SOFTWARE ARCHITECTURE

Recitation 5, February 18th, 2022

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.

–Software Architecture in Practice

Architectural Drivers

- You make design decision based on/to meet requirements for the following drivers:
  - Functional requirements
  - Quality requirements
    - Performance
    - Availability
    - Scalability
    - …
  - Technical and business constraints

Quality Attributes → Design Decisions

Driver	Requirement	Decisions
Availability	System should be up and responding to requests to 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 with in X seconds, or System should respond to N requests per second	Optimize model storage and loading/Optimize the ways predictions are stored and retrieved/Cache frequent requests.

Trade Offs

- Often, various quality attributes are in conflict with each other.
- Optimizing one desired quality (accuracy) may decrease another desired quality (efficiency).
- Can you think of additional examples?
- Choosing which qualities you care about most and by how much is an important part of design.

Trade Offs Visualization

Assume we want to *minimize* f1 and f2.

ML Trade-offs: Accuracy vs. Interpretability

Architectural Diagrams

- Visual representation of the architecture; useful for documentation/communication
- Notations
  - Informal, UML, etc.
  - In this course
    - We do not require a particular notation
    - We do require
      - Clear & consistent meanings for symbols with a legend
      - Accompanying text if the diagrams alone are not sufficient.

Signs of Bad Documentation

- All lines look the same (arrows mean nothing or could mean many things).
- Inconsistent use of notation
- No key/legend
- Too little or too much detail
- Implementation details mixed with architectural abstractions
- Missing relationships between elements
- For accompanying textual documentation
    - Incomplete prose/rationalle
    - No discussion of alternatives

Signs of Bad Documentation

- Next, we'll look at some examples.
- For each example
  - What can you understand from this diagram?
  - What qualities are important?
  - What is done well?
  - What could be improved?

Example 1

Central System
RBPM
Component
-Spring-
Client System
Out Queue
In Queue
Integration Logic
-Hibernate-
Threads
<<class>>
TicketHandler
handle()
TFH Process
Database
-MySQL-
Monitoring Service
Firewall
External
Service
- Virtual Machine -

Example 2

Figure X: Overall Software System Structure

Example 3

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.

Activity: Doorbell Camera

You work for a company that makes doorbell cameras. The doorbell camera market is very crowded, and you want to add a feature that differentiates your product from the competition. Management has decided the best way to do this is to add customization to real-time doorbell activity alerts. The feature will use computer vision to identify people approaching the door. It must:

- mention if the person is in your contacts (with a similar photo)
- mention if the person is someone it's seen before
- mention if a the person approaching is carrying a package
- mention if the person is unknown and not carrying a package

Most of your doorbells operate on a battery, though some high-end models have a power supply. Your doorbells connect to a smarthome "hub" device (e.g. Google Home or Amazon Echo) over WiFi, which in turn is connected to a cloud-based platform that handles routing notifications to homeowners via an app.

Activity: Doorbell Camera

Your job is to design the architecture for this system.

1. Discuss ideas and ***tradeoffs*** between design choices.
2. Choose one design and draw an ***architecture diagram***.
   (Where does the model live?)
3. ***Justify*** why you chose this design.