Software Design

On the importance of planning…

Why do Projects Fail?

Steve McConnell describes how small projects aren't necessarily representative of the problems you'll encounter on larger projects:

> People who have written a few small programs in college sometimes think that writing large, professional programs is the same kind of work -- only on a larger scale. It is not the same kind of work. I **can build a beautiful doghouse in my backyard in a few hours.** It might even take first prize at the county fair's doghouse competition. But that *does not imply that I have the expertise to build a skyscraper*. The skyscraper project requires an entirely more sophisticated kind of expertise.

Architecting

Greek: ἀρχιτέκτων (architéktōn)

- **ἀρχι-** (arkhi-) meaning "chief" or "master"
- **τέκτων** (tektōn) meaning "builder" or "carpenter"

Architecting

What Doesn't Work

Dr. Paul Dorsey:

Projects are frequently built using a strategy that almost guarantees failure.

Building a large information system is like constructing a 20-story office building. If a bunch of electricians, plumbers, carpenters and contractors meet in a field, talk for a few hours and then start building, the building will be unstable if it even gets built at all. At one of my presentations, an audience member shared the quip that:

> "If building engineers built buildings with the same care as software engineers build systems, the first woodpecker to come along would be the end of civilization as we know it."

https://www.dulcian.com/articles/dorsey_top10reasonssystemsprojectsfail.pdf

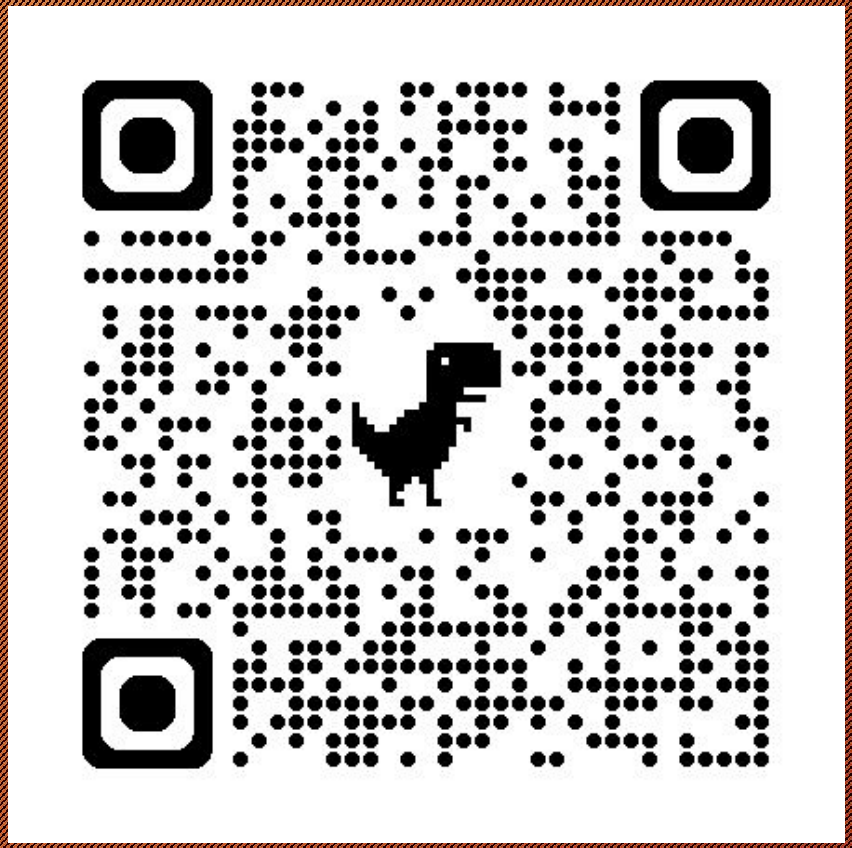
Software Engineering Failures

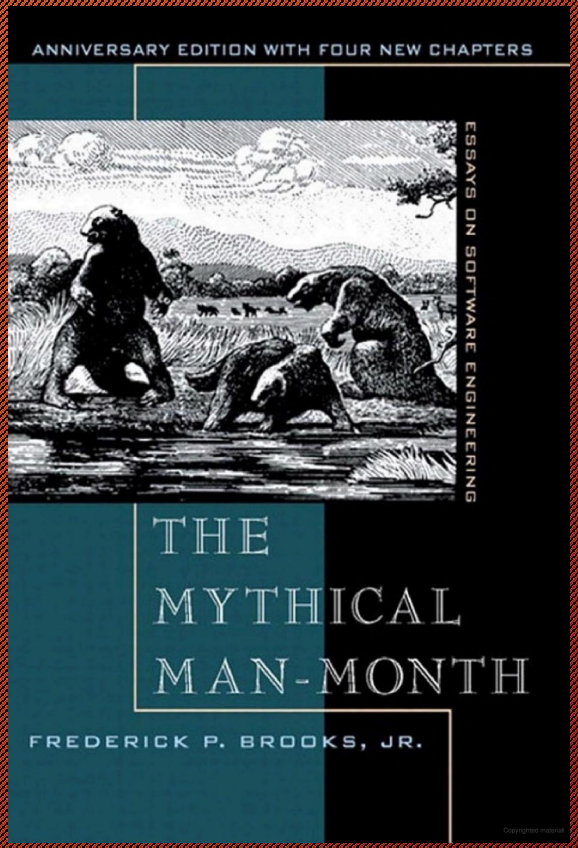
IBM survey in the success / failure rates of "change" projects finds;

1. Only 40% of projects met schedule, budget and quality goals
2. Best organizations are *10 times* more successful than worst organizations → 10x
3. Biggest barriers to success listed as people factors:
   - Changing mindsets and attitudes - 58%
   - Corporate culture - 49%.
   - Lack of senior management support - 32%.
4. Underestimation of complexity listed as a factor in 35% of projects

http://calleam.com/WTPF/?page_id=1445

Speaking of IBM …





Mythical Person-Month

- **Optimism** and Estimation,
- Person-Month **Myth**,
- Conceptual **Integrity**,
- **Communication** Breakdown,
- Changing **Requirements**,
- Lack of **Testing**,
- Inadequate **Documentation**.

Best Practices

1. Development process. Make this a conscious choice. Consider size and scope of project. Agile is not always the answer.
2. Requirements. Are you creating what the customer wants? Are there non-functional requirements? (efficiency etc.)
3. **Architecture**. How do the pieces fit together?

…

Best Practices (continued)

4. **Design**. Agile does *not* mean no planning! (or no documentation) Guiding principle: keep it simple (You Ain't Gonna Need It – YAGNI). How much design before coding?
5. **Construction**. Daily build and smoke test. Continuous or frequent integration.
6. **Peer reviews** of code.
7. **Testing**.

Mythical Person-Month

- Conceptual **Integrity**,
- **Realistic** Planning & Estimation,
- Effective **Communication**,
- **Change** Management,
- Thorough **Testing**,
- Comprehensive **Documentation**,
- Team Dynamics.

System Architecture

Making a plan

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. The term also refers to documentation of a system's "software architecture." Documenting software architecture:

- facilitates communication between stakeholders,
- documents decisions about high-level design, and
- allows reuse of design components and patterns between projects.

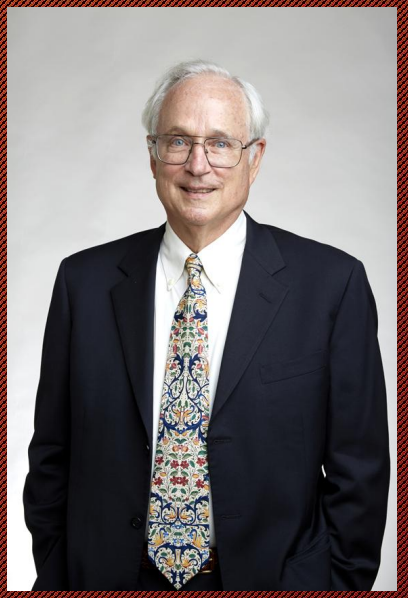
Software Architecture – Why?

- The software architecture discipline is centered on the idea of **reducing complexity through abstraction and separation of concerns.**

Fundamental Theorem of Software Engineering
– Butler Lampson

"All problems in computer science can be solved by another level of indirection."

"…except for the problem of too many levels of indirection."

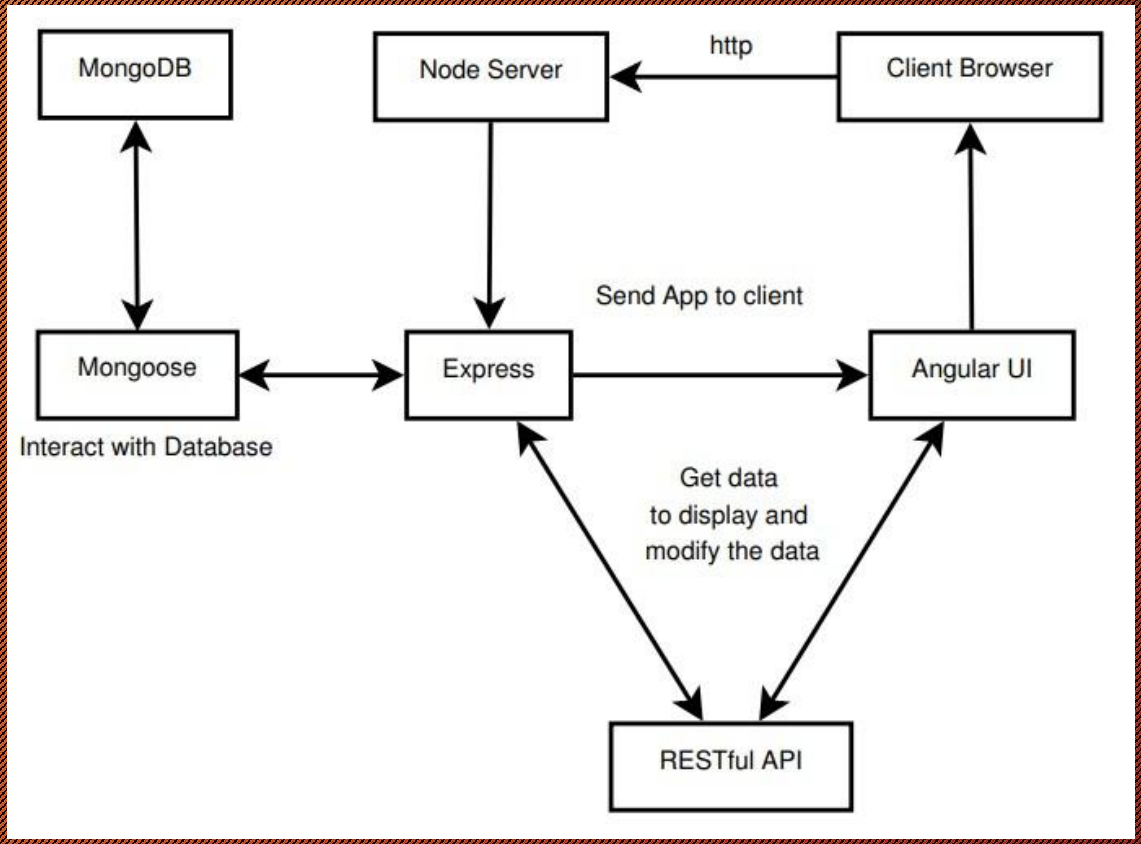


Software Architecture – Why?

- The software architecture of a program or computing system is a *depiction* of the system that *aids in the understanding of how the system will behave*.
- Need a unifying architectural vision to ensure system qualities such as performance, modifiability, and security.
- Focus on the *interface* between the components (one of the most error-prone aspects of system design)

http://www.sei.cmu.edu/architecture/

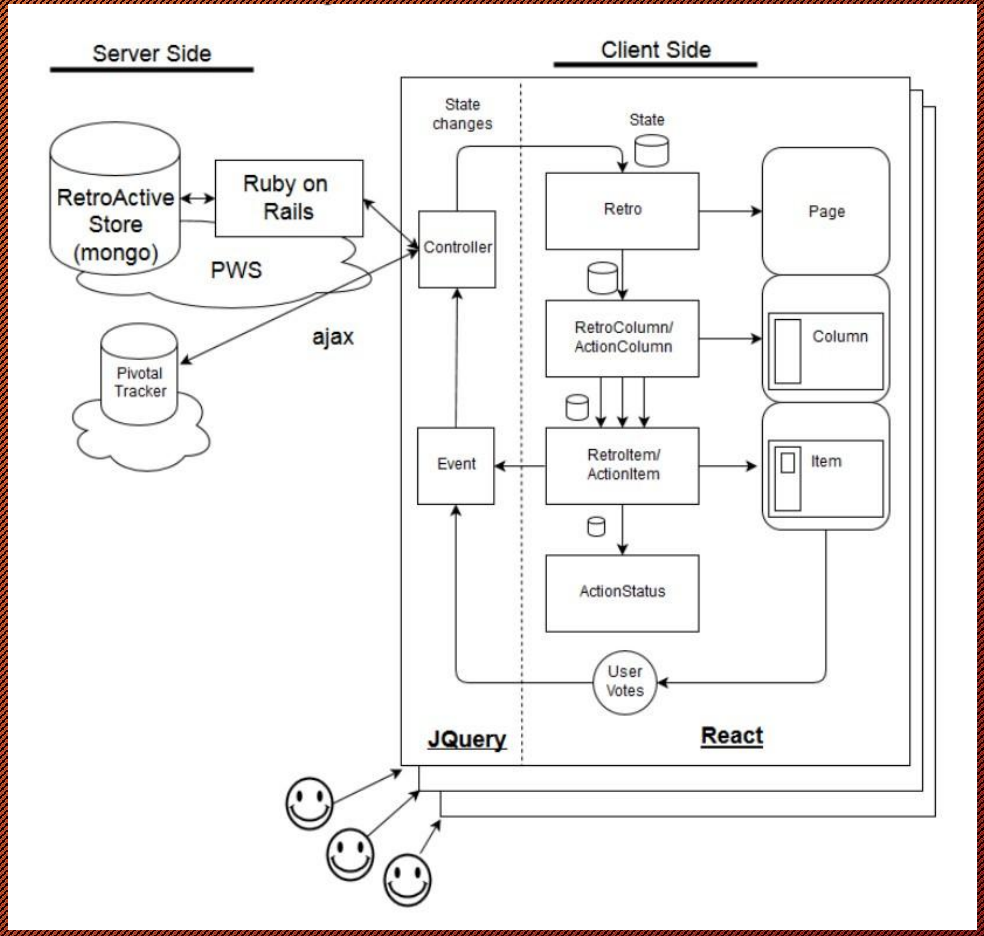
Architecture – Example 1



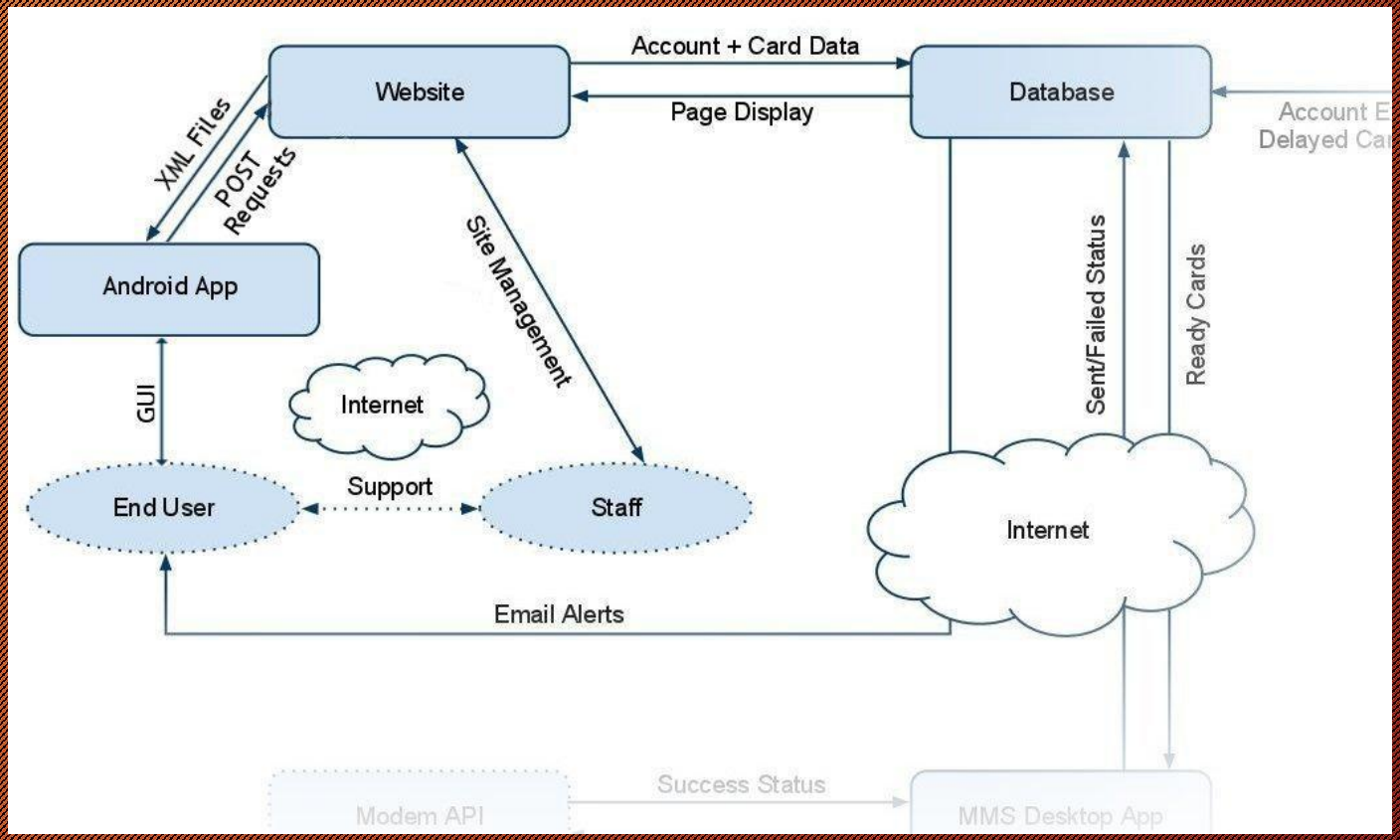
From Avaya final report (2015)

Architecture– Example 2

From Pivotal final report (2016)



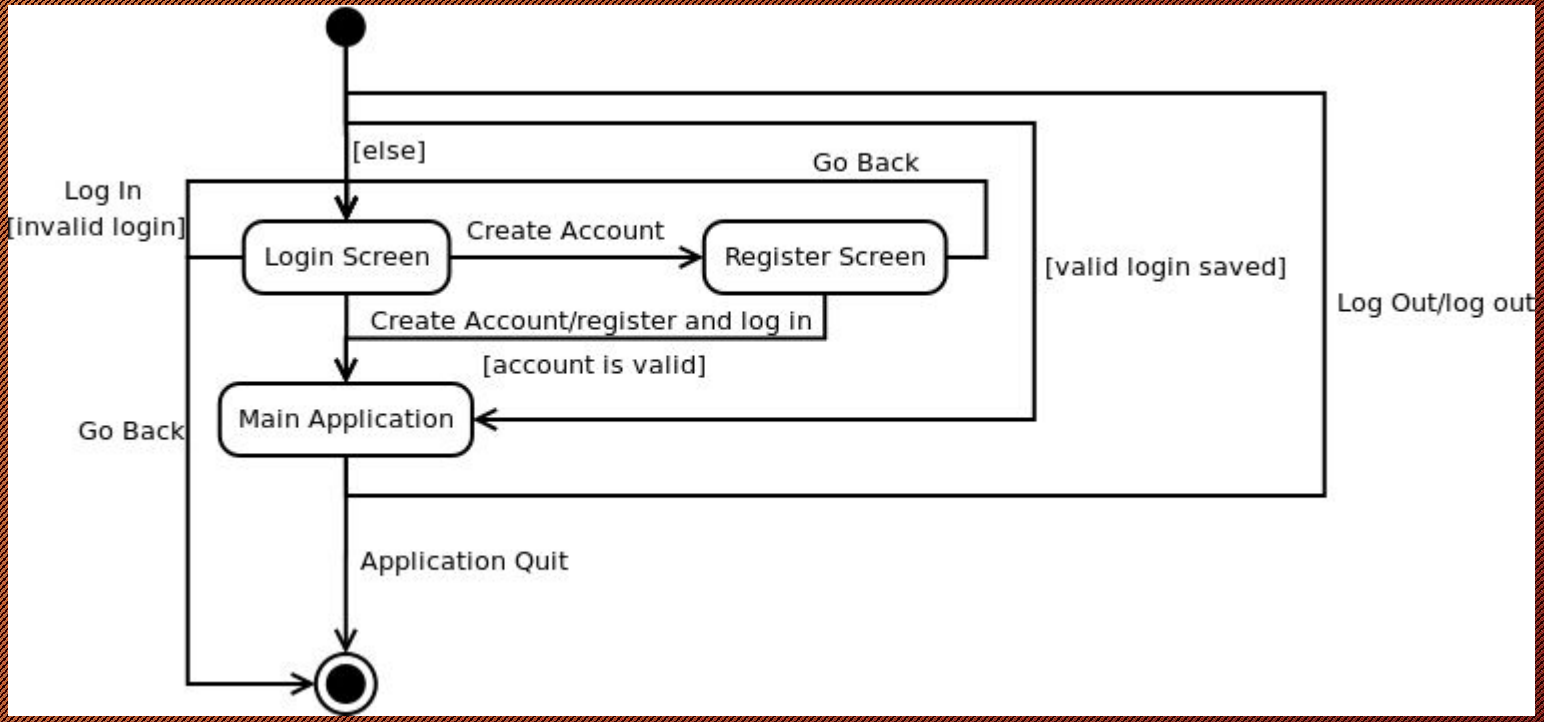
Architecture – Example 3



From ModsDesigns final report (2011)

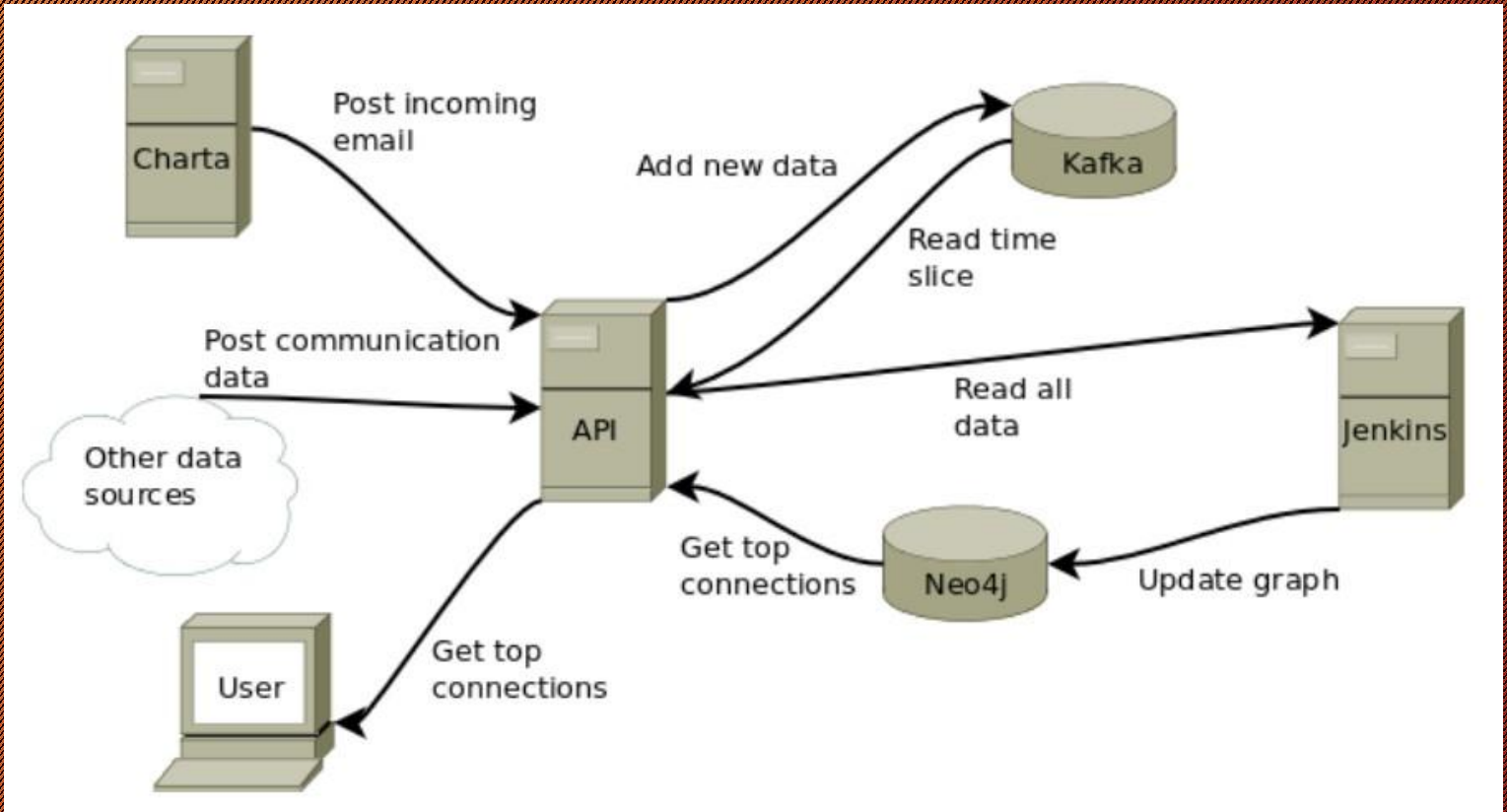
State Diagram

See ModsDesigns (2011)



System flow, supplements architecture

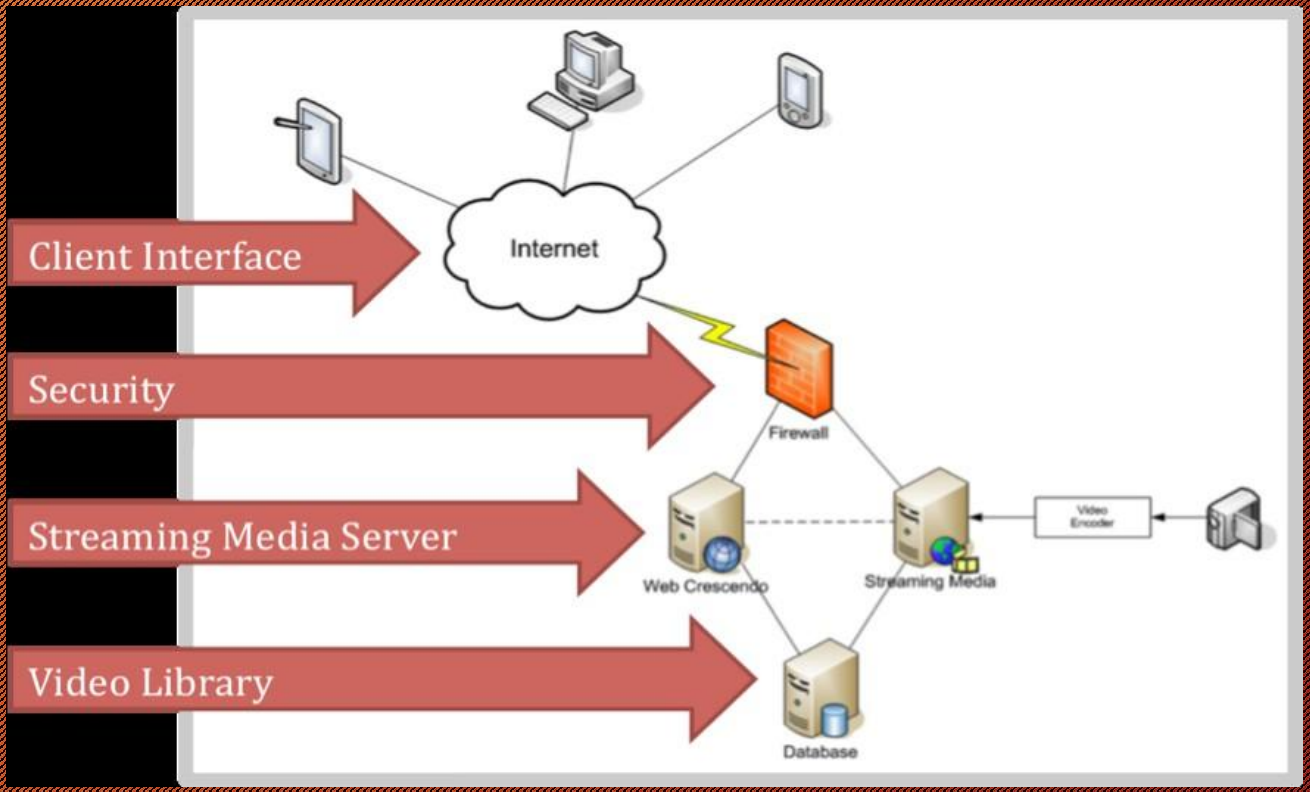
Architecture – Example 4



From FullContact final report (2014)

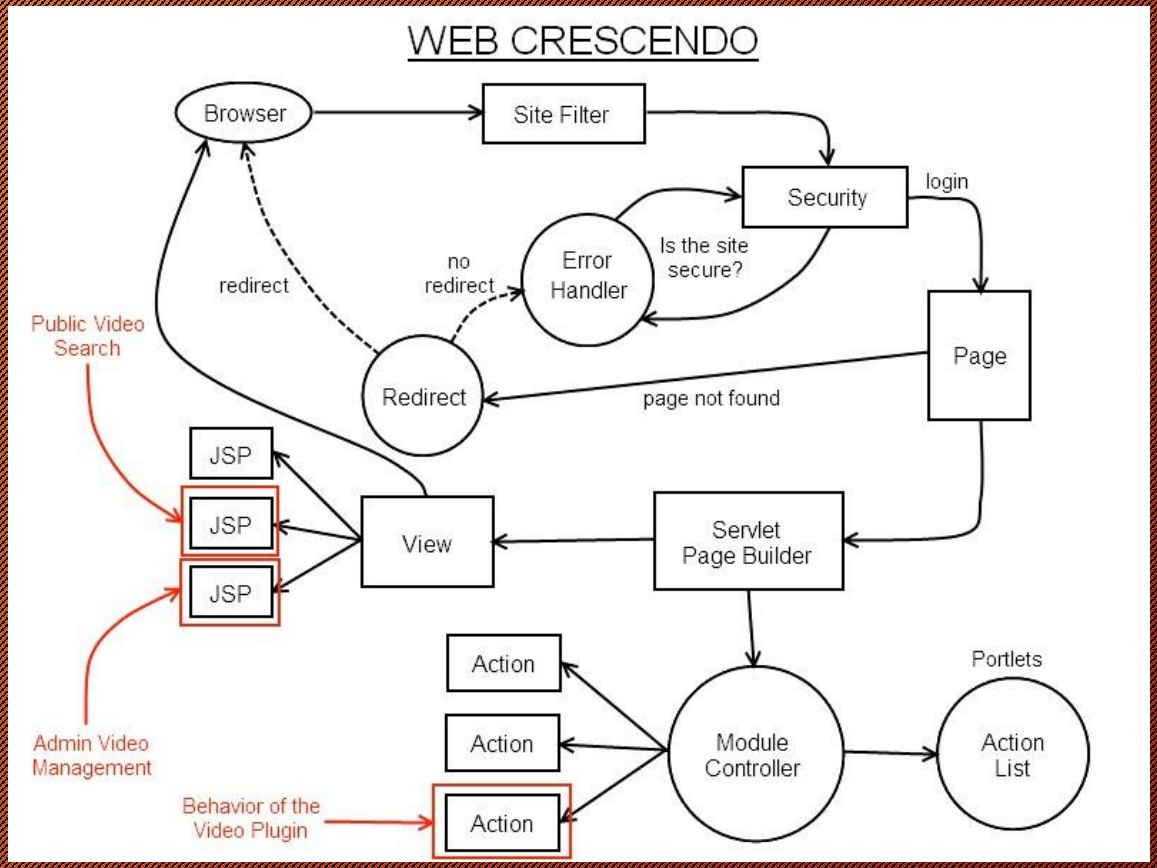
Architecture– Example 5

From SMT final report (2011)



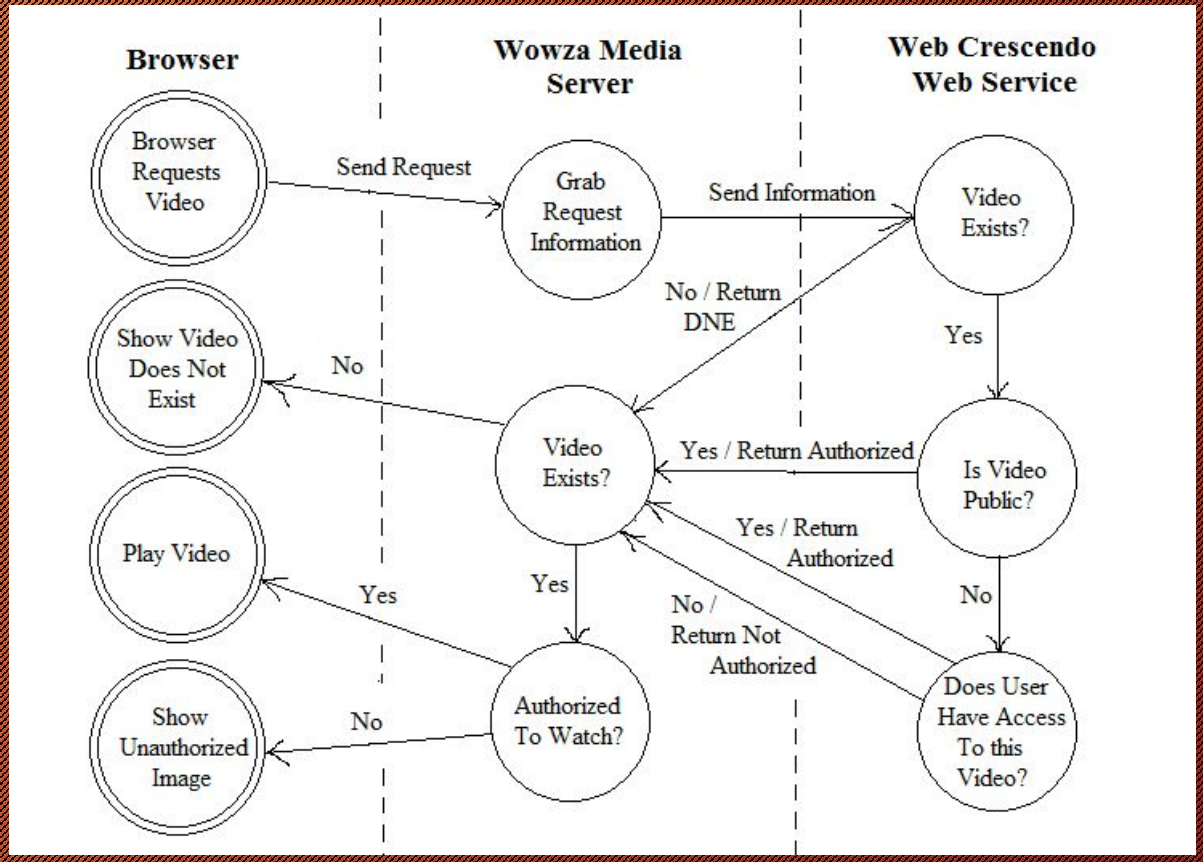
Architecture– Example 6

From SMT (2011)



Finite Automata/Activity Diagram

From SMT (2011)



System flow, supplements architecture

More Examples

More links available on Design Document page

Technical Design

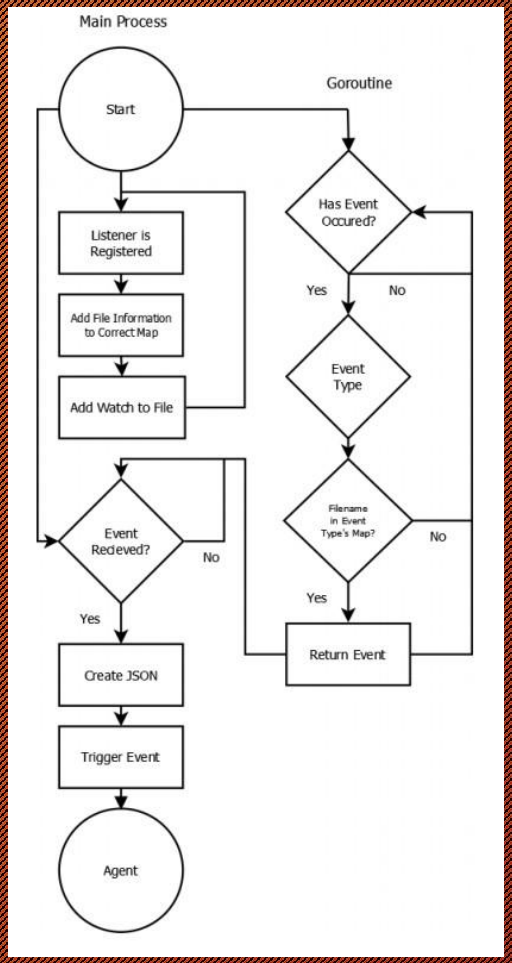
Adding some details

Technical Design

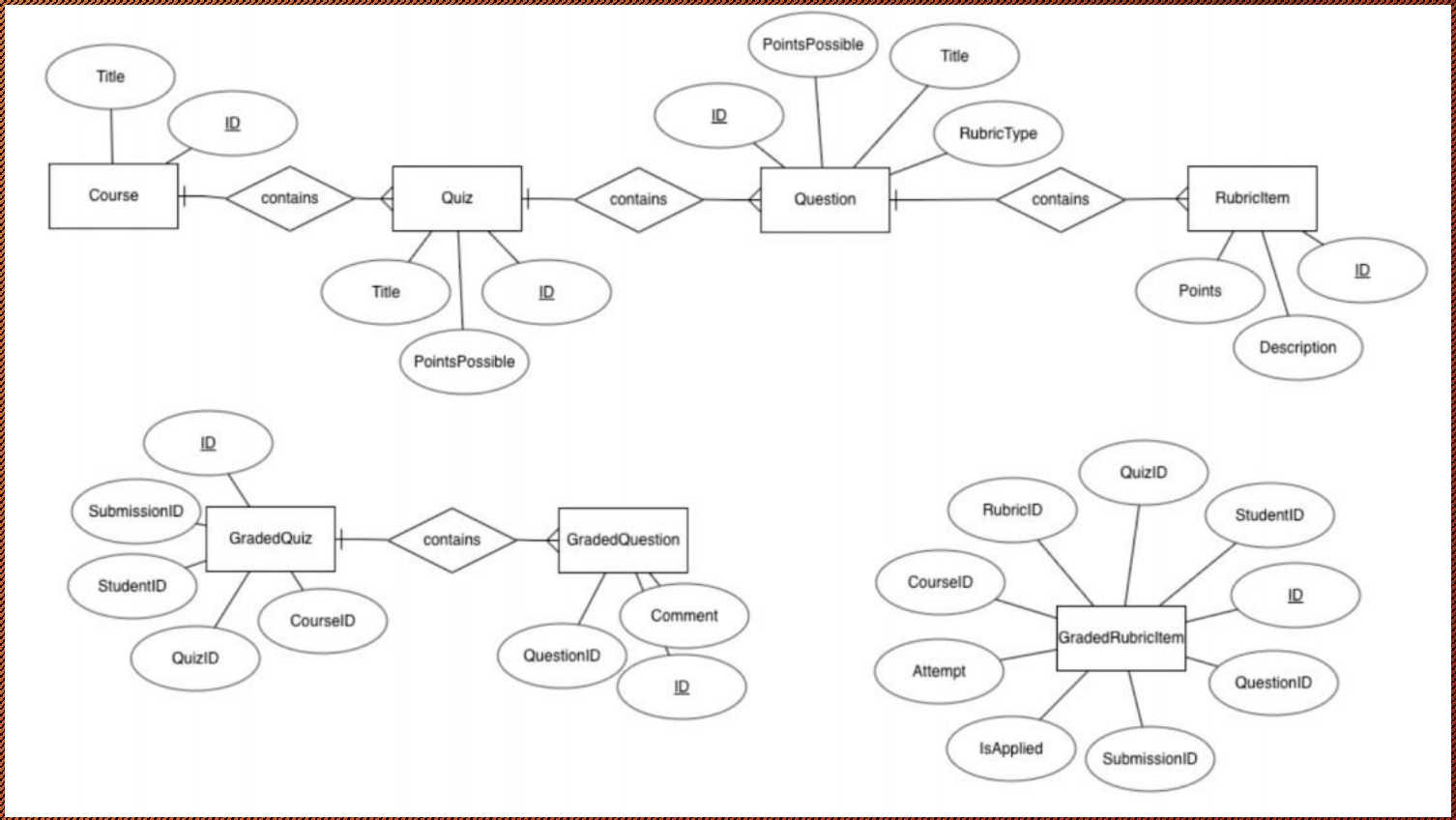
- Architecture diagrams focus on the interface between components. They are “big picture” drawings.
- It can also be important to focus on details of a particular component.
- These diagrams are likely more familiar to you.

Flowchart

From JumpCloud final report (2014)



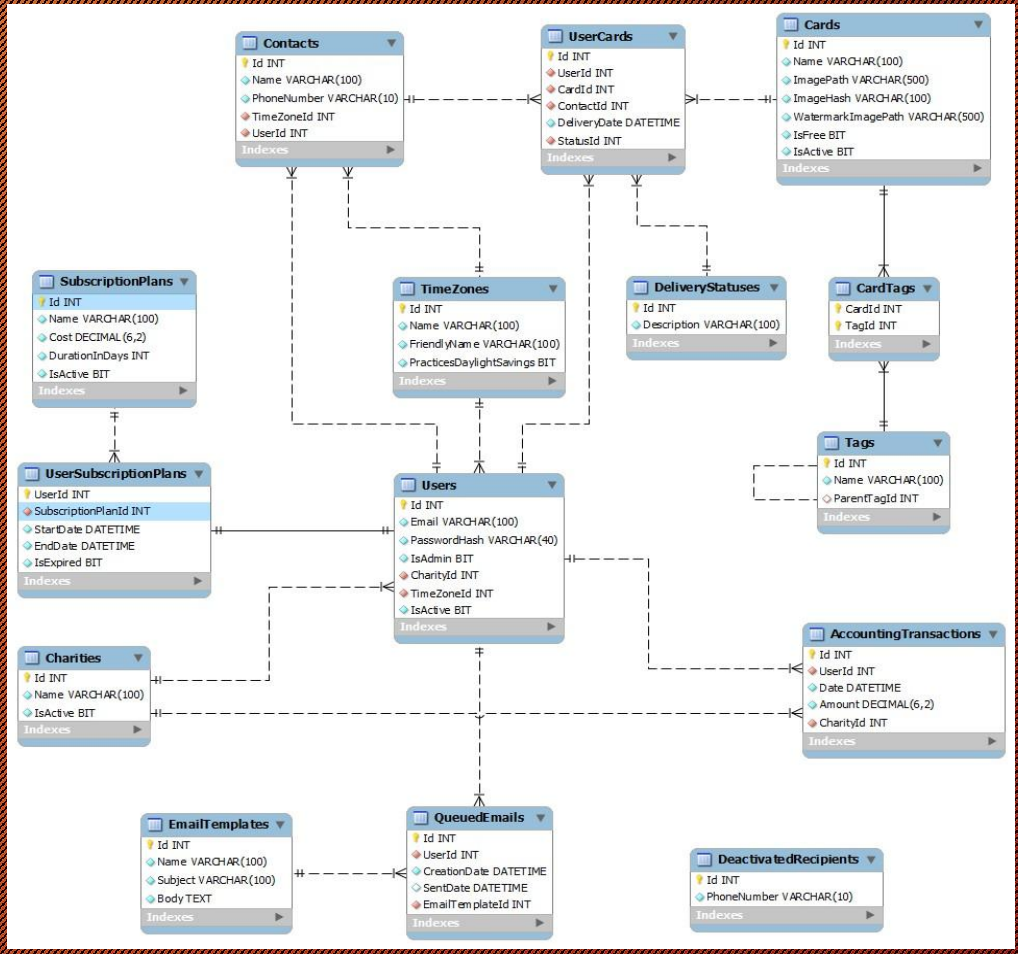
Entity-Relationship Diagram



From CSM Paone (Fall 2020)

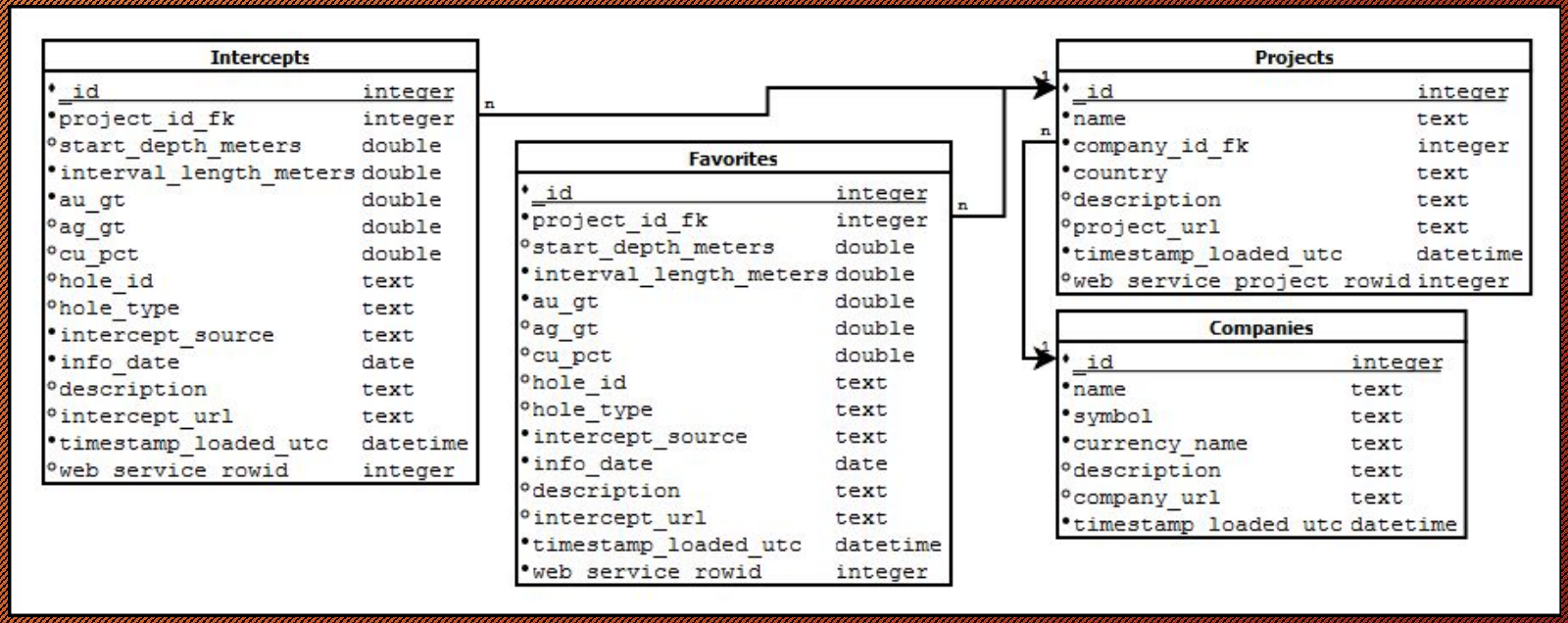
Database Schema

From ModsDesigns (2011)



Database Schema

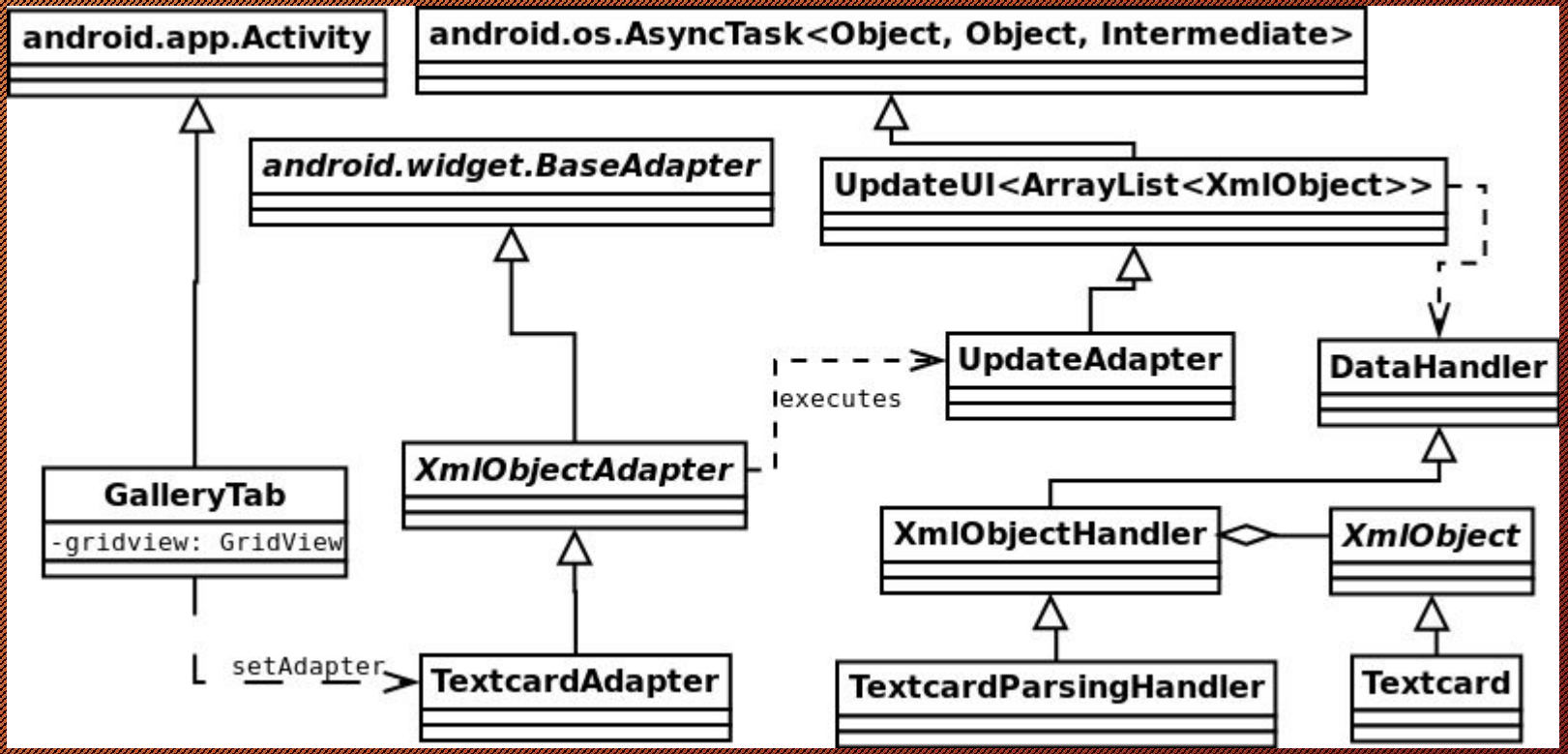
From Newmont 2 (2011)



For database tables you create, include supporting text that describes the various fields and relationships. That level of detail is not needed for tables in an existing customer system.

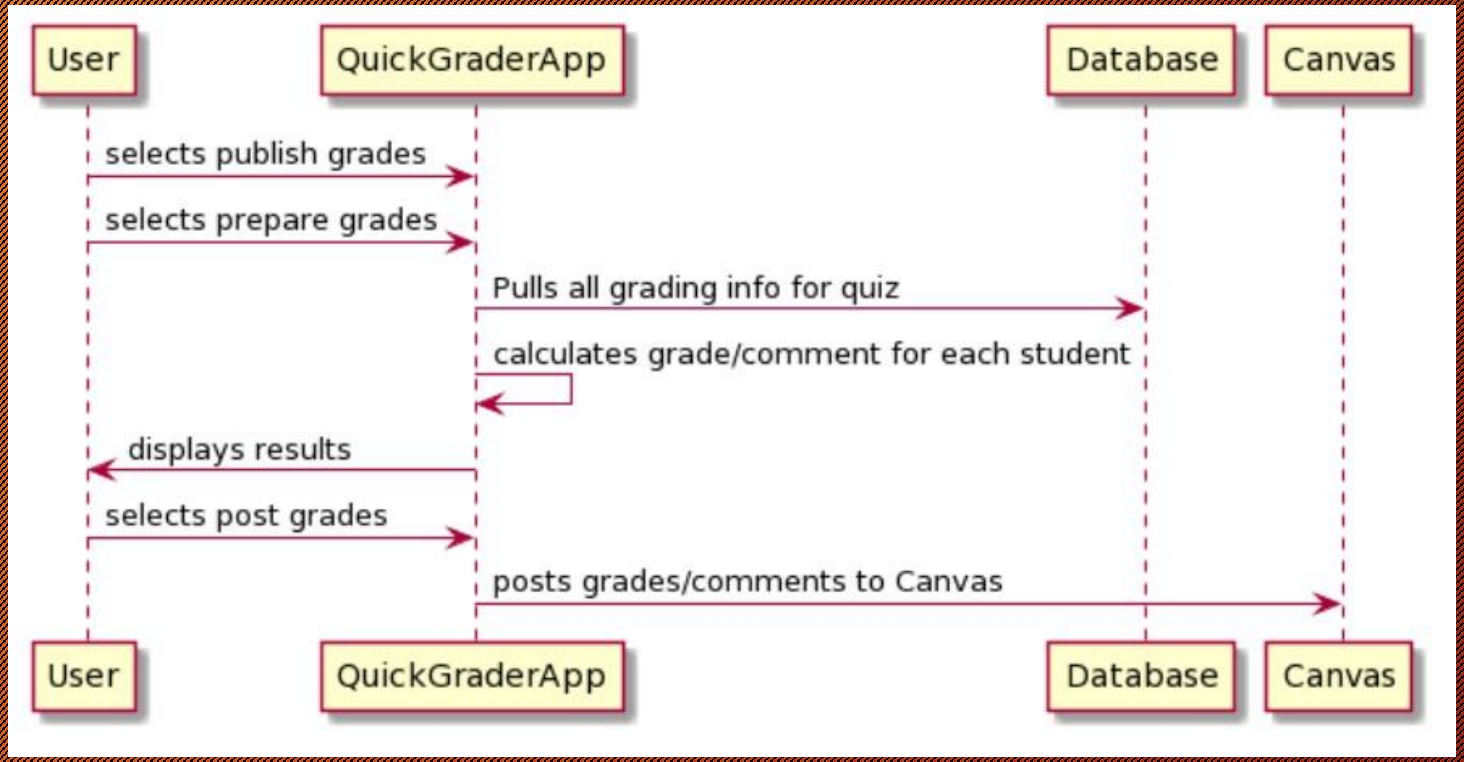
UML

From ModsDesigns (2011)



Remember that Dia has an option to not show attributes/methods.

Sequence Diagram

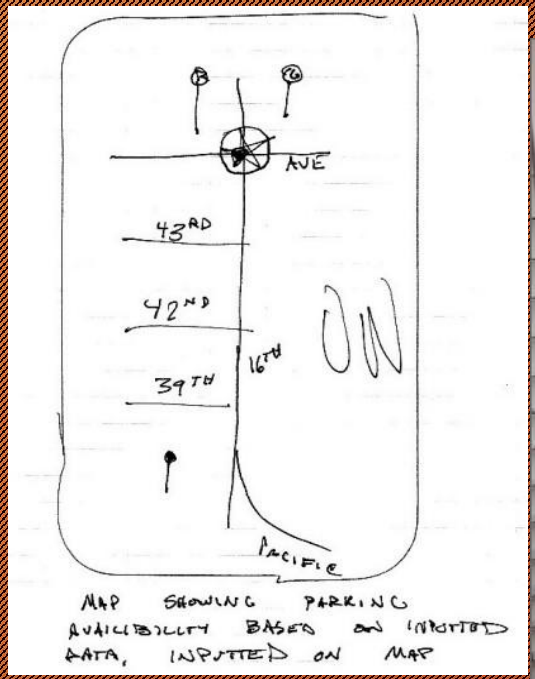


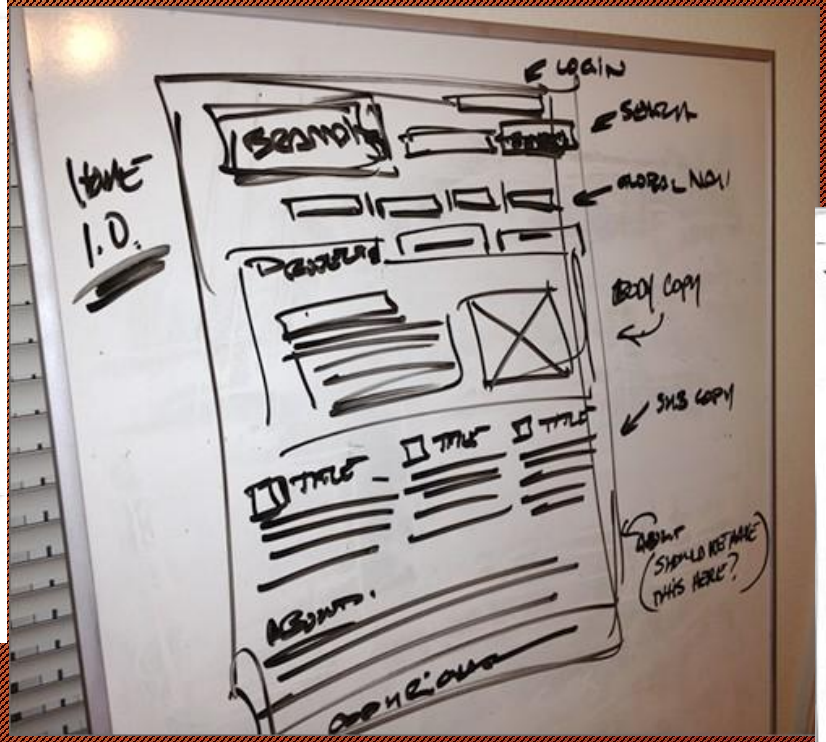
From CSM Paone (Fall 2020)

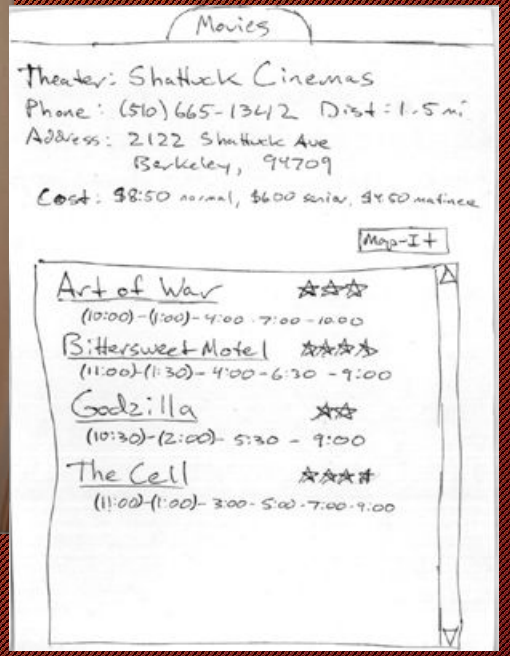
Interface Design

Can humans use this?

Rough Sketches

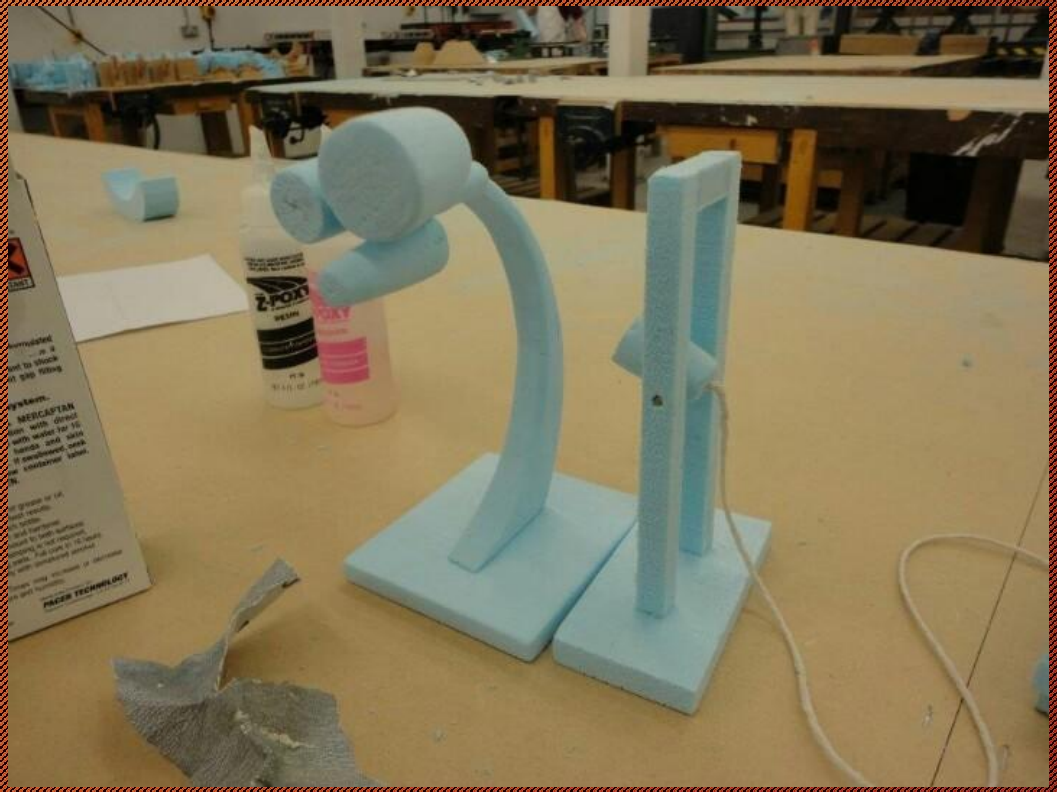






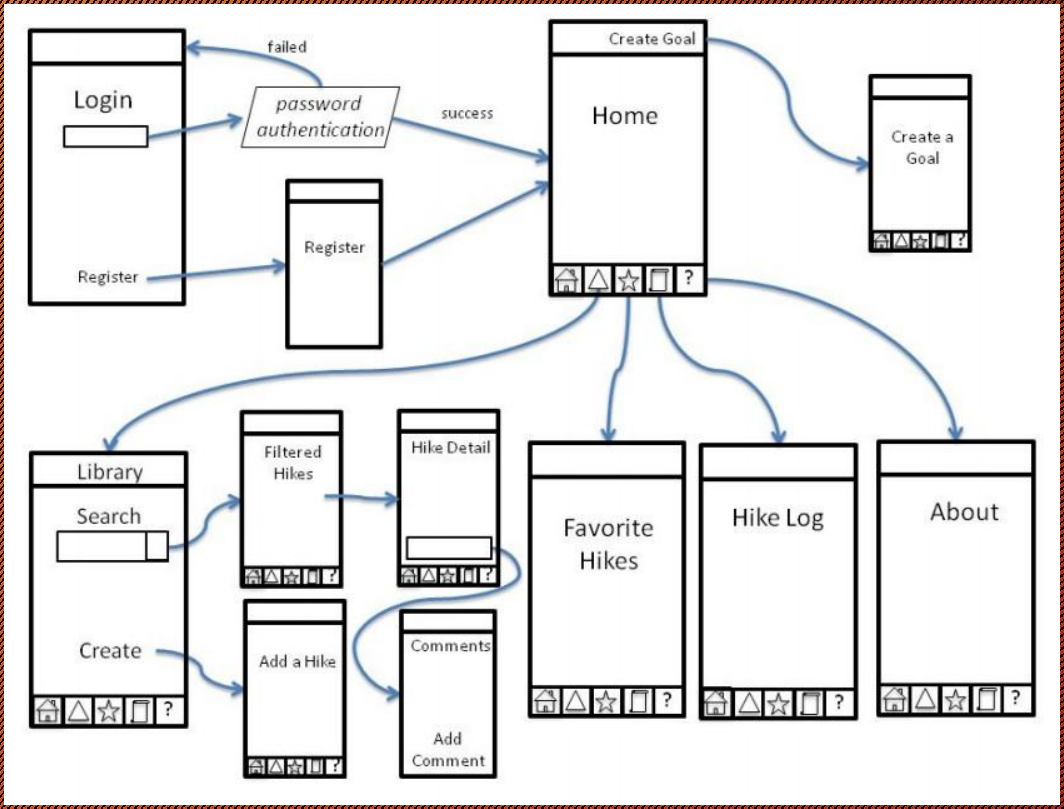
From CSM Thompson (2022)

Sketches Don’t Have to be Drawings



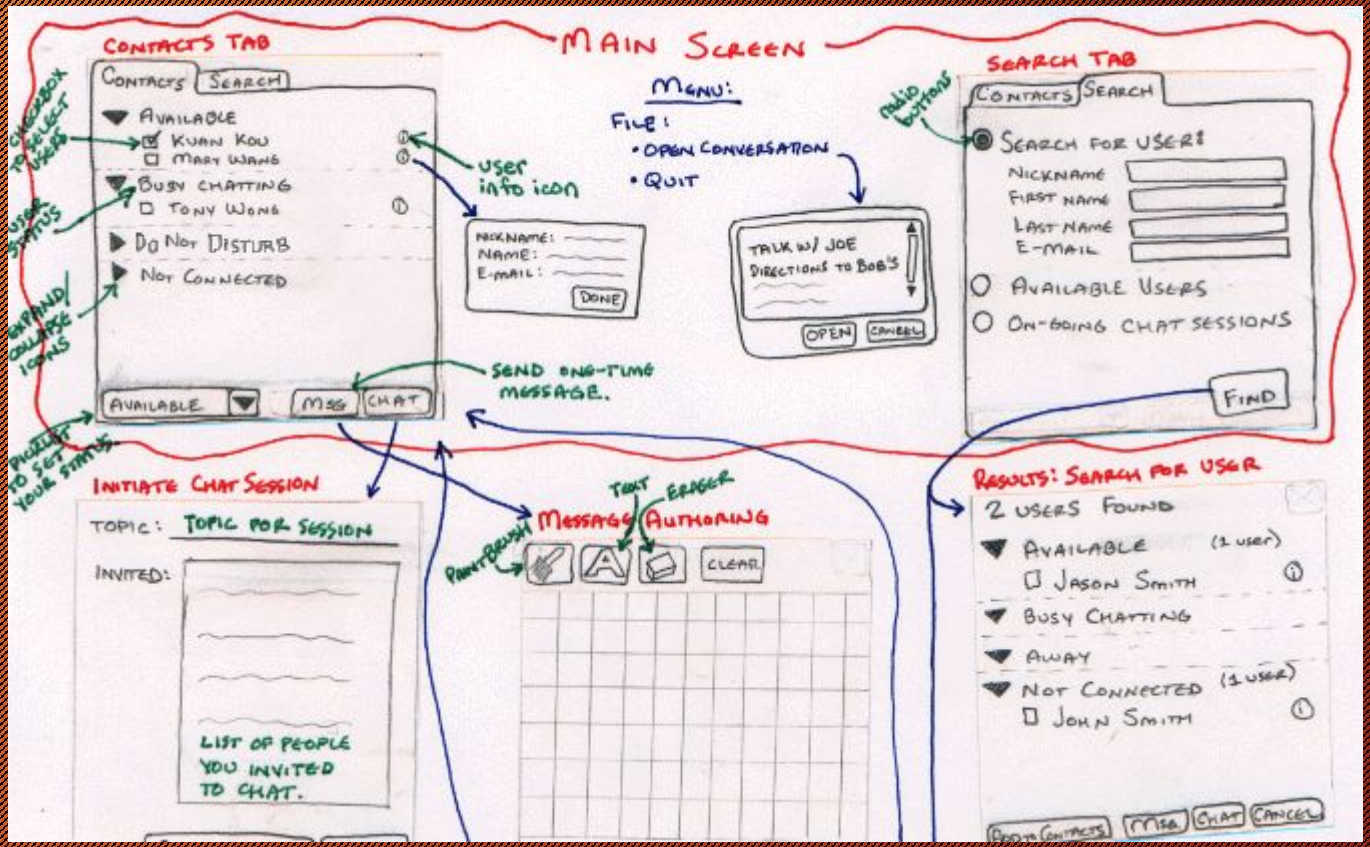
From CSM Thompson (2022)

User Interface Flow



From CO Mountain Mamas (2015)

User Interface Flow



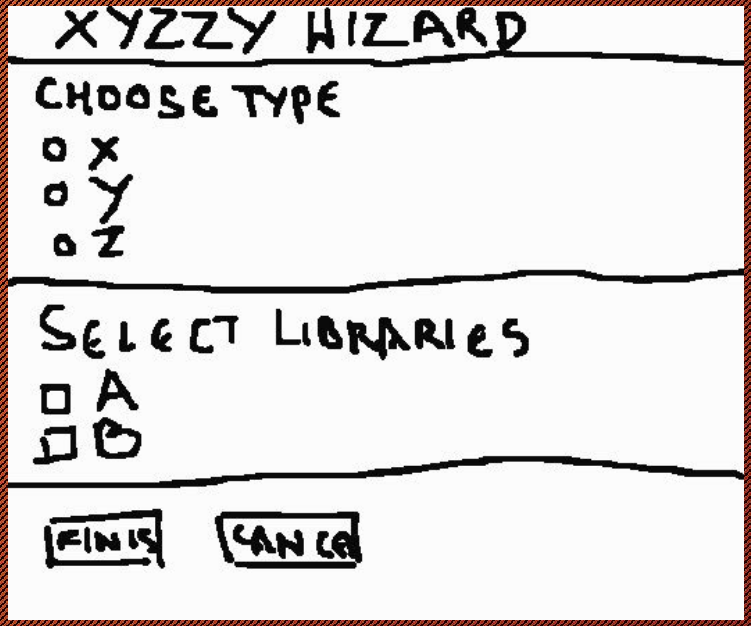
From CSM Thompson (2022)

Sketches and Prototypes

How sure are you?

Sketches

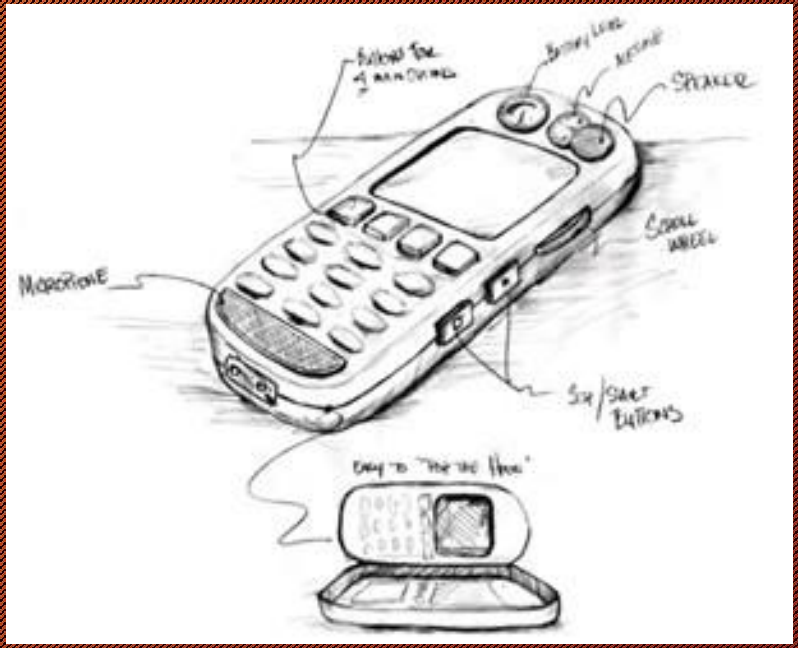
- Fast
- Cheap
- Numerous
- Ambiguous



Use when you're still figuring things out

Prototypes

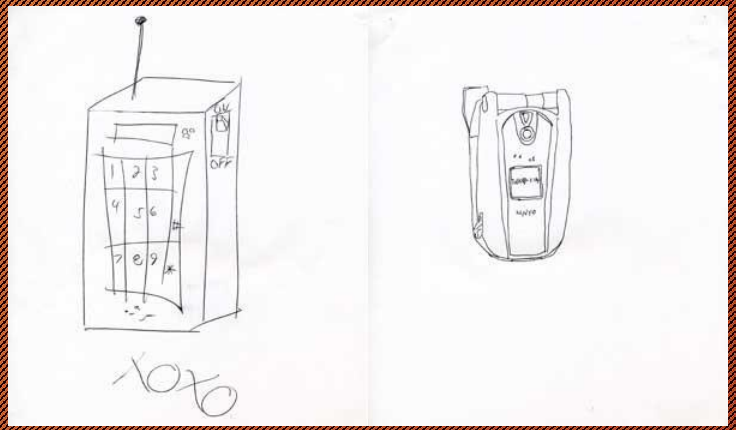
- Expensive (time or $)
- Formal
- Presentational
- Finalized



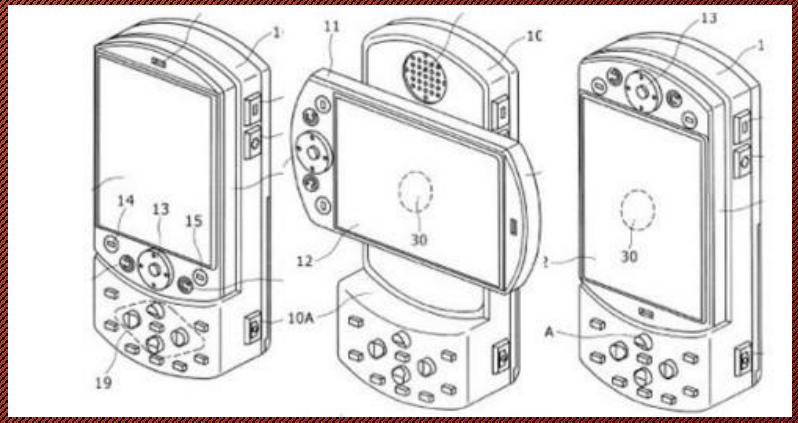
Use when you've got it mostly figured out

Sketches vs. Prototypes

- Questions to ask:
  - How sure are you about the design?
  - How deep into implementation are you?
  - What level of detail does your client want?
  - What is a good balance between cost and fidelity?
  - What audience are you presenting this to?



vs.



Your Turn

Design exploration

Design Presentation

- Dates/Times/Locations on website (Coming soon!)
  - Go to Schedule and look for Design Proposal Presentations (in an open week before Sprint 3)
  - Your team should be listed (if not, contact your advisor ASAP)
- Present to 2-3 other teams
  - Practice with an audience
  - Immediate feedback
- 12 minutes in length
  - Overview of project requirements
  - Overview of implementation design
- See website for rubric and other details

Brainstorm with Your Team

- Talk about your requirements
- Brainstorm how to represent the *architecture*
  - These will be reviewed during advisor meetings… goal here is just to ensure you have some ideas to flesh out

- This is the planning phase. For the final report you will update the architecture diagram and add text descriptions of all the components. For now you just need diagrams.
- You may also add more technical design details in the final report.