



Relaxed Coding of Quality Software

The Benefits of Static Code Analysis





About me

- ▶ Java developer for 15 years
- ▶ Working for hello2morrow
 - ▶ [Sonargraph Architect](#) (Java + SWT, Maven, Ant)
 - ▶ [Jenkins Plugin](#)
 - ▶ [SonarQube Plugin](#)
 - ▶ [Sonargraph Explorer](#), Eclipse RCP
- ▶ Interested in coding best practices



Questions to be answered

- ▶ What is quality software?
- ▶ Why are we not relaxed?
- ▶ What can we do about it?
- ▶ Why should we care?



Quality aspects

Functional, observed at runtime:

- ▶ Functional correctness
- ▶ Performance
- ▶ Security
- ▶ ...

Non-functional, „embodied in the static structure of the software system“:

- ▶ *ilities: Maintainability, extensibility, testability, scalability, modularity, ...

See: http://en.wikipedia.org/wiki/Non-functional_requirement

Things adding stress to a developers life

Outside world

- ▶ Unclear requirements
- ▶ Deadlines
- ▶ Evolution of frameworks, changes in API
- ▶ Changing team members
- ▶ Changing priorities

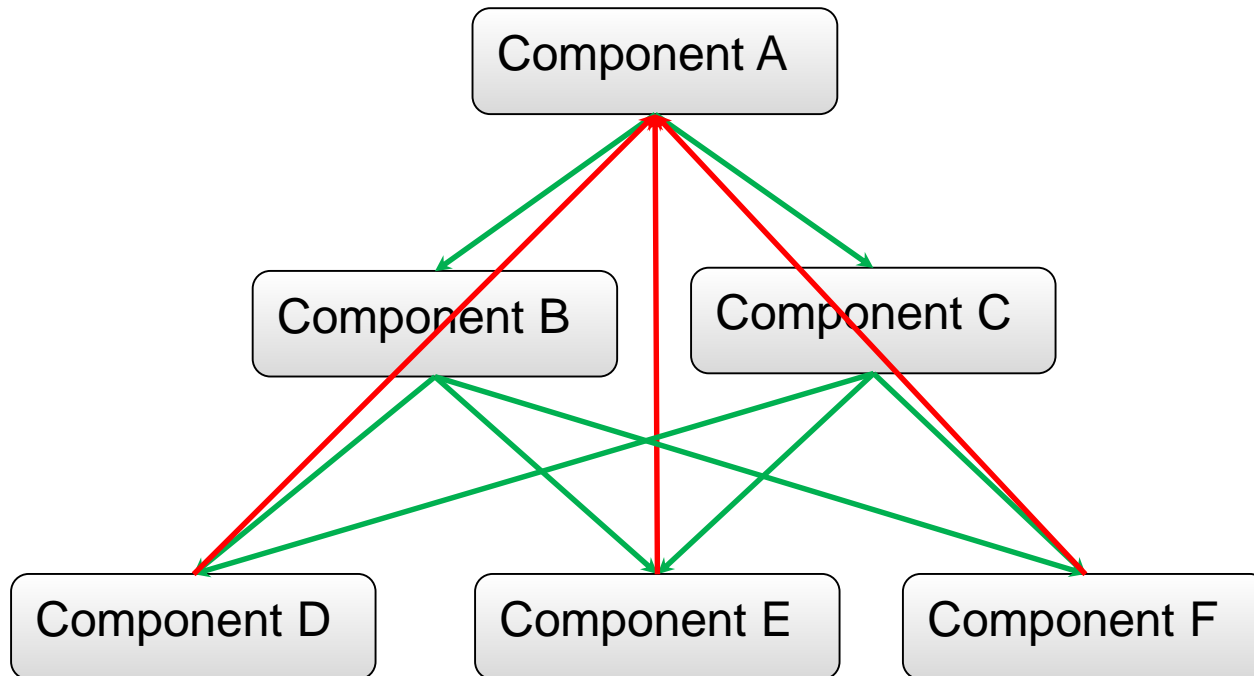
Inside world

- ▶ Complexity of code base, aka „Spaghetti-Design“
- ▶ Bad distribution of complexity
- ▶ Bad test coverage
- ▶ Bugs and potential bugs
- ▶ Code duplication
- ▶ Missing coding standards

No / too much / outdated / useless documentation

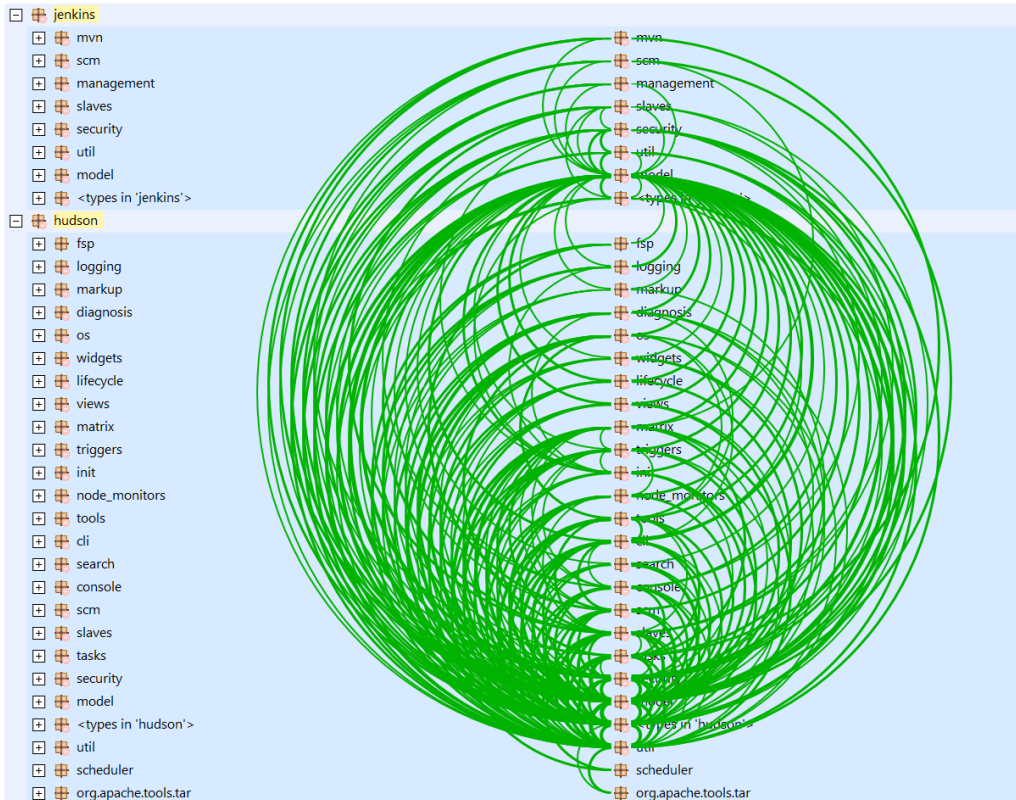
See: <http://docs.sonarqube.org/display/SONAR/Developers%27+Seven+Deadly+Sins>

Complexity increase



“It is the dependency architecture that is degrading, and with it the ability of the software to be maintained.” (Robert C. Martin)

Software Erosion – Symptoms

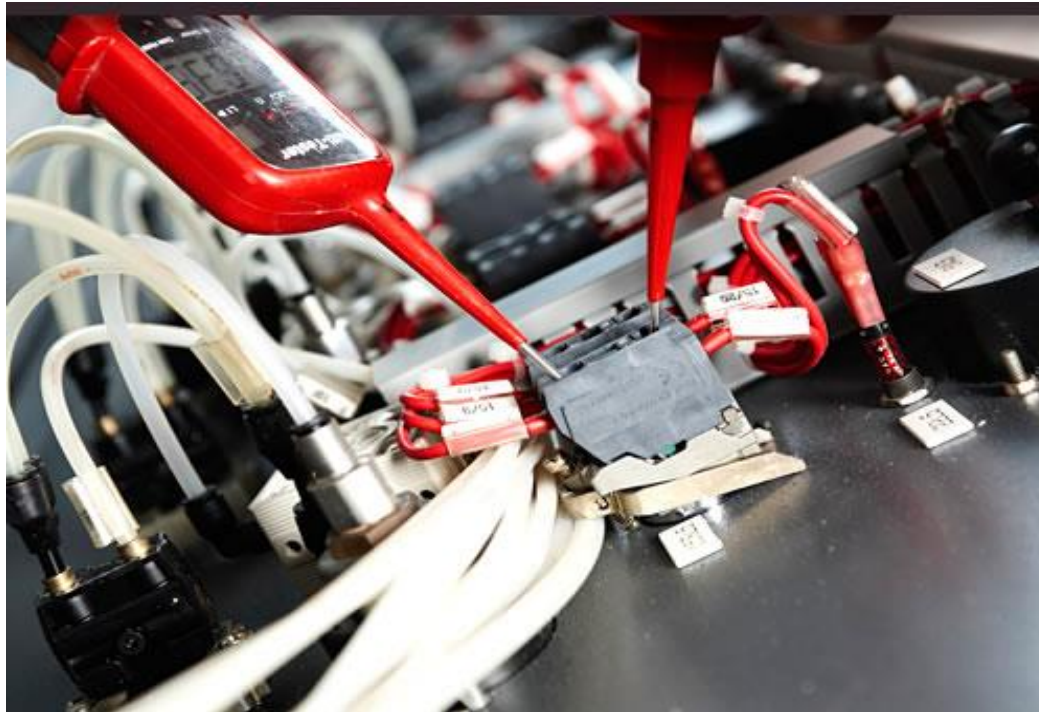


- ▶ Immobility
- ▶ Opacity
- ▶ Fragility
- ▶ Rigidity
- ▶ Viscosity

(Robert C. Martin)



Get back into control...



“You can’t manage what you can’t control, and you can’t control what you don’t measure” (Tom DeMarco)



What we can do about it

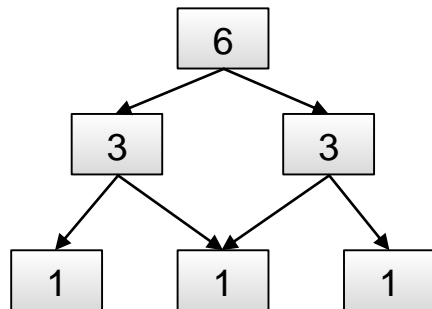
Improve our toolchain and use static code analysis to **automatically**

- ▶ ... monitor the complexity of the code base at macro level: Detect architecture violations, cyclic dependencies between packages, control overall coupling
- ▶ ... control the distribution of complexity at micro level: Control cyclomatic complexity of methods, lines of code in source file, number of parameters, etc.
- ▶ ... detect missing test coverage
- ▶ ... find bugs and potential bugs
- ▶ ... find code duplication
- ▶ ... check for violations of coding standards

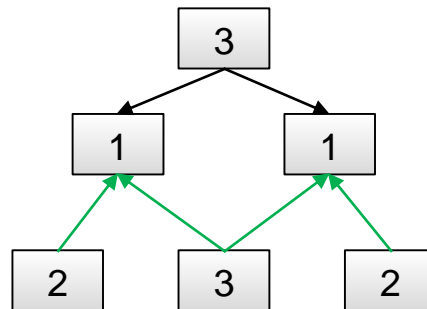


Metrics for Coupling (John Lakos)

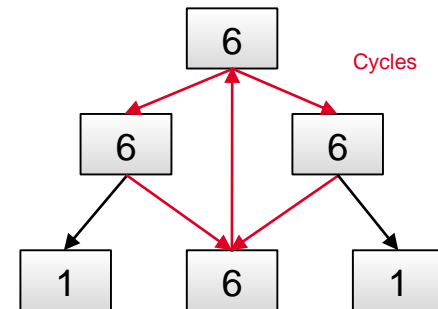
- ▶ Depends upon:
The number a component directly and indirectly depends upon (+1 for itself)
- ▶ ACD (Average Component Dependency):
The sum of all depends upon values divided by the number of components



$$\text{ACD} = 15/6 = 2,5$$

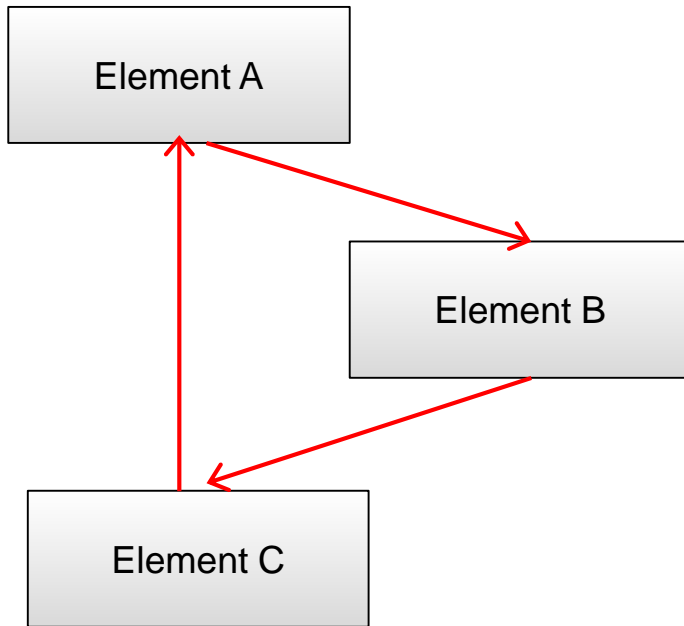


Dependency Inversion
 $\text{ACD} = 12/6 = 2$

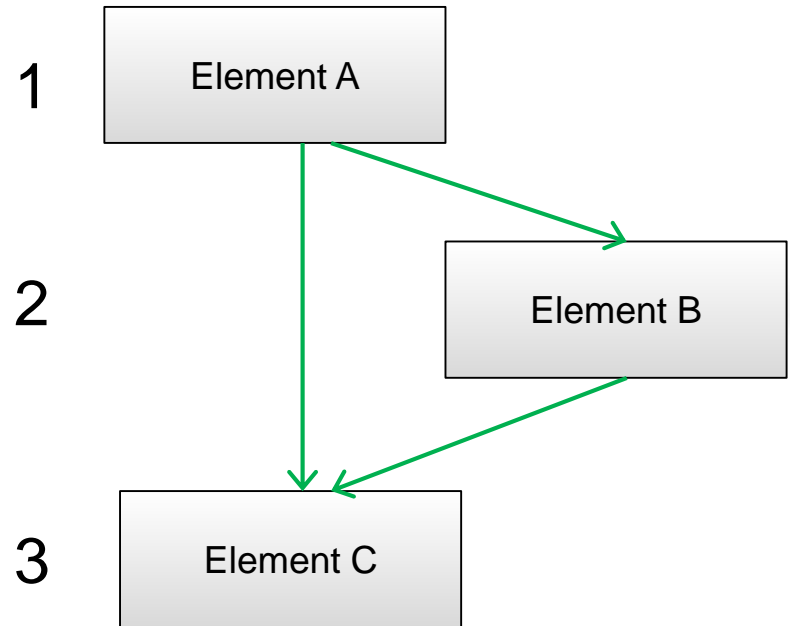


$$\text{ACD} = 26/6 = 4,33$$

Impact of cycles

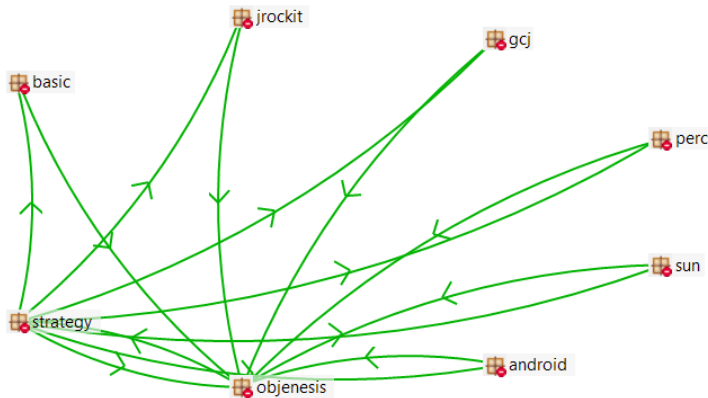


Level



Example for Structural Quality

▶ Spring 4.0.0

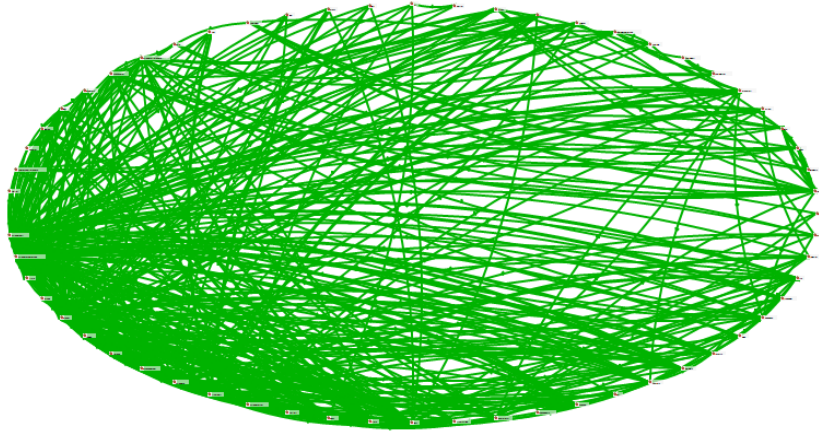


- ▶ Consists of > 20 projects
- ▶ 359 packages, 4519 types
- ▶ 12 packages are involved in cycles
- ▶ 3 package cycle groups
- ▶ Biggest cycle group: 8 Packages
- ▶ ACD: 27
- ▶ NCCD: 4.4



Example for Structural Erosion I

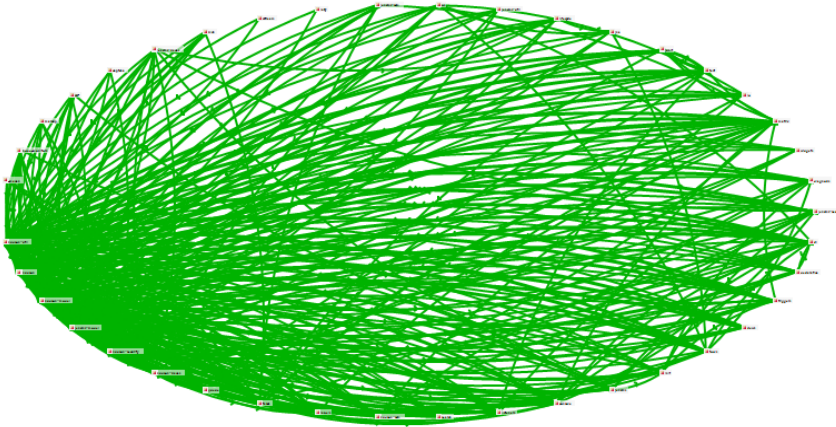
▶ ActiveMQ 5.5.1



- ▶ 122 packages, 2352 types
- ▶ 66 packages are involved in cycles
- ▶ 4 package cycle groups
- ▶ Biggest cycle group: 59 Packages
- ▶ ACD: 395
- ▶ NCCD: 41.2

Example for Structural Erosion II

▶ Jenkins Core 1.512

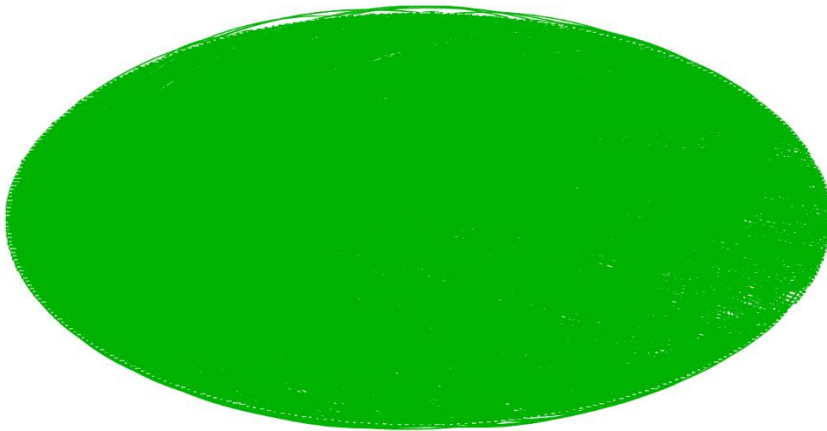


- ▶ 62 packages, 2090 types
- ▶ 41 packages are involved in cycles
- ▶ 1 package cycle group
- ▶ Biggest cycle group: 41 Packages
- ▶ ACD: 445
- ▶ NCCD: 49.8



Example for Structural Erosion III

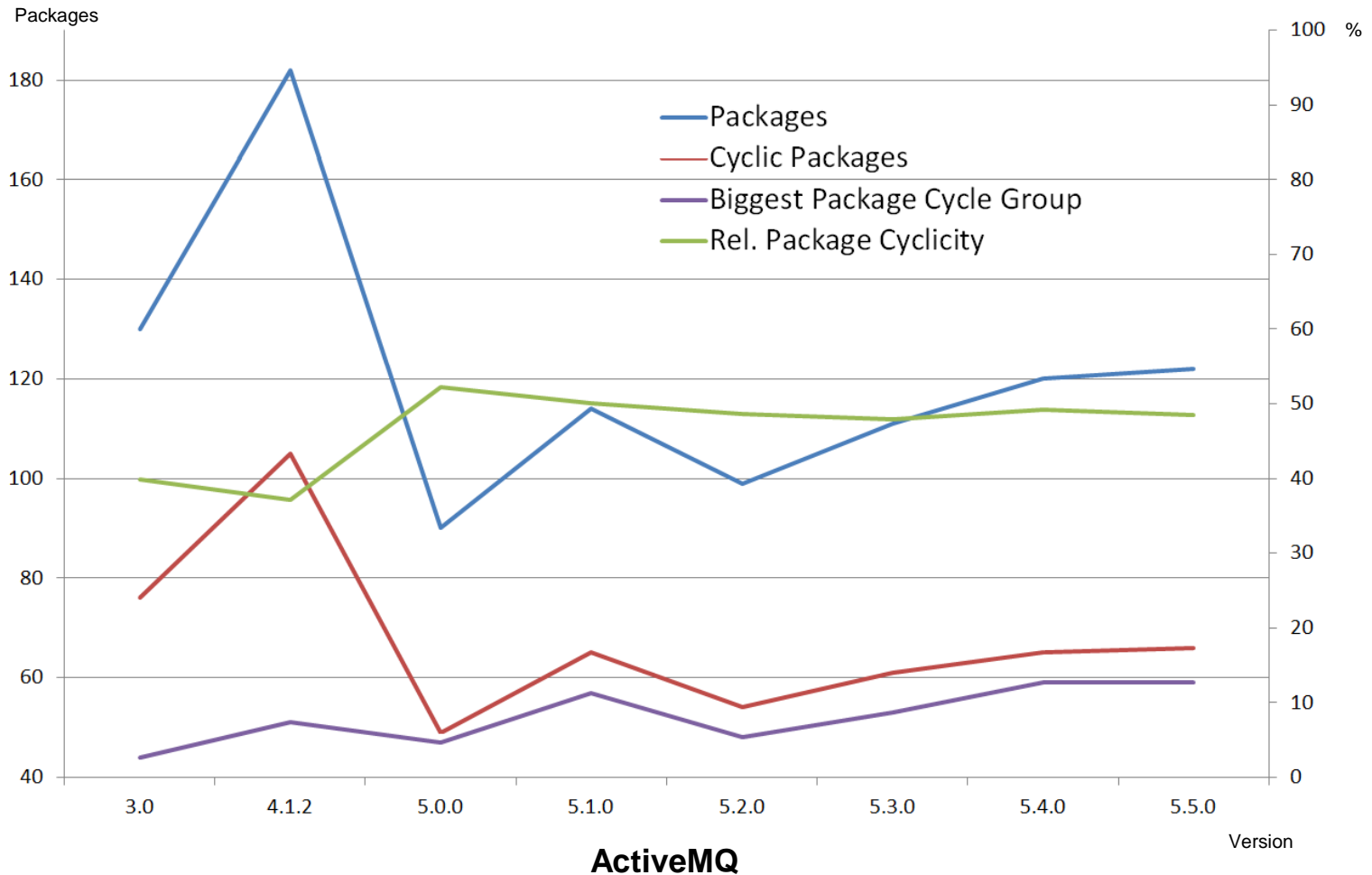
▶ JDK 1.7



- ▶ 852 packages, ~ 19 500 types
- ▶ 681 packages are involved in cycles
- ▶ 36 package cycle groups
- ▶ Biggest cycle group: 346 Packages
- ▶ ACD: 1097
- ▶ NCCD: 92.9



Package Cycles over Time

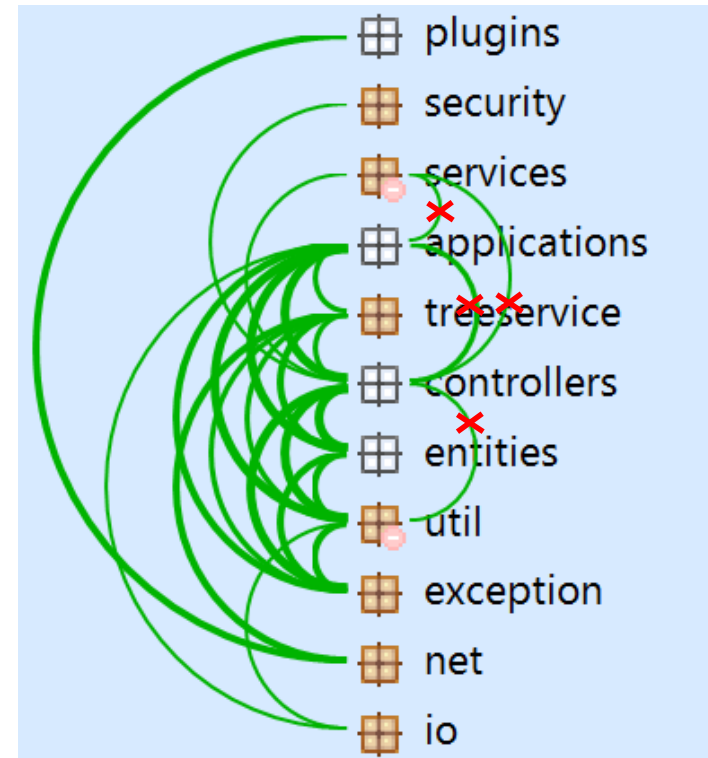


Structural Debt Index

This metric gives an idea for the required effort to clean up the dependency structure.

Calculation:

- ▶ Packages with more outgoing dependencies are above packages with more incoming dependencies
- ▶ Packages that are part of package cycle groups are sorted by calculating the difference between outgoing and incoming dependencies. Special rules for draws.
- ▶ All upward going dependencies are considered bad
- ▶ $SDI = 10 * (\text{type dependencies to cut}) + (\text{code refs of dependencies to cut})$





Structural Debt Index - Examples

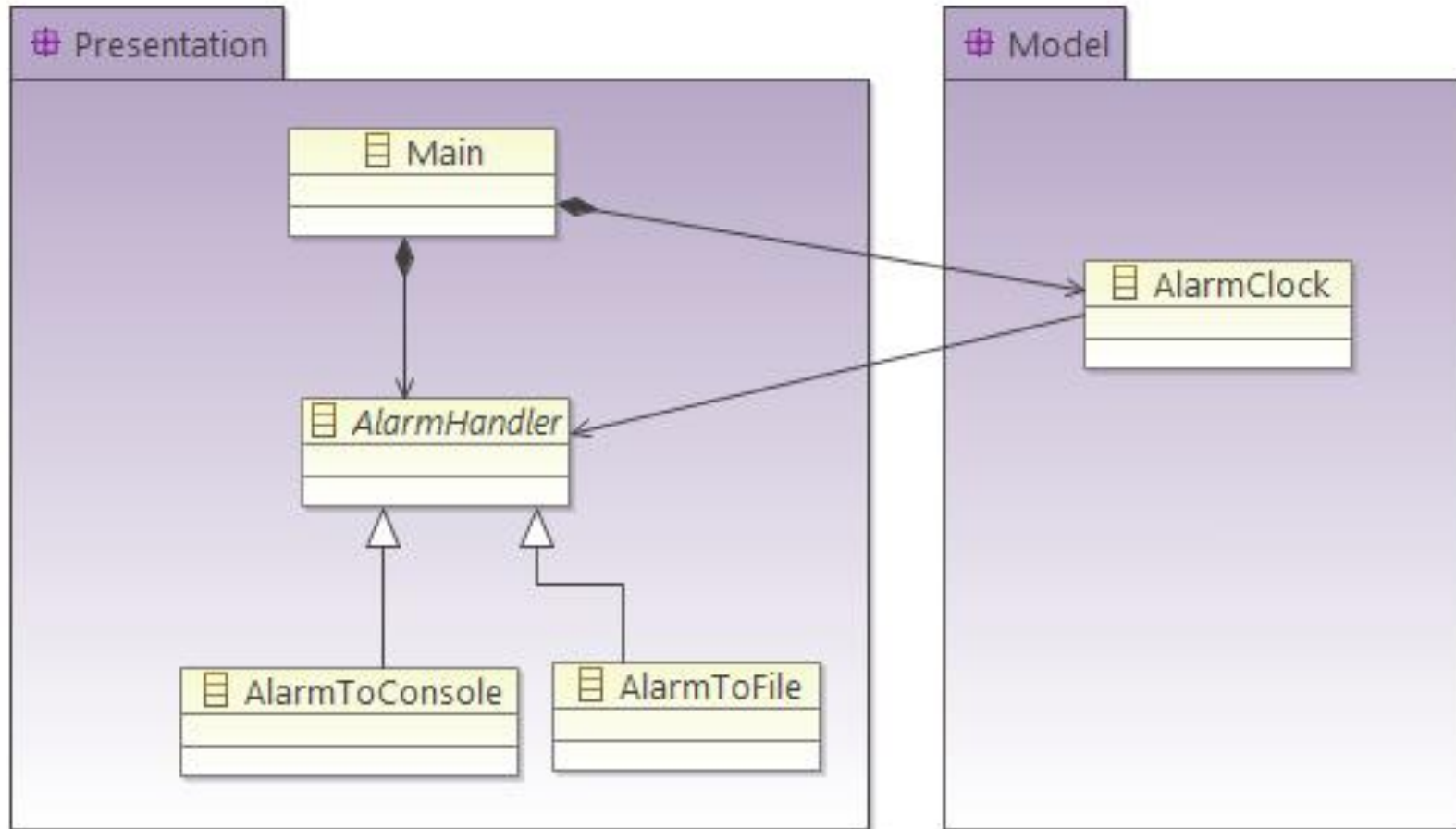
▶ Spring 4.0.0:	211
▶ Active MQ 5.5.1:	8 564
▶ Jenkins 1.512:	15 675
▶ JDK 1.7:	604 144



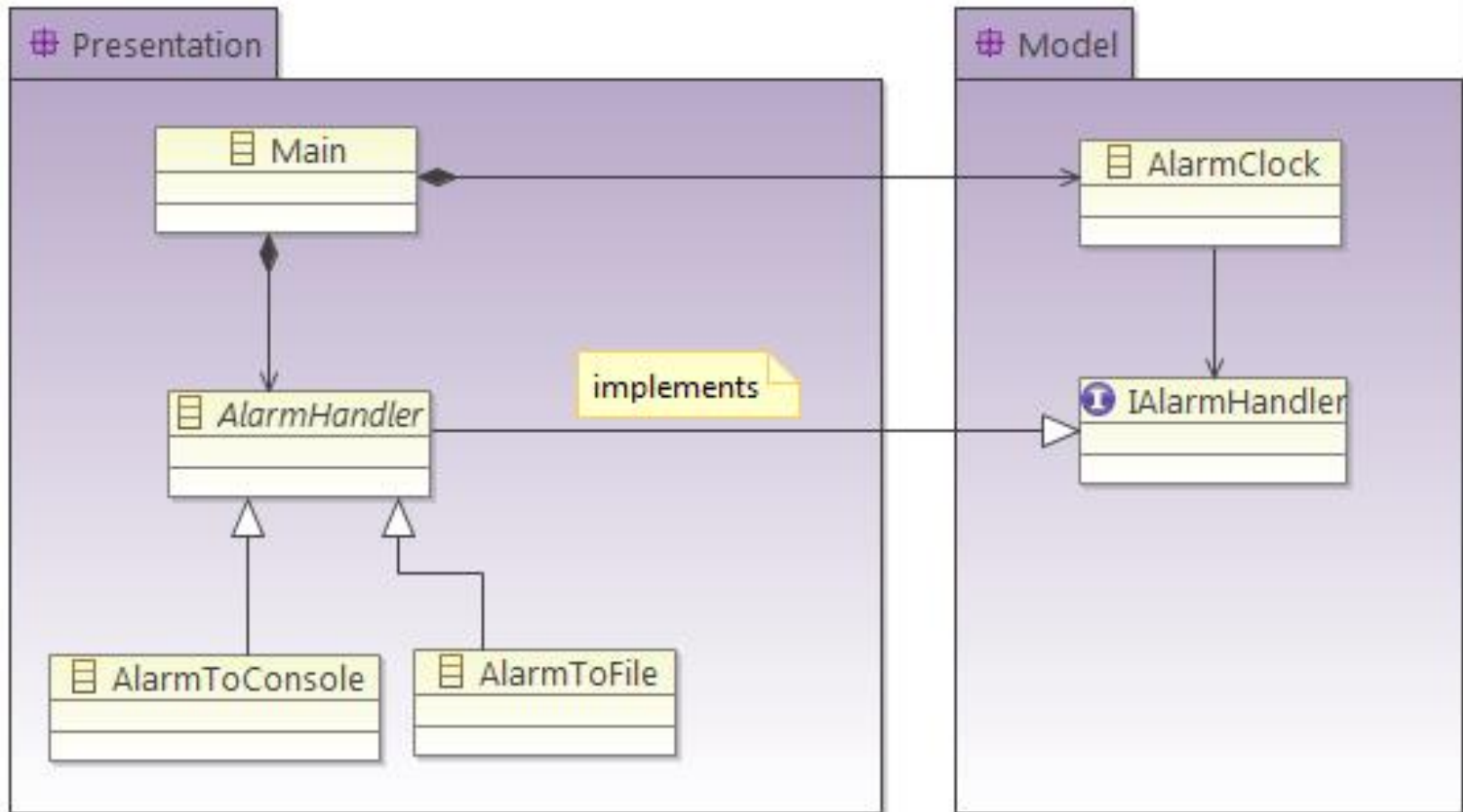
How to get out?



Refactoring example I



Dependency inversion





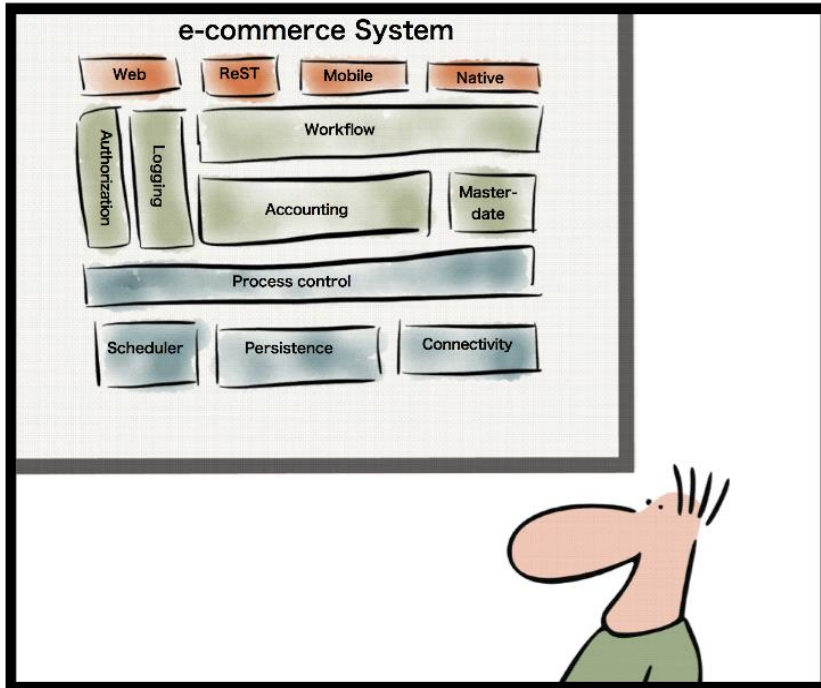
Control complexity at the micro level (class)

Useful metrics to avoid large complex classes and methods:

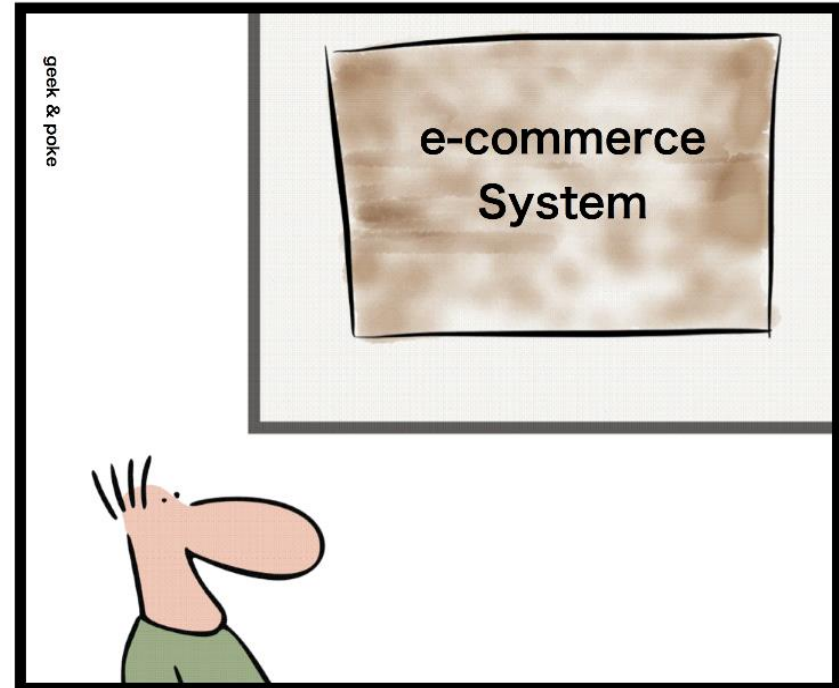
- ▶ Class level: LOC, number of methods, LCOM4
- ▶ Method level: LOC, number of parameters, cyclomatic complexity

Findbugs, PMD, Checkstyle help to find defects at this level.

“How to draw the architecture of your system”



RULE 1: JUST MAKE IT NICE



RULE 2: AND NOT REALISTIC!!!

<http://geekandpoke.typepad.com/.a/6a00d8341d3df553ef016764fffd81970b-pi>

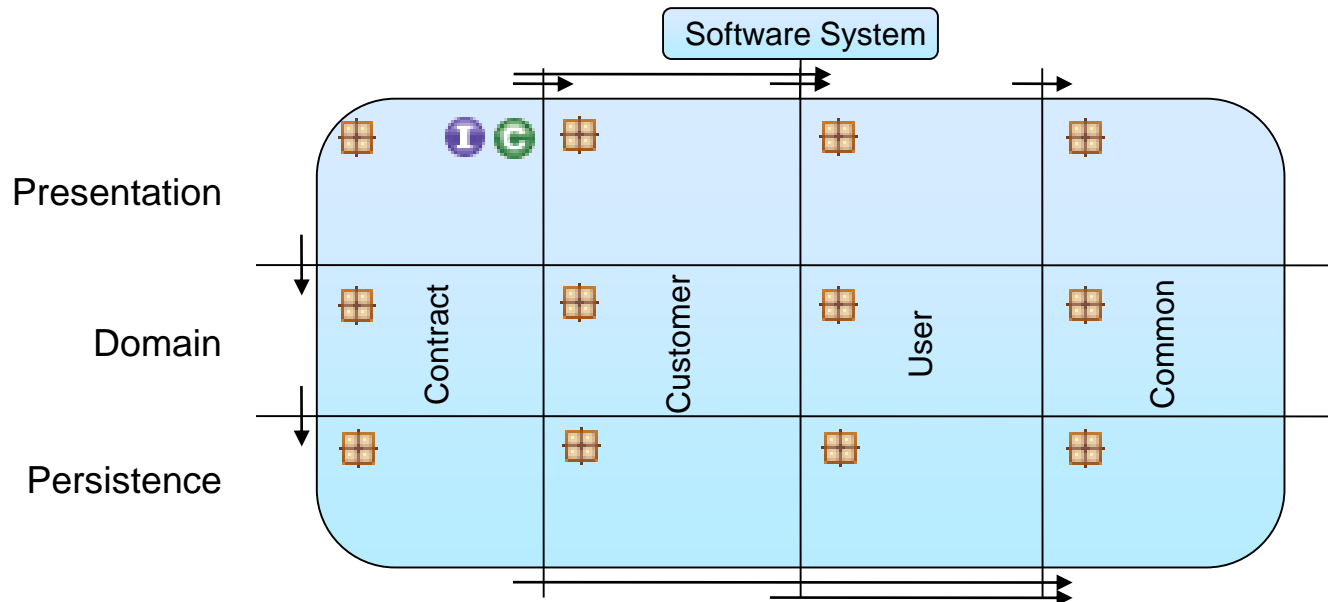
We need abstractions to understand and solve complex problems!



Control complexity at the macro level (architecture)



Define an Architecture Blueprint



- Step 1: Divide horizontally into layers by technical aspects
- Step 2: Divide vertically into slices by domain driven aspects
- Step 3: Define dependencies
- Step 4: Connect source code to the architecture



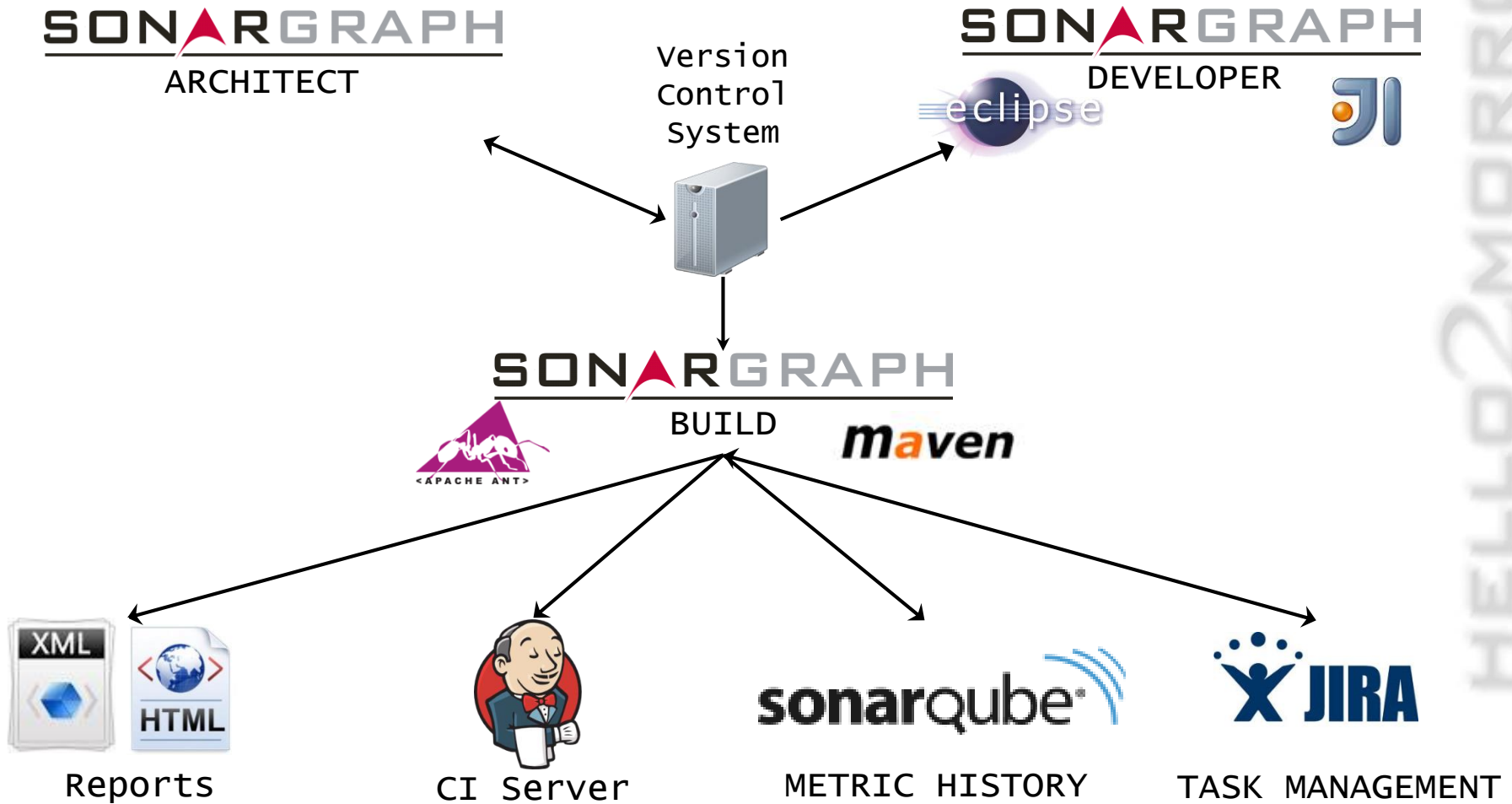
Best practices

- ▶ On existing projects, start with a small metric set
- ▶ Be patient and get management on board: Improvements won't happen automatically but need hard work
- ▶ Track your progress
- ▶ Metrics are NOT the solution, but only a vehicle to pin down potential problems. Don't optimize for metric values only!
- ▶ Reflection beats static analysis -> control its usage
- ▶ Static analysis is not the right method to find memory leaks and other performance problems

Remember: „A fool with a tool is still a fool“



Integration into the workflow

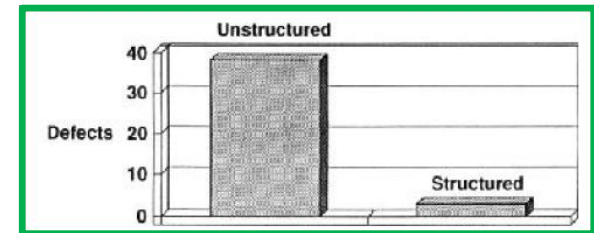
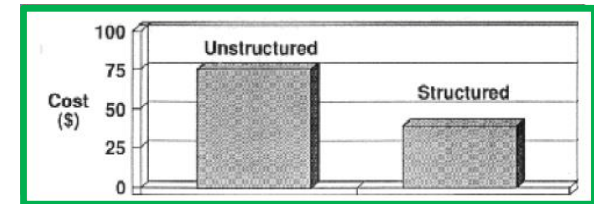
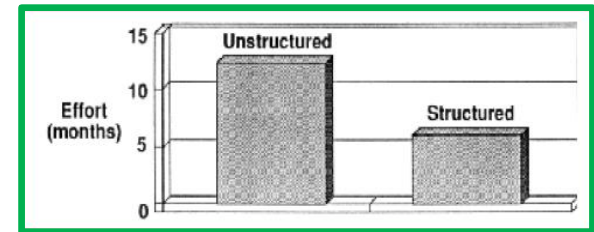
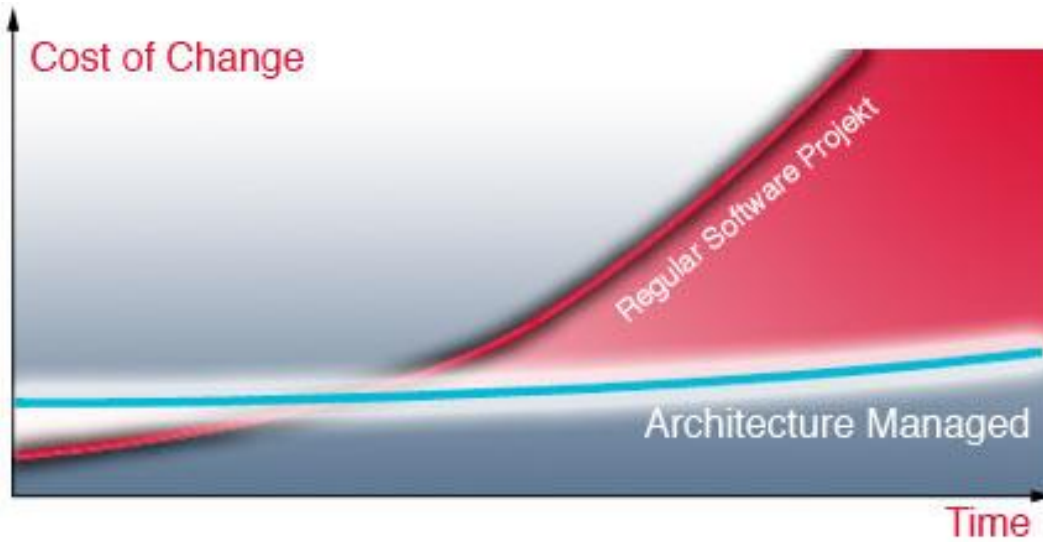


HELLO2MORROW



Quick demo

Why we should care



Barry M. Horowitz, DoD Study

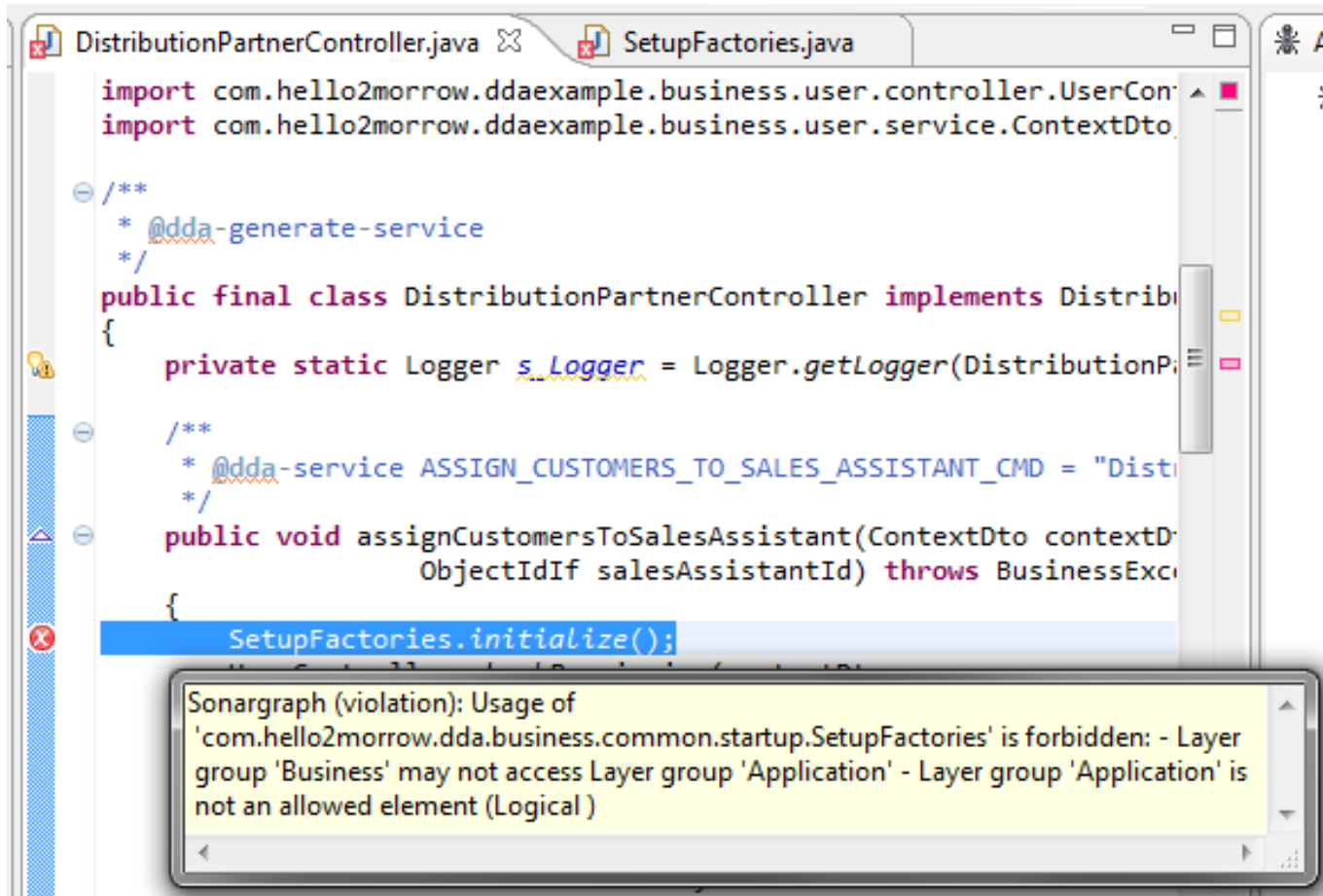
Further info

- ▶ i.kellner@hello2morrow.com
- ▶ Twitter: [@i_kellner](https://twitter.com/i_kellner)
- ▶ Whitepapers, DZone RefCard, etc. on our web page:
<http://www.hello2morrow.com>
- ▶ Blog: <http://blog.hello2morrow.com>

References

- ▶ Applying UML And Patterns, Craig Larman, Prentice Hall 2000
- ▶ Agile Software Development, Robert C. Martin, Prentice Hall 2003
- ▶ Large-Scale C++ Software Design, John Lakos, Addison-Wesley 1996
- ▶ Design Patterns, Gamma et al., Addison-Wesley 1994
- ▶ Controlling Software Projects: Management, Measurement, and Estimates, Tom DeMarco, Prentice Hall, 1982
- ▶ The Mythical Man Month, Frederick P. Brooks, Addison-Wesley, 1975, 1995
- ▶ The Pragmatic Programmer: From Journeyman to Master, Andrew Hunt, David Thomas, Addison-Wesley, 1999
- ▶ Copy & Paste & Bug, Dr. Elmar Jürgens, <http://entwicklertag.de/2012/vortraege/copy-paste-bug>
- ▶ <http://www.agilearchitect.org>

Sonargraph Eclipse Integration into Source Editor



The screenshot shows the Eclipse IDE with two Java files open: `DistributionPartnerController.java` and `SetupFactories.java`. The `DistributionPartnerController.java` file contains the following code:

```
import com.hello2morrow.ddaexample.business.user.controller.UserCon
import com.hello2morrow.ddaexample.business.user.service.ContextDto

/**
 * @dda-generate-service
 */
public final class DistributionPartnerController implements Distribi
{
    private static Logger s_Logger = Logger.getLogger(DistributionP

    /**
     * @dda-service ASSIGN_CUSTOMERS_TO_SALES_ASSISTANT_CMD = "Disti
     */
    public void assignCustomersToSalesAssistant(ContextDto contextD
        ObjectIdIf salesAssistantId) throws BusinessExce
    {
        SetupFactories.initialize();
    }
}
```

A Sonargraph violation tooltip is displayed over the `SetupFactories.initialize();` line. The tooltip text is:

Sonargraph (violation): Usage of 'com.hello2morrow.dda.business.common.startup.SetupFactories' is forbidden: - Layer group 'Business' may not access Layer group 'Application' - Layer group 'Application' is not an allowed element (Logical)

Sonargraph Jenkins CI Build Server Plugin

Jenkins Suchen ingmar

Jenkins > SGNG-CI-Build > Sonargraph AUTO-AKTUALISIERUNG EINSCHALTEN

- [Zurück zur Übersicht](#)
- [Status](#)
- [Änderungen](#)
- [Arbeitsbereich](#)
- [Jetzt bauen](#)
- [Projekt Löschen](#)
- [Konfigurieren](#)
- [Set Next Build Number](#)
- [Sonargraph](#)**
- [Git Abfrage-Protokoll](#)

Build-Verlauf (Trend)

#4260	04.11.2014 17:21:21	▲
#4259	02.11.2014 20:46:26	▲
#4258	02.11.2014 18:51:22	▲
#4257	02.11.2014 01:56:33	▲
#4256	31.10.2014 15:36:43	▲
#4255	31.10.2014 15:06:18	▲
#4253	31.10.2014 13:45:59	▲
#4252	31.10.2014 03:01:10	▲
#4251	31.10.2014 02:00:59	▲
#4250	30.10.2014 23:01:01	▲
#4249	30.10.2014 22:02:18	▲

[RSS aller Builds](#) [RSS der Fehlschläge](#)

Sonargraph

[Show most recent Sonargraph Report](#)

Violating Type Dependencies

Short Term

Structural Debt Index (SDI)

Short Term

Highest ACD

Short Term

Byte Code Instructions

Short Term

HELLO2MORROW

Sonargraph SonarQube Plugin (Web Interface)

Home Sonargraph - CRM Domain Example Configuration Administrator » Log out Search

Version 7.1.5 - 12. Apr 2012 14:14 Time changes... Configure widgets Edit layout Manage dashboard

Dashboard

- Hotspots
- Reviews
- Components
- Violations Drilldown
- Time Machine
- Clouds
- Design
- Libraries

CONFIGURATION

- Manual Measures
- Action Plans
- Settings
- Exclusions
- Links
- Project Roles
- History
- Project Deletion

sonar

Structural Debt Index **94**
1 open tasks
(with 1 references in code) ▲

Cost of Structural Debt ▲
1.034 USD

Architecture **8** violating type dependencies
3 violating types (1,6%)
14 violating references
0 ignored violations
0 unassigned types (0,0%)

Warnings ▲
10 total warnings
2 cycle groups
8 duplicate code blocks
0 threshold violations
0 workspace warnings
0 ignored warnings

Relative Package Cyclicity **13,0%**
5 biggest cycle group size
5 cyclic packages (12,5%)
3 type dependencies to cut (approx.)
4 references to remove (approx.)

ACD (John Lakos) ▲
16,1
2,4 NCCD (John Lakos)
24.533 byte code instr.

Violations **887**
Rules compliance **83,1%**

▲ Blocker	0
▲ Critical	3
▲ Major	248
▼ Minor	525
▼ Info	111

Package tangle index **0,6%**
> 3 cycles

Dependencies to cut
1 between packages
2 between files

Lines of code **7.608**
10.141 lines
2.610 statements
185 files

Classes **190**
40 packages
695 methods
116 accessors

Sonargraph Explorer: Extensible Analysis

Groovy Script

```
def NodeAccess node = result.addNode("Synchronized")
IJavaVisitor v = javaAccess.createVisitor()
v.onMethod {
    JavaMethodAccess method ->
    if (method.isSynchronized()) {
        result.addElement(method)
        result.addNode(node, method)
    }
    v.visitChildren(method)
}
javaAccess.visitParserModel(v)
```

Create Metrics, Issues, etc.

The screenshot shows the Sonargraph Explorer interface with the 'Metrics Preview' tab selected. The interface includes tabs for 'Elements (!)', 'Dependencies', 'Tree (!)', 'Issues Preview', and 'Metrics Preview'. Below the tabs, there is a header 'Name [60 elements]' and a list of 15 elements, each preceded by a blue triangle icon. The elements are:

- org.apache.cassandra.concurrent.JMXEnabledThreadPoolExecutor.shutdown()
- org.apache.cassandra.concurrent.JMXEnabledThreadPoolExecutor.shutdownNow()
- org.apache.cassandra.config.Schema.clear()
- org.apache.cassandra.db.ColumnFamilyStore.createColumnFamilyStore(Table,String,IPa)
- org.apache.cassandra.db.ColumnFamilyStore.loadNewSSTables()
- org.apache.cassandra.db.ColumnFamilyStore.loadNewSSTables(String,String)
- org.apache.cassandra.db.compaction.CompactionTask.addToTotalBytesCompacted(lor
- org.apache.cassandra.db.compaction.LeveledManifest.add(SSTableReader)
- org.apache.cassandra.db.compaction.LeveledManifest.getAllLevelSize()
- org.apache.cassandra.db.compaction.LeveledManifest.getCompactionCandidates()
- org.apache.cassandra.db.compaction.LeveledManifest.getEstimatedTasks()
- org.apache.cassandra.db.compaction.LeveledManifest.getLevelSorted(int, Comparator<
- org.apache.cassandra.db.compaction.LeveledManifest.promote(Iterable< SSTableReader
- org.apache.cassandra.db.compaction.LeveledManifest.repairOverlappingSSTables(int)
- org.apache.cassandra.db.compaction.LeveledManifest.replace(Iterable< SSTableReader>
- org.apache.cassandra.db.compaction.LeveledManifest.sendBackToL0(SSTableReader)
- org.apache.cassandra.db.compaction.LeveledManifest.serialize()

Some of our more than 200 customers

SIEMENS

UOB
大華銀行

KÜHNE+NAGEL

COMMERZBANK

ÖNB
OESTERREICHISCHE NATIONALBANK

Ford



Standard Chartered

sanofi aventis

DAIMLER-BENZ

Deutsche Bank



CREDIT SUISSE

Capgemini

T-Systems