Software Architecture

AI Engineering - Recitation 5

Overview

- Defining Software Architecture
- Architectural Drivers
- Making Design Decisions
- Telemetry
- Activity - COVID-19 detection using smartphones
- Architectural Diagrams

Defining 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

Make design decisions based on / to meet requirements for:

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

Making Design Decisions

- Identify qualities for the system using the drivers
- Come up with more than a few designs
- Evaluate designs with respect to qualities
- Compare designs and identify tradeoffs

Design Decisions

Driver	*Requirement*	*Decision (what) & Rationale (why, how)*
Availability	System should be up and running for 99% of the time	Have more than one instances of the service running behind a load balancer
Scalability	System should be able to scale with the number of requests up to X requests per second while meeting latency requirements	Have more than one instances of the service running behind a load balancer
Performance	System should respond to a request within X seconds, or system should respond to N requests per second	Optimize model storage and loading, or the way predictions are stored and retrieved

Don’t forget Telemetry

- Why?
  - Monitor system operation
  - Ensure system goals are met
  - Improve the system

- Challenges
  - What data to collect?
  - How much to collect?
  - Where to store it?
  - How to avoid bottlenecking?
  - What would it cost?
  - How to ensure privacy?

Activity - COVID-19 Detection Using Smartphones

- 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.
- You are trying to decide on where to deploy the ML model, whether on the phone or in the cloud?

How would you architect this system?

- What would be the interesting qualities in the system?
- What are the parts of the system?
- Is there any other alternative to deploy the model?
- How would the qualities be analyzed in each design?
- What are the tradeoffs of each design?
- How would you devise the telemetry system?

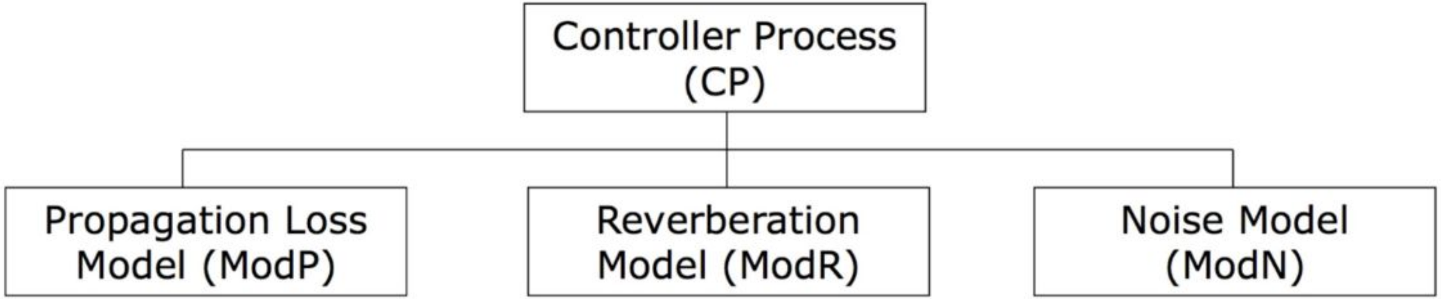
Choose one final design

Architectural Diagrams

- Useful to document and communicate design decisions and rationale behind architectures
- Should be easy to understand by others (depending on the type of audience)
- Many formal and informal notations (whiteboard, UML etc.)
- In this course:
  - You can have your own notation
  - BUT, you should make sure that your diagrams
    - Have clear meanings with key/legend
    - Use lines, boxes, shapes, arrows etc. in a consistent way
    - Are accompanied by text, if the whole picture cannot be conveyed sufficiently

Signs of Bad Documentation

- All lines look the same (arrows don’t mean anything, or could mean many things)
- Inconsistent notations
- No key/legend provided
- Too little or too much detail
- Implementation details mixed with architectural abstractions
- Missing relationships between elements
- Incomplete rationale - poor design descriptions
- No discussion of alternatives



Controller Process (CP)

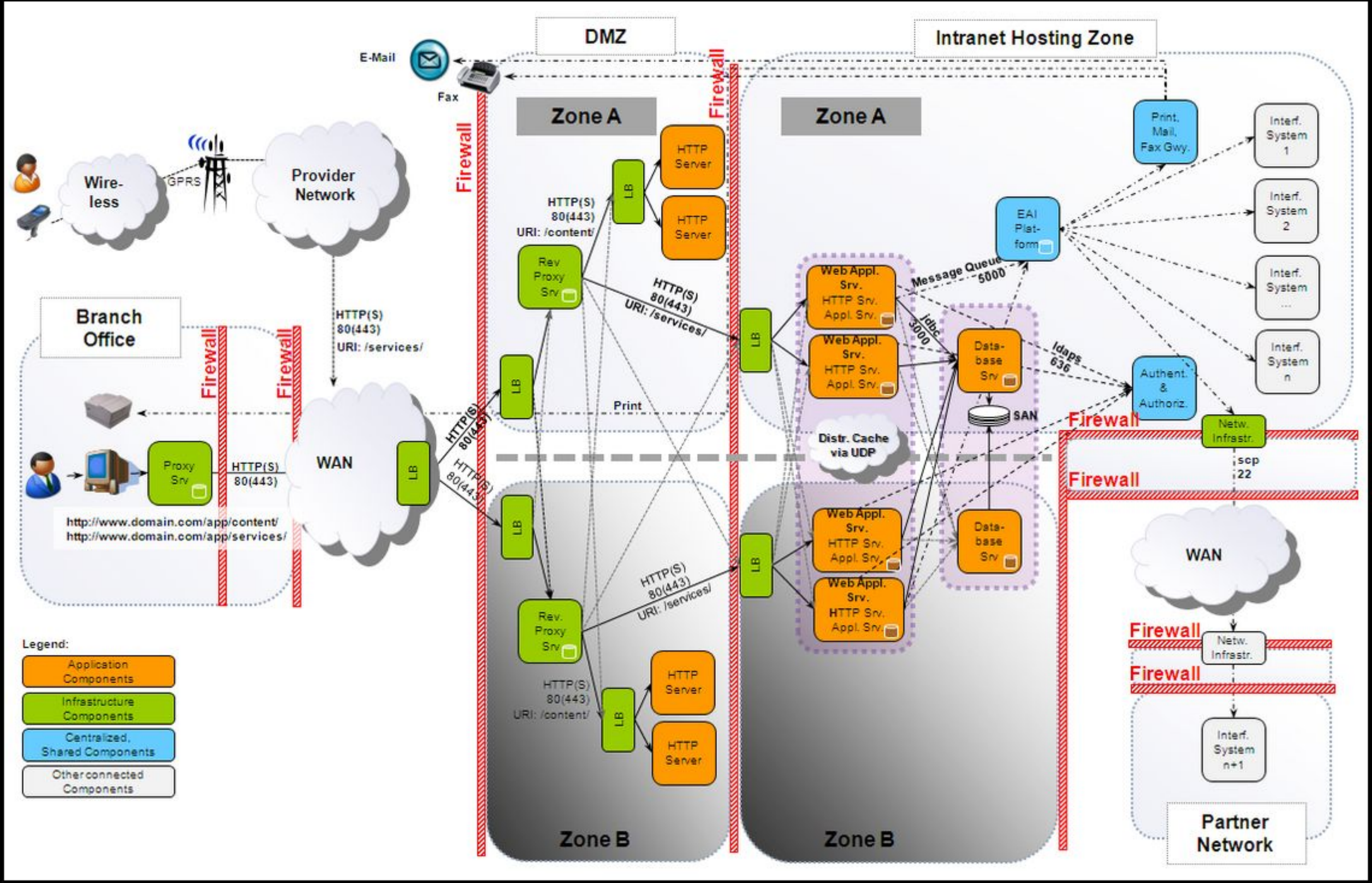
Propagation Loss Model (ModP)

Reverberation Model (ModR)

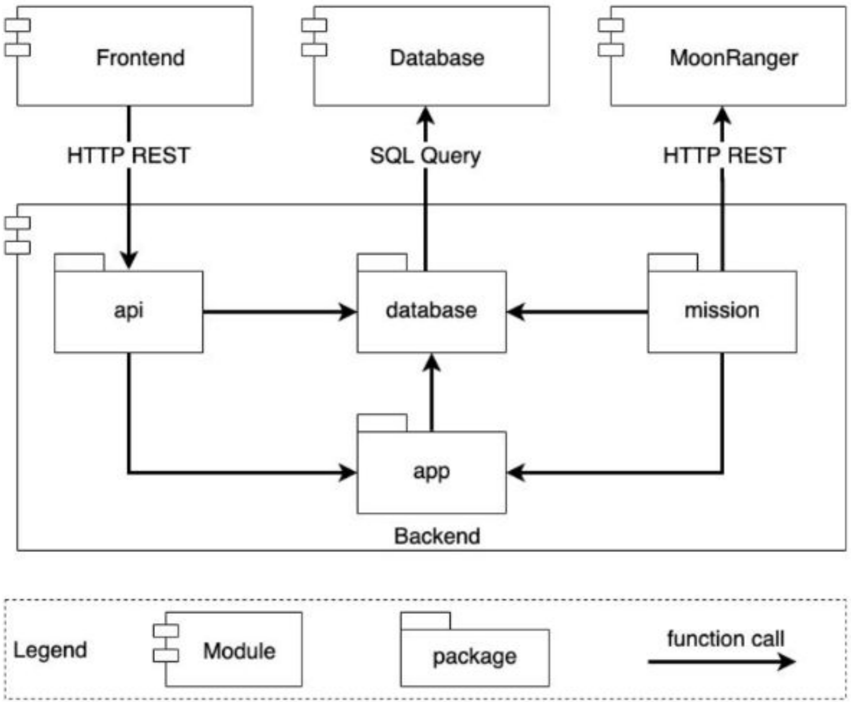
Noise Model (ModN)

Figure X: Overall Software System Structure

A Practical Guide to Software Architecture on WordPress.com



Backend Implementation View



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.

General Tips for Architectural Diagrams

- Think about what design decisions you want to show in the diagram
- Be consistent and unambiguous
- Do not overload the meaning of lines, boxes, etc.
- Use multiple diagrams if needed (keep each diagram simple)
- Include text to supplement diagram if needed
- Inventing your own notation is okay, as long as it clearly and concisely conveys the meaning