Leveraging Maven 2 for Agility

About Tim
About Luke

Iowa Student Loan

Established 1979
Private, Non-profit corporation
ISL's Mission

"To help Iowa students and families obtain the resources necessary to succeed in postsecondary education"

Extreme Programming at ISL
When we used Ant...

- Long running builds
- Complex builds
- Inconsistent build approaches
- Build logic duplication
- Dependency management overhead

Spending lots of time focused on build process
Long Running Builds

New Ant target "quick-build"

+ faster feedback
- more Teamcity breaks
More build breaks

<table>
<thead>
<tr>
<th>Build Breaks</th>
<th>Alpha</th>
<th>Gamma</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBoss Out of Memory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beginner BH Fail Tests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Need BFP &amp; NC for QA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVN Locked/Unlocked</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fatality of the System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Running Config</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firefox, Not Running/Auto</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database Incompatible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing Problems on OS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNS server died</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flaky Test</td>
<td>3/9</td>
<td>3/9</td>
</tr>
</tbody>
</table>

Firefighter
fix or improve builds
Explaining decline in Velocity

Wrote build times on whiteboard

<table>
<thead>
<tr>
<th>TESTS</th>
<th>Alpha</th>
<th>Genesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT</td>
<td>10</td>
<td>35</td>
</tr>
<tr>
<td>DB</td>
<td>12</td>
<td>36</td>
</tr>
<tr>
<td>-BROWSER</td>
<td>50</td>
<td>200</td>
</tr>
<tr>
<td>-BROWSER</td>
<td>270</td>
<td>320</td>
</tr>
<tr>
<td>64 COMMIT</td>
<td>195 (3% min)</td>
<td>432 (7% min)</td>
</tr>
<tr>
<td>ALL</td>
<td>660</td>
<td>1200 (20 min)</td>
</tr>
</tbody>
</table>
Used Hypersonic

Identified longest-running tests and re-wrote them each iteration
Replaced slow, brittle browser tests with user story tests

Watij => FitNesse

complex builds
clever tricks

search-and-replace to set a configuration token from the value in a properties file

```xml
<target name="env" depends="init">
  <copy file="$build.env.property.file" overwrite="true"/>
  <filenames file="$env.properties.file"/>
</copy>

<replace file="$build.env.property.file">
  <replaceAll token="dev.database1" value="$(ENV.DEV_DATABASE1)"/>
</replace>

<replace file="$build.env.property.file">
  <replaceAll token="COMPUTERNAME" value="$(ENV.COMPUTERNAME)"/>
</replace>

<propertyfile file="$build.env.property.file"/>

<copy file="$build.classes.dir" overwrite="true">
  <filenames file="$build.env.property.file"/>
</copy>
</target>
```
automate everything

build logic duplication
added PMD & Cobertura

seemed wrong to duplicate build logic in order to detect duplication

learning a new build.xml
build improvements gone awry

+ innovation
- inconsistency
- learning premium

inconsistency

We created “base-webapp” to provide a good template of the structure for new projects
Maybe Agile isn't scalable

Maybe consistency isn't so important

JUnit test to enforce Ant common Ant targets
Dependency Management Overhead

Third-party libraries
IntelliJ IDEA Grievances

```java
public class AlphaBuildProjectUnitTest extends AbstractBuildProjectUnitTest {
    public void setUp() throws Exception {
        setProjectPath(".");
        setProjectXMLFileName("alpha-web.xml");
        setBuildXMLFileName("build.xml");
        setClamspachMainPropertyList(new String[] {"clamspach.main.jar"};
        setClamspachTestPropertyList(new String[] {"clamspach.test.jar"};
        super.setUp();
    }
}
```

Painful to upgrade/add new dependencies
Upgrading took time, more likely to be pushed off until the infamous "later"

shared code

big ball of mud
Webflow Upgrade Day!
Took a coordinated effort

Enter Maven 2
Appeal of Maven

Resistance
Complacency with Ant

Nobody wanted to rewrite the build
Convincing Developers

- Started with compromise
- Timebox an hour a day
- Make both builds green
- Show by pairing

Cross Team Pairing

- Show other teams better way
- Book club
- Always available assistance from other teams
Migration from Ant to Maven

Organizational Concerns
Explaination to Business Stakeholders

Developers make technical decisions
Transparency is essential

Being transparent about our decisions builds trust between developers and business
Explanation to Business Stakeholders

The main points were that Maven:
- provides standards and consistency
- reduces overhead
- makes developers more efficient

Learning Curve
Infrastructure Needs

archiva

Benefits
Benefits

The benefits we expected to get from Maven were:
- consistency across projects
- simpler builds
- reduced build logic duplication
- improved dependency management
Change Impact

- Localized changes
- Faster Builds
- No fear
- Overhead of creating modules is smaller

Tools Impact

- IntelliJ, Eclipse -- all use the POM
- Easy to add metrics
Code Impact

- Cleaner code
- SOLID for modules

Social Impact

Optimism replaced negative feelings towards sharing code

- More cross-team collaboration
Business Impact

- Able deliver change predictably and faster

Unexpected Benefits

Positive side effects of module extraction
- Cleaner code
- Our code is treated as first class libraries/binaries
Conclusion

Maven introduces best practices
<table>
<thead>
<tr>
<th>Problem</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long running builds</td>
<td>Extract modules (small builds, not gargantuan builds)</td>
</tr>
<tr>
<td></td>
<td>Binary versions available</td>
</tr>
<tr>
<td>Build breaks (fragile builds from not running all tests because it took too long)</td>
<td>Extracted modules and version freezes made more stability back to one authoritative build (per module)</td>
</tr>
<tr>
<td>Complex builds</td>
<td>Using parent project POM</td>
</tr>
<tr>
<td>Inconsistent build approaches</td>
<td>Standard Build Lifecycle</td>
</tr>
<tr>
<td></td>
<td>Maven Conventions</td>
</tr>
<tr>
<td></td>
<td>archetype:generate</td>
</tr>
<tr>
<td>Build logic duplication</td>
<td>Standard Build Lifecycle</td>
</tr>
<tr>
<td></td>
<td>Maven Plugins</td>
</tr>
<tr>
<td>Dependency management overhead</td>
<td>Version Management</td>
</tr>
<tr>
<td></td>
<td>Binary deployment</td>
</tr>
<tr>
<td></td>
<td>Transitive dependencies</td>
</tr>
</tbody>
</table>

Maven enhances Agile by

- removing distractions
- embracing modularity
- focusing on code
- lowering overhead
Maven made us focus more on code and not on build process.

Less distraction, more business value.
References

Better Builds with Maven
How to Guide for Maven 2.0

Maven
The Definitive Guide

Questions