

Software Engineering for Data Scientists

Software Design

David Beck^{1,2}, Joseph Hellerstein^{1,3},
Bernease Herman¹, Daniel Jones³

¹eScience Institute

²Chemical Engineering

³Computer Science Engineering

University of Washington

October 30, 2018



Co-Locate With Your Project Team

- Choose one member who has a word processor running on their computer
 - MS Word, Libre Office, Google Docs





Software Design

"...specification of a software artifact, intended to accomplish goals, using a set of primitive components ..." [wikipedia]



Class Timeline

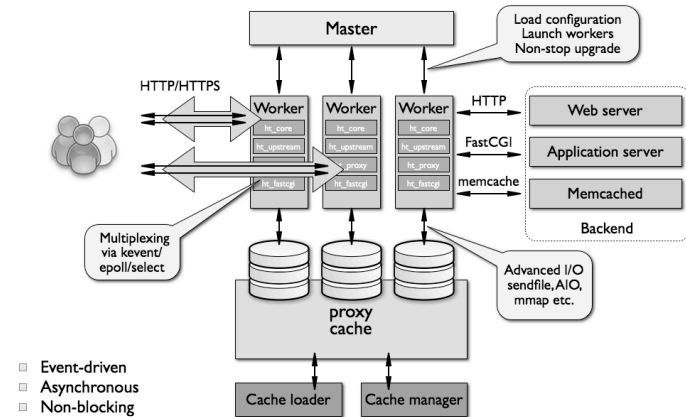
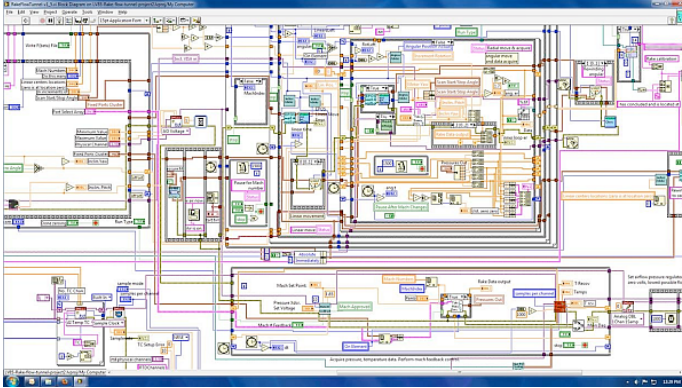
- 30 min: Lecture
 - Design: why & how
 - Use cases
 - Component specification
- 20 min: Team component specification (4 members)
- 10 min: Team reports



Design: Why & How



What Makes A Design Understandable?



- Few components with clear roles
 - Use abstraction hierarchies
- Few interactions between components
 - Carefully choose the features and interfaces
- Similarity with other designs
 - Use design patterns



Benefits of a Software Design

- Provides a systematic approach to a complex problem
- Finds bugs before you code
- Enables many people to work in parallel
- Promotes testability



Steps in Design

1. Functional design:

- Describe what the system does (use cases)

2. Component design: Specify the components

- Each use case has a "Top level" component.
- Sub-components implements portions of the use case
 - Ideally want many components that are common across use cases

Iterate. Iterate. Iterate.



Functional Design



Running Example: Design of ATM



What Do We Do With ATMs?



- Get cash
- Deposit checks
- Check balances
- These are examples of *Use Cases*.
- Implied use case – User authentication.



Describing a Use Case

- What information the user provides
 - E.g., command entered with its options
- What responses the system provides
 - E.g., prompts, plots, error messages

Authenticate User Use Case

User: Put ATM card in reader

ATM: Display 'Enter PIN'

User: Enters PIN on keyboard

ATM: [if correct] Show main menu
[if incorrect] Display 'Enter PIN'



Organization of Functional Specification

- Background. The problem being addressed.
- User profile. Who uses the system. What they know about the domain and computing (e.g., can browse the web, can program in Python)
- Data sources. What data you will use and how it is structured.
- Use cases. Describing at least two use cases
 - Objective (e.g., withdraw money from an ATM)
 - Expected interactions between the user and your system



Component Design



Specification of Components

- Describe components with sufficient detail so that someone with modest knowledge of the project can implement the code for the component.
 - Name
 - What it does
 - Inputs (with type information)
 - Outputs (with type information)
 - Interactions - how use other components



Developing Component Specifications

1. What are the components in the use cases?
2. What components are already available?
3. What are the sub-components needed to implement those components that aren't already available?

Do 1-2 for each such component



Example Component Specification

Find Primes $< N$

- Name
 - `FindPrimes`
- What it does:
 - Finds the primes that are less than N
- Inputs (with type information)
 - N , an integer
- Outputs (with type information)
 - List of integers



Describing Interactions With Pseudo Code

- Pseudo code for finding the primes $< N$ if the one component is divides
 - For $n < N$
 - `is_prime = True`
 - For $d < n$
 - If `divides(n, d)`
 - Then `is_prime = False`
 - If `is_prime`, add n to `list_of_primes`
- "Rules" for pseudo code
 - Consistent use of flow control & variable names
 - Readable English where details are sparse



ATM Components by Use Case

- Authenticate user
 - Database with user PIN
 - User interface that reads ATM card
 - User interface that reads user PIN
 - Control logic
- Get cash
 - Database with users cash balance
 - User interface that reads user cash requested
 - Cash drawer interface that dispenses cash
 - Control logic



The Next 20 Minutes

- Your charter: Design for Authenticate User
 - List of components of their interactions
 - Who calls whom for each step in the use case
 - Component(s) design using the design template
- What you'll report: Very brief
 - Summary of design
 - Issues, open questions



Team Component Specifications

20 min



Team Results

10 min



Report

- What we designed
 - Which components
 - What detail
- What issues encountered
- What do differently



Begin Project Functional Specification



Project Standups

- What you've accomplished to date
- Next steps
- Blockers

