Controlling Your Architecture

Magnus Robertsson

Jayway

About me

:: Loves coding

- Have been coding just about everywhere
- Code can be beautiful!

:: Loves the Digital Home

- Have 5 km of wires in my home!

:: Worked with Java since 1996

:: Passionate OpenSource contributor

- http://www.aspectme.org/

Agenda

:: How do we build systems that last?

:: Ways of enforcing architectural rules

:: How to enforce your architecture

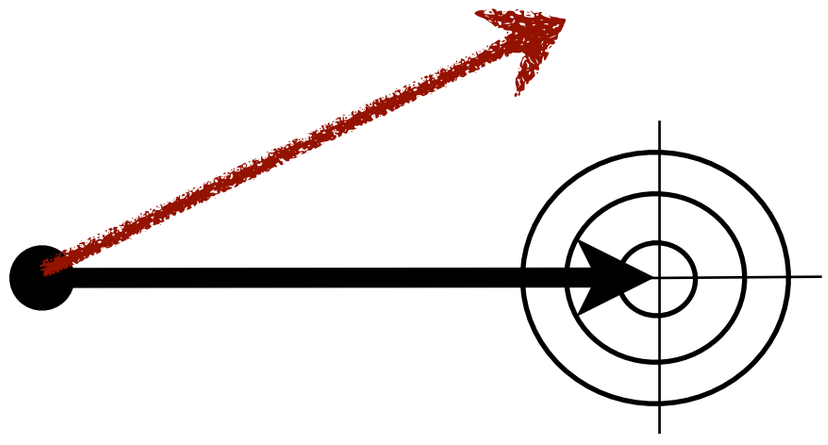
Jayway

Why do systems become ”big balls of mud”?

:: Architectural drift

- The result of an architecture that isn’t *communicated* and/or *enforced*
- Causes code smells, technical debt, anti-pattern architectures (big-ball-of-mud, stovepipe, etc)

- The way we enforce architectural rules is bad!
  - Possible root causes; resistance of change or ignorance



jayway

Ways of enforcing architectural rules

:: Manual

- Code inspection by experienced developers (possibly documents)
- ”This code smells, fix it”
- ”This code doesn’t follow the guidelines”

:: Static

- Based on the structure of the code
- ”Do not let the view layer call JDBC directly”

:: Dynamic

- Based on the program flow
- ”Only one thread at the time is allowed in method `foo(int)`”

jayway

Code rulez!

:: The most important document we have

:: Why not express architectural rules in code and/or meta-data?

- Use tools to enforce them

:: Simply put, lets automate the enforcement of architectural rules!

Static code analysis

- FindBugs
- Checkstyle
- Cobertura
- Emma
- Structure101
- SonarJ
- Sonar
- CodeCity
- ...
- (AspectJ)

Example: FindBugs

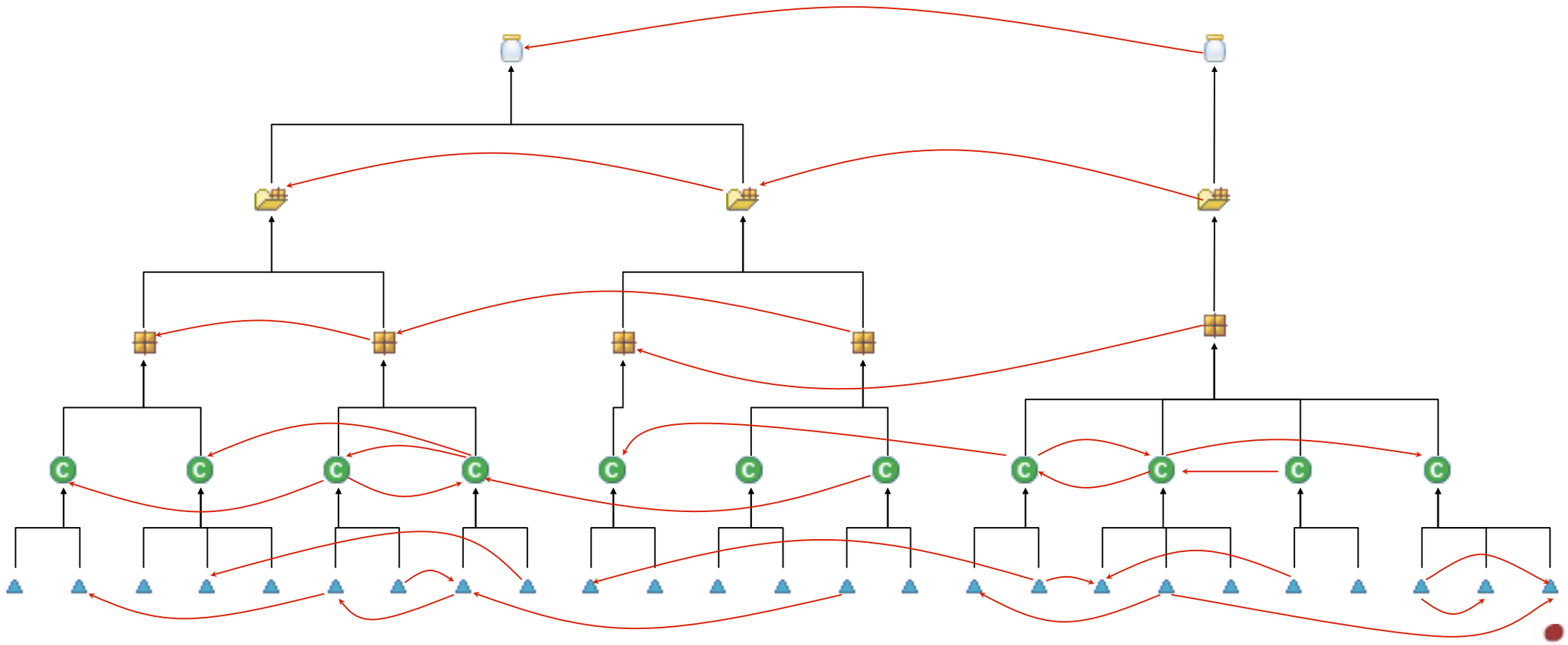
:: Look for bug patterns, i.e. a code idiom that often result in an error

- Malicious code vulnerability
- Bad practice
- Internationalization
- Multithreaded correctness
- Dodgy
- Correctness
- Performance

Example: Structure101

:: Measures structural complexity (XS) by

- Fat packages, classes, methods
- Tangled packages and classes



jayway

Example: AspectJ

:: ”Straight SQL communication must not be used. Only use JdbcTemplate.”

Dynamic code analysis

- Assert

- AspectJ

Example: AspectJ

:: ”Flush Hibernate session before a JDBC operation”

:: ”Only one thread is allowed when accessing this resource”

So, what can I do within my project?

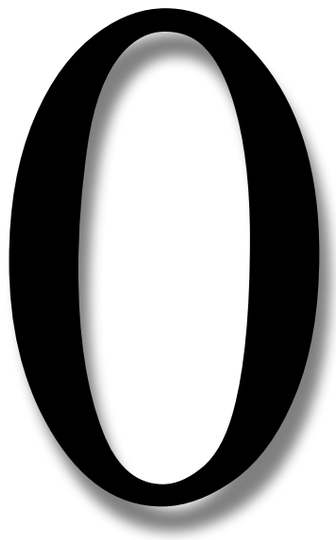
:: Use Structure101 and FindBugs to do an architectural review

- Non-intrusive and easy to do

:: Introduce FindBugs and Structure101 (or similar) to the build system

- First for information only (i.e. warnings)
- Then change to errors...

:: Add aspects for dynamic analysis at test and/or runtime



[Zero]

jayway

Our vision

- Zero XS (technical debt) is possible for large systems!
- Zero FindBugs errors/warnings a must
- Zero compiler warnings is easy
  - Learn to hate the yellow triangle in Eclipse
- … and Zero errors will be easier to achieve!

- Easily ignored, if more than 10 warnings...
  - Have to be diciplined

Summary

:: Reference architectures are good,
the way we enforce them is bad

:: Tools exist for enforcing your architecture. Start today!

- No more ”Word architectures”

:: Be disciplined! Fix issues as they arise.

Certified Java Professionals

www.jayway.se