Recap: software development process

Activities and steps
- Requirements engineering
- Design and architecture
- Implementation
- Verification and Validation
- Deployment and Maintenance

Recap: requirements engineering

The process of eliciting, analyzing, documenting, and maintaining requirements.

- Types of requirements:
  Functional requirements, Non-functional requirements, Additional constraints.
- Common mistakes and challenges:
  Implementation details instead of requirements, unclear scope, changing/evolving requirements.
- Possible strategies for eliciting requirements:
  Interviews, observations, use cases, user stories, prototyping

Recap: software architecture vs. design

Architecture and design
- Lower complexity: separation of concerns, well defined interfaces
- Simplifies communication, effort estimation, and progress monitoring

<table>
<thead>
<tr>
<th>Architecture: What is developed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller uses Model</td>
</tr>
<tr>
<td>View sees Client</td>
</tr>
<tr>
<td>Business logic layer</td>
</tr>
<tr>
<td>Presentation layer</td>
</tr>
<tr>
<td>Data access layer</td>
</tr>
<tr>
<td>DB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design: How are the components developed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client uses Model</td>
</tr>
<tr>
<td>Controller updates Model</td>
</tr>
</tbody>
</table>
Today

- More on activities and steps in a systematic software development process.
- Project pitches for possible class projects.

Software development process

Activities and steps
- Requirements engineering
- Design and architecture
- Implementation
- Verification and Validation
- Deployment and Maintenance

Back to the fridge

Activities and steps
- Requirements engineering
- Design and architecture
- Implementation
- Verification and Validation
- Deployment and Maintenance

Verification vs. validation

Verification (did we build it right)
- Does the software meet its specification
- Usually an internal process
- E.g., formal verification and software testing

Validation (did we build the right thing?)
- Does the specification reflect the client’s needs?
- Usually an internal and external process
- E.g., acceptance testing
Verification: static vs. dynamic analysis

**Static analysis**
- Reason about the program without executing it
  - Code/design reviews
  - Type checking of a compiler
  - Rule/pattern-based analysis
  - Formal verification

**Dynamic analysis**
- Execute the program and observe its behavior
  - Software testing
  - Profiling

**Static analysis: example**

```java
double avg(double[] nums) {
    int n = nums.length;
    double sum = 0;
    int i = 0;
    while (i<n)
        sum = sum + nums[i];
        i = i + 1;
    double avg = sum / n;
    return avg;
}
```

```java
double avg(double[] nums) {
    int n = nums.length;
    double sum = 0;
    int i = 0;
    while (i<n)
        sum = sum + nums[i];
        i = i + 1;
    double avg = sum / n;
    return avg;
}
```
Dynamic analysis: example

Software testing

double avg(double[] nums) {
  int n = nums.length;
  double sum = 0;
  int i = 0;
  while (i<n)
    sum = sum + nums[i];
  i = i + 1;
  double avg = sum / n;
  return avg;
}

A test for the avg function:

@Test(timeout=1000)
public void testAvg() {
  double nums = new double[]{1.0, 2.0, 3.0});
  double actual = Math.avg(nums);
  double expected = 2.0;
  assertEquals(expected, actual, EPS);
}

Summary: verification vs. validation

Verification (did we build it right)
- Does the software meet its specification
  - Static analysis
    - Reason about the program without executing it
  - Dynamic analysis
    - Execute the program and observe its behavior

Validation (did we build the right thing?)
- Does the specification reflect the client’s needs?