Technical Debt:
Unintentional Vs Intentional
*Hands On*

Christos Kotselidis & Bijan Parsia
COMP61511 Week 3
October 2018
Technical Debt

• Results from every day decisions we make
• Should I hack it? Should I push it?
• “There is always time for later corrections”
• …after all is just software…right?
Real-world use cases

- Unintentional Debt
  - Example 1: Push, push, push…

- Intentional Debt
  - Example 2: …don’t worry, we will fix it later…
**Example 1: Push, Push, Push…**

1. Code
2. Commit
3. Pull Request
4. Gate
5. Code Review
6. Merge (Push)
7. Regress

- Very structural
- Protection against “unintentional debt”
- …but does it??
Example 1: Push, Push, Push...

<table>
<thead>
<tr>
<th>File</th>
<th>Hash</th>
<th>Committed Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>commands.py</td>
<td>44ce3af</td>
<td>13 Jun</td>
</tr>
<tr>
<td>Delete MaxineBuildAndRunOnJenkins.txt</td>
<td>6228111</td>
<td>13 Jun</td>
</tr>
<tr>
<td>Create README.md</td>
<td>00e8ba6</td>
<td>13 Jun</td>
</tr>
<tr>
<td>Create README.md</td>
<td>07f9750</td>
<td>13 Jun</td>
</tr>
<tr>
<td>Create README.md</td>
<td>6d86878</td>
<td>13 Jun</td>
</tr>
<tr>
<td>Create README.md</td>
<td>ddee786</td>
<td>13 Jun</td>
</tr>
<tr>
<td>Create README.md</td>
<td>3096c4</td>
<td>13 Jun</td>
</tr>
<tr>
<td>README.md</td>
<td>253caed</td>
<td>13 Jun</td>
</tr>
<tr>
<td>README.md</td>
<td>3c569c0</td>
<td>13 Jun</td>
</tr>
<tr>
<td>README.md</td>
<td>2f56c66</td>
<td>13 Jun</td>
</tr>
<tr>
<td>README.md</td>
<td>354fd1b</td>
<td>13 Jun</td>
</tr>
<tr>
<td>README.md</td>
<td>0c8ecad</td>
<td>13 Jun</td>
</tr>
</tbody>
</table>
Example 1: Push, Push, Push...

1. Code
2. Commit
3. Pull Request
4. Gate
5. Code Review
6. Merge (Push)
7. Regress

Example 1:
Push, Push, Push…
Example 1: Push, Push, Push…

15% Electricity Increase

WAT?!?!?!?
Example 1: Push, Push, Push...

1. 13 pushes
2. 1 Regression per push
3. Oh no….
Example 1: Push, Push, Push…

Debt calculation (1/3)

- 1200 Watt regression machine
- 8 hours per regression
- 13 regressions (+1 after fix) = 14

Electricity Cost = KWhs * Price = 134.4 * 0.12 = 16 GBP (+tax/fees) = 18-20 GBP
Example 1: Push, Push, Push…
Debt calculation (2/3)

- Training Costs
- 2 persons (trainer + trainee)
- 1 hour training * 2 = 2 hours

**Training Cost** = PHs * Price = 
2 * 50 = 100 GBP (+insurance, etc.) = 170 GBP
Example 1: Push, Push, Push…
Debt calculation (3/3)

**Total Cost** = Training Cost * Electricity Cost = 190 GBP

• Excluding delay release costs
• …and other indirect costs…
Example 1: Push, Push, Push…

Solution

• Assess regression testing
• Understand trade-offs
  – Coverage vs operational costs
• Design new Continuous Integration (CI) Framework
Example 1: Push, Push, Push...

Solution

Vasilis Tsarnas 9:27 PM
@Christos Kotselidis no wonder why you had +15% electricity consumption by the server that runs Jenkins

Vasilis Tsarnas 9:28 PM
there are quite a lot of branches already as I can see

1 reply Today at 9:15 AM

Christos Kotselidis 2 minutes ago
LOL, that's a screenshot for next year's course! 😄
Example 1: Push, Push, Push…

Conclusions

• Simple SW malpractices can lead to:
  – Unintended Debt
  – Delays
  – Increased Operational Costs

• Solution?
  – Compulsory pre-push code reviews
  – More training on how to use tools
  – Hooks to check per user frequent pushes
Example 2: …don’t worry, we will fix it later…

- **Disclaimer:** I take full responsibility for that.
Example 2: …don’t worry, we will fix it later…

- A method that compiles a function for a specific architecture (ARM)

```java
/**
 * This method compiles an input .c file to any ARM target architecture
 */
public void compile() {
    final ProcessBuilder compile =
        new ProcessBuilder("arm-none-eabi-gcc", 
                         "-march=armv7-a", 
                         "-g", 
                         "test_armv7.c", 
                         "-o", 
                         "test_armv7.o");
    compile.redirectOutput(gccOutput);
    compile.redirectError(gccErrors);
    try {
        gcc = compile.start();
        gcc.waitFor();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```
Example 2: ...don’t worry, we will fix it later...

- **Inconsistent comment**: not any ARM, just ARMv7
- **Hardcoded constants|flags|etc.**: e.g. “arm-none-eabi-gcc”
- **Wrong class name**: MaxineARMTester → What if we want to implement ARMv8?

  ...don’t worry, we will fix later...

```java
/**
 * This method compiles an input .c file to any ARM target architecture
 */
public void compile() {
    final ProcessBuilder compile =
        new ProcessBuilder("arm-none-eabi-gcc", "-march=armv7-a", "-g", "test_armv7.c", "-o", "test_armv7.o");
    compile.redirectOutput(gccOutput);
    compile.redirectError(gccErrors);
    try {
        gcc = compile.start();
        gcc.waitFor();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```
Example 2: Past and Future

- **First commit (April 2014):**

  - Commits on Apr 29, 2014
    - Split arm t1x, asm unit tests, parameterize the simulator
    - kotsolidis committed on 29 Apr 2014

- **Fast Forward to October 2017:**
  - Hey! Let’s do the ARMv8 Port!
  - All the infrastructure is ready!
  - We just plug in the ARMv8 tools to MaxineTester
  - Maximum a 2 hours job!
  - **YEAH! LET’S DO IT!!!!!**
Example 2: Intentional Debt Ghosting

- “Ooops! Sorry team, I forgot…”
  - “back then, I hardcoded everything”
  - “also the class is not properly designed, so now we have to split in two”
  - “oh, we also need to refactor all ARMv7 tests to use new class!”
  - “oh no!!! We also need to create two separate paths of execution!”
  - “…oh….oh…oh”
Example 2: Intentional Debt Ghosting

- Add missing method
- Code cleanup and documentation
- Code cleanup
- Remove obsolete method
- Remove obsolete method
- Fix package naming
- Add new method for parsign register file from qemu error file
- Fix package naming, add b test in aarch64 assembly tests
- Fix package naming
- Split asm test packages between armv7 and aarch64
- Fix input gdb script for aarch64
- Add correct implementation of startup.s in aarch64
Lessons Learned

• Example 1: push, push, push…
  – Unintentional debt can appear even if everything looks OK!
  – Increased Operational Costs
  – Attention to details!!!

• Example 2: “…don’t worry we will fix it later…”
  – Intentional debt underestimated
  – Kept lurking in the code for 3 years!
  – And finally got its revenge
  – Attention to debt assessment!!!
Conclusions

• **Debt no matter if intentional or unintentional**
  – Is usually there
  – It creates costs: money, work effort, etc.

• **Aim for quality code**
  – Always justify your intentional debt
  – Risk assessment is critical