diagrams
    - Have clear meanings with key/legend
    - Have consistent use of notations (lines, boxes, etc.)
    - Accompanying text if the diagrams alone are not sufficient to convey the whole picture

Signs of Bad Documentation

- All lines look the same (arrows don't mean anything, or could mean many things)
- Inconsistent use of notations
- Lack of key or legend
- Too little or too much detail
- Implementation details mixed with architectural abstractions
- Missing relationships between elements
- Incomplete prose/rationale - poorly described designs
- No discussion of alternatives

Example Diagram 1

- What do you understand from this diagram? What qualities are important?
- What is missing? What additional information would you add?

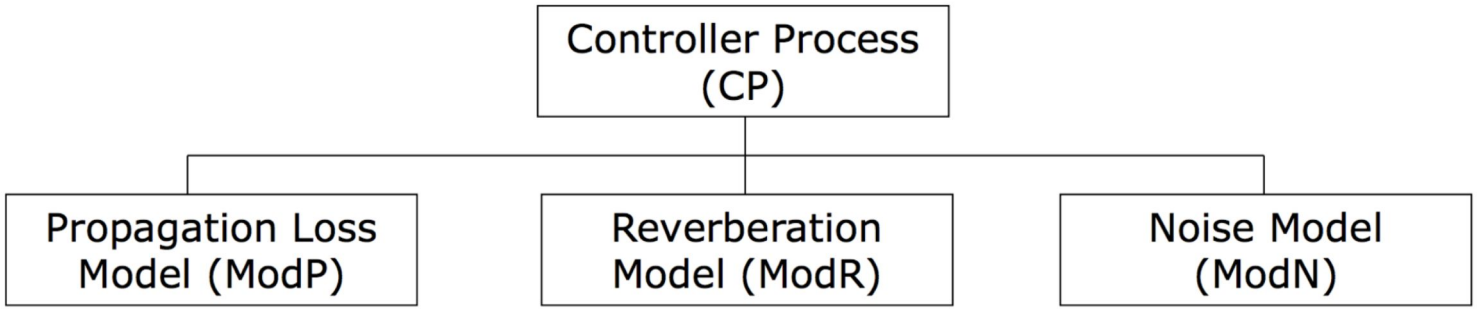
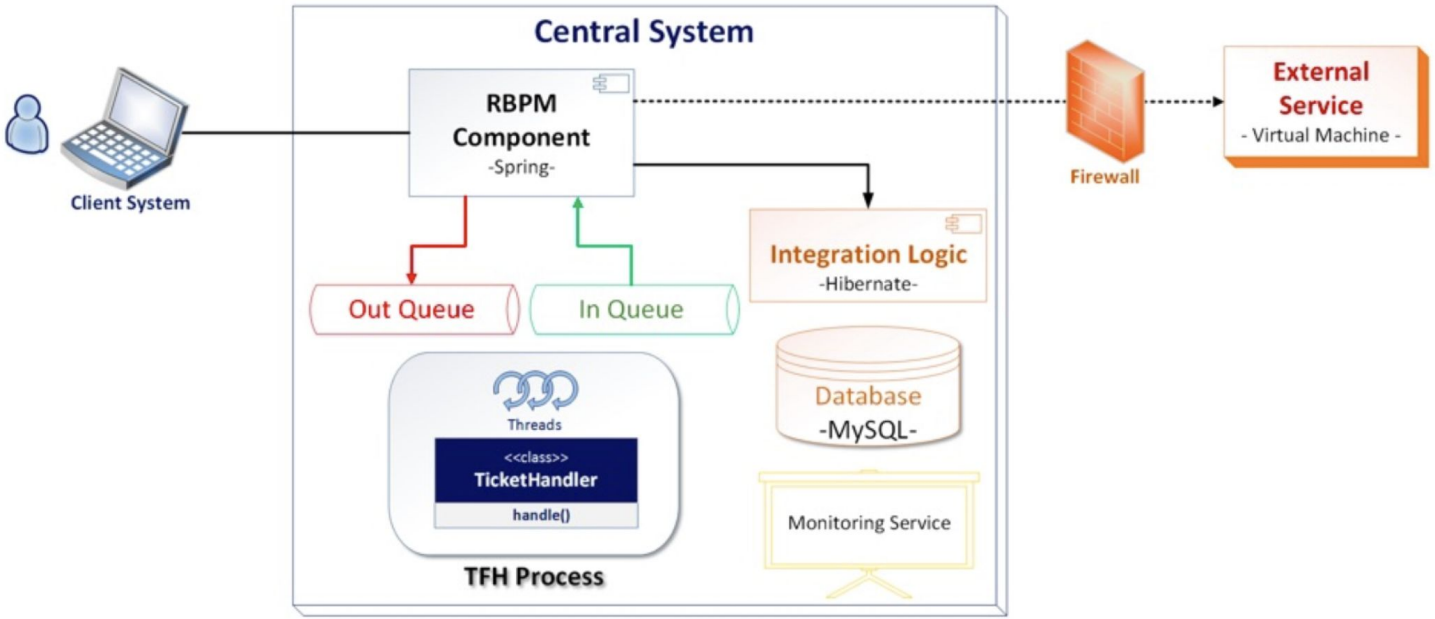
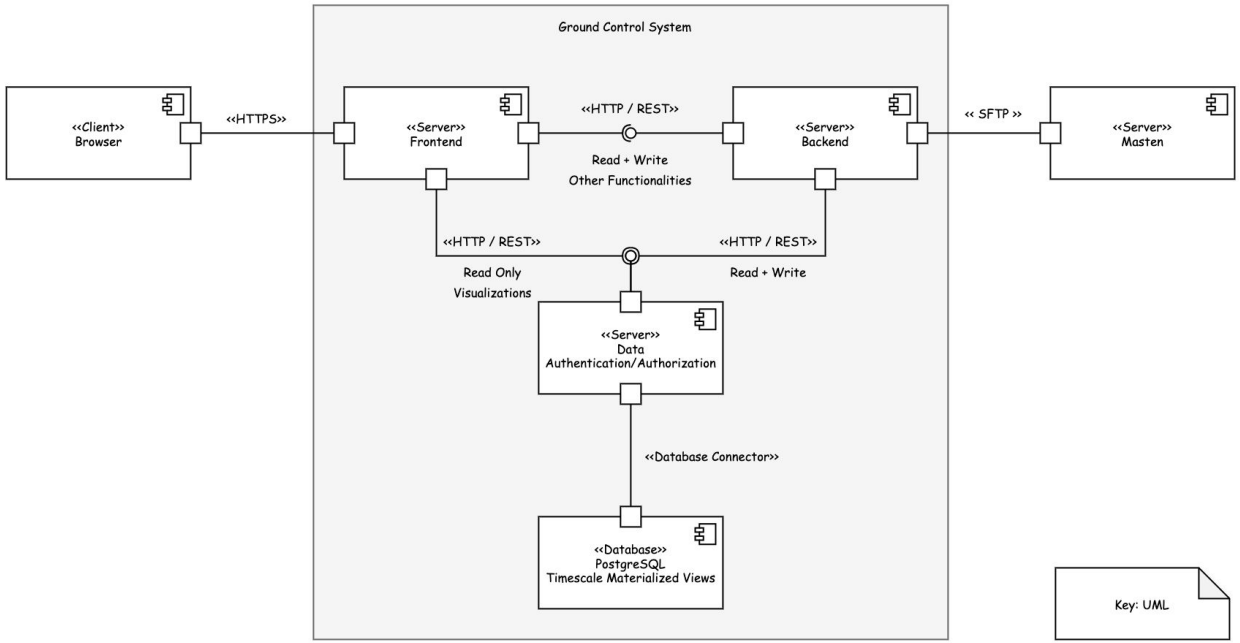


Figure X: Overall Software System Structure

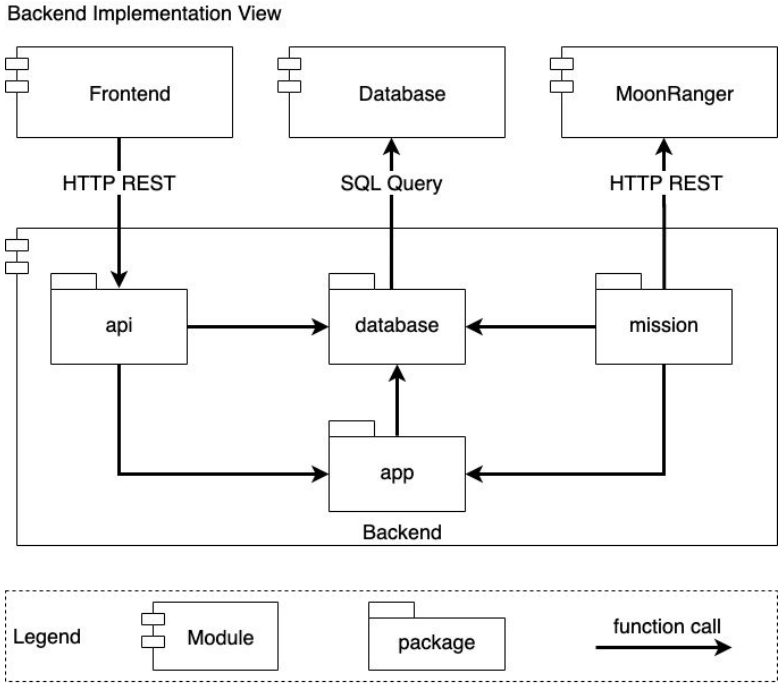
Example Diagram 2



Example Diagram 3 (UML)



Example Diagram 4



api: Setup of backend server to handle api requests. Defines valid routes that the server provides. Contains all the api endpoint functions that handle requests in and out of the frontend. Endpoint functions are named 'GetXxxx' or 'PostXxxx'. Initialised by main.

database: Setup of database connection. Contains all the access functions to the database. Access functions return data stored in a struct model. Functions are named 'SelectXxxx' or 'InsertXxxx'. Initialised by main.

mission: Setup of MoonRanger external connections. Contains the functions to pull data in or push data out of the backend system. Initialised by main.

app: Contains the core logic of the backend. api, database and mission are to minimise the amount of logic handling, and call appropriate functions from app.

Methodology

- Identify: what design decisions do I want to show?
- Identify quality attributes of interest
  - Focus on one or two quality attributes in each diagram
- Specify software elements
  - Use different shapes for different kinds of elements
- If needed, annotate each element with attributes
  - What information do I need to reason about the quality attributes from step 2?
- Identify relationships between elements
  - Use different types of edges for different relationships
  - Clearly define the meaning of an edge and its labels
- Analyze the impact of the design decisions on the quality attributes

Example Scenario

- Detect the presence of COVID-19 from audio recordings and gyro sensors using an app
- Users lie on their back and place the smartphone on their chest, and a deep neural network will classify audio and movement as COVID-19 symptoms or not
- The machine learning model can be put as part of the app, or can be deployed as its own microservice in the cloud. Which one would you opt for, and why?

Methodology

- Identify: what design decisions do I want to show?
  - ??
- Identify quality attributes of interest
  - ??
- Specify software elements
  - ??
- Annotate each element with attributes, what information do I need to reason about the quality attributes from step 2?
  - ??
- Identify relationships between elements
  - ??

Methodology

- What design decisions do I want to show?
  - Where should we perform the ML tasks? Phone, or in the cloud infrastructure?
- Identify quality attributes of interest
  - Performance (latency), Power/battery consumption, Cost to the consumer, Operational cost, Ease of deployment (frequency?), Scalability, Availability (network availability, etc.)
- Identify system elements
  - Platforms: Phone (containing the app, sensors, etc.), Cloud server
  - ML tasks: collect data from sensors, inference, model updation
- If needed, annotate each element with its attributes
  - Platforms: Available computational resources/power, available space, etc.
  - ML tasks: Estimated amount of computation (data size, number of operations)

Methodology

- Identify relationships between elements
  - Edges between platforms: data transfer (direction, protocol, labeled with mb/sec, type of request/response)
  - Edge from task to platform: allocation of a task on a platform
- Analyze the impact of design decisions on quality attribute requirements
  - Performance (latency)?
  - Power/battery consumption? Space?
  - Cost to the consumer, Operational cost?
  - Ease of deployment / deployment frequency?
  - Scalability?
  - Availability (network availability, etc.)?

Key Takeaways

- Think about: what design decisions you want to show in the diagram
- Be consistent and unambiguous
- Don't overload the meaning of lines, boxes, etc.
- Use multiple diagrams if needed (keep each diagram simple)
- Include text to supplement diagram if needed
- It's OK to invent your own notation, as long it clearly and concisely conveys the meaning

Thank You!