A Design Recipe

EECS 230

Spring 2016

Good software design

- Correct
- Efficient
- Simple

Code isn’t just for computers

In practice, other people need to read it:

- Your boss

Code isn't just for computers

In practice, other people need to read it:

- Your boss
- Your colleagues

Code isn't just for computers

In practice, other people need to read it:

- Your boss
- Your colleagues
- Your successors

Code isn’t just for computers

In practice, other people need to read it:

- Your boss
- Your colleagues
- Your successors
- You in the future

A recipe

1. Problem analysis
2. Signature, purpose, and header
3. Examples
4. Strategy
5. Coding
6. (Testing)

Example

Goal: Write a function that sums a vector of doubles.

Step 1: Problem analysis

Step 1: Problem analysis

We need a function that takes a vector<double> and returns a double.

Step 2: Signature, purpose, header

```
// Sums a vector of doubles
double sum(vector<double> doubles)
```

Step 3: Examples

```
// Sums a vector of doubles

// Examples:
// - sum({}) == 0
// - sum({1, 2, 3, 4}) = 10

double sum(vector<double> doubles)
```

Step 4: Strategy

```
// Sums a vector of doubles

// Examples:
// - sum({}) == 0
// - sum({1, 2, 3, 4}) = 10

// Strategy: structural iteration
double sum(vector<double> doubles)
{
        ...

        for (double d : doubles)
                ... d ...

        ...
}
```

Step 5: Coding

```
// Sums a vector of doubles

// Examples:
// - sum({}) == 0
// - sum({1, 2, 3, 4}) = 10

// Strategy: structural iteration
double sum(vector<double> doubles)
{
    double result = 0;

    for (double d : doubles)
        result += d;

    return result;
}
```

Strategies

structural iteration iterate over an existing vector

Strategies

structural iteration iterate over an existing vector

generative iteration iterate producing results while some condition holds

Strategies

structural iteration iterate over an existing vector

generative iteration iterate producing results while some condition holds

domain knowledge translate non-programming knowledge into code

Strategies

structural iteration iterate over an existing vector

generative iteration iterate producing results while some condition holds

domain knowledge translate non-programming knowledge into code

function composition combine other functions to get the desired result

Strategy: structural iteration

```
result fun(vector<T> v, ...)
{
        ...
        for (T a : v)
                ...
        ...
}
```

Strategy: generative iteration

```
vector<T> fun(...)
{
        vector<T> result;

        while (...)
                ... result.push_back(...) ...

        return result;
}
```