Software Engineering for Data Scientists
Software Design

David Beck[1,2], Joseph Hellerstein[1,3], Jake VanderPlas[1,4]

[1]eScience Institute

[2]Chemical Engineering

[3]Computer Science Engineering

[4]Astronomy

University of Washington

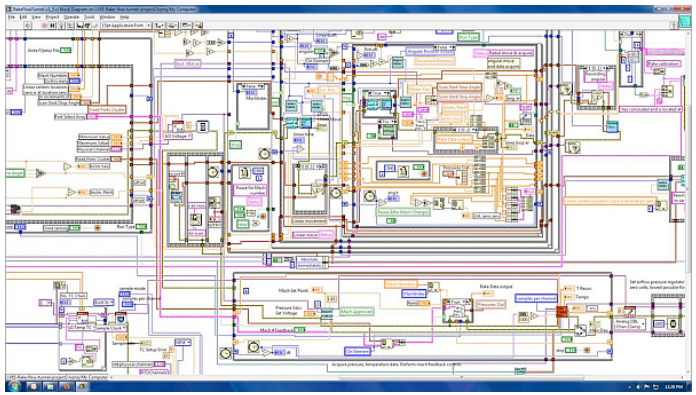
March 13, 2016

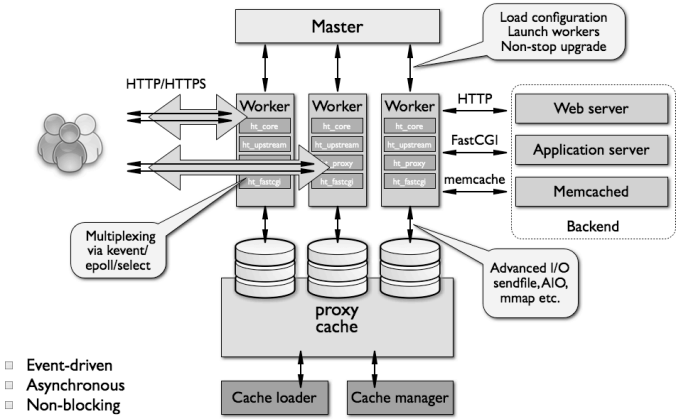
Beck, Hellerstein & VanderPlas, 2017

Software Design

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

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 Building Software

1. Describe what the system does (use cases)
2. 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.

Use Cases

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'

Component Specifications

"Functional" 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)
  - 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

Pseudo Code: "How Use Other Components"

- 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

Summary

- Use cases
  - Details of the user interactions with the system
- Design
  - Component specifications
    - Name
    - What it does
    - Inputs
    - Outputs
    - How it uses other components