Technical Debt:
Unintentional Vs Intentional
Hands On
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?
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...
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...

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
- ...other indirect costs...
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 test file to any ARM available system.
 */
public void compile() {
    final ProcessBuilder removeFiles = new ProcessBuilder("/bin/rm", "-rR", "test_armv7.bin", "test_armv7.bin");
    compile.redirectOutput(gccOutput);
    compile.redirectError(gccErrors);
    try {
        removeFiles.start().waitFor();
        gcc = compile.start();
        gcc.waitFor();
    } catch (Exception e) {
        System.err.println(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 test file to any ARM available system.
 */
public void compile() {
    final ProcessBuilder removeFiles = new ProcessBuilder("/bin/rm", "-R", "test_armv7.bin", "test_armv7.bin");
    final ProcessBuilder compile = new ProcessBuilder("arm-none-eabi-gcc", "-c", "-DSTATIC", "-mfloat-
    compile.redirectOutput(gccOutput);
    compile.redirectError(gccErrors);
    try {
        removeFiles.start().waitFor();
        gcc = compile.start();
        gcc.waitFor();
    } catch (Exception e) {
        System.err.println(e);
        e.printStackTrace();
    }
}
```
Example 2: ...then...now...

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

- **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, yeah.., also we 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

<table>
<thead>
<tr>
<th>Issue</th>
<th>Author</th>
<th>Date</th>
<th>Commit Hash</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add missing method</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>e3d4db1</td>
<td></td>
</tr>
<tr>
<td>Code cleanup and documentation</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>e42eb10</td>
<td></td>
</tr>
<tr>
<td>Code cleanup</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>b68e9dd</td>
<td></td>
</tr>
<tr>
<td>Remove obsolete method</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>4c8501c</td>
<td></td>
</tr>
<tr>
<td>Remove obsolete method</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>d6924b3</td>
<td></td>
</tr>
<tr>
<td>Fix package naming</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>577d1fc</td>
<td></td>
</tr>
<tr>
<td>Add new method for parsign register file from qemu error file</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>0e75279</td>
<td></td>
</tr>
<tr>
<td>Fix package naming, add b test in aarch64 assembly tests</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>562fe4d</td>
<td></td>
</tr>
<tr>
<td>Fix package naming</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>0262eb4</td>
<td></td>
</tr>
<tr>
<td>Split asm test packages between armv7 and aarch64</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>2a4d5c8</td>
<td></td>
</tr>
<tr>
<td>Fix input gdb script for aarch64</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>542de5d</td>
<td></td>
</tr>
<tr>
<td>Add correct implementation of startup.s in aarch64</td>
<td>kotseidis</td>
<td>10 days ago</td>
<td>f13f95c</td>
<td></td>
</tr>
</tbody>
</table>
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!
  – ...they mostly, come back to you...
  – Attention to debt assessment!!!
Conclusions

• Debt no matter if intentional or unintentional
  – Is usually there
  – It creates costs: sometimes monetary, sometimes work-wise, or both

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