A Unified Approach to Architecture Conformance Checking

Andrea Caracciolo, Mircea Filip Lungu, Oscar Nierstrasz

http://scg.unibe.ch
Architecture Erosion

Design ≠ Code
Architecture Erosion

Compliance Monitoring

RULES
1. you can....
2. you can’t...
3. you can....
4. you can’t
Compliance Monitoring

60% non-automated techniques *

* How Do Software Architects Specify and Validate Quality Requirements? (ECSA 2014)
Compliance Monitoring
Compliance Monitoring

Tools
Compliance Monitoring

Specification Formalisms

A
Б
诶
Compliance Monitoring
Compliance Monitoring
Persistence cannot depend on Service Impl must have annotation "@Service"
System cannot contain cycles
Rules

Persistence cannot depend on Service
Service must have annotation "@Service"
System cannot contain cycles
Rules

Persistence cannot depend on Service
Service must have annotation "@Service"
System cannot contain cycles

Entities

Persistence = Package with name:"app.*persistence.*"
Service = Package with name:"app.*service.*"
Rules

Persistence cannot depend on Service

only Service can have annotation "@Service"

System cannot contain cycles

Entities

Persistence = Package with name: "app.*persistence.*"

Service = Package with name: "app.*service.*"
Rules

Persistence cannot depend on Service
only Service can have annotation “@Service”
System cannot contain cycles

Entities

Persistence = Package with name: "app.*persistence.*"
Service = Package with name: "app.*service.*",
name!: "*persistence*"
Rules

Maintainability

Method can only be named "String"
only Package can contain dead methods

Compatibility

XMLTag must have attribute "String"
WebResource must have content "String"

Reliability

Method must catch Class

Performance

WebResource must have latency < int ms
WebResource must handle load from int users
Persistence cannot depend on Service

depend-on(app.xx.persistence.yy, app.xx.service)
depend-on(app.xx.persistence.yy2, app.xx.service)
depend-on(app.xx.persistence.yy3, app.xx.service)
Persistence cannot depend on Service

\[
\begin{align*}
\text{depend-on(app.xx.persistence.yy, app.xx.service)} &= True \\
\text{depend-on(app.xx.persistence.yy2, app.xx.service)} &= False \\
\text{depend-on(app.xx.persistence.yy3, app.xx.service)} &= False
\end{align*}
\]
Dictō

Persistence cannot depend on Service
Service must have annotation “@Service”
System cannot contain cycles
Dicto
Automated Architectural Tests. With emphasis on the Diff. What's this?

- Repository: https://github.com/studer-raimann/ILIAS
- Commit: ec5af7e96c6b3efbe4c0e5461d7d0815b833a
- Compared to Commit: 1d8c0b57f7c02a0df2a3a82a541dec10e46

742 Total Violations
468 Added Violations
0 Resolved Violations

- only GUIClasses can depend on iILanguage
  + 275

- WholelliasCodebase cannot depend on SuppressErrors
  + 182

- iiExceptionsWithoutTopLevelException can only depend on iiExceptions
  + 7

- WholelliasCodebase cannot depend on eval
  + 3

- WholelliasCodebase cannot depend on exitOrDie
  + 1

Exit and die are a bad idea in both development and production. In development you have no idea what went wrong and in production the user receives a white page and has no idea what's going on. The implemented exception handling does not work if you use exit or die.

If you want to send a file consider using: Services/FileDelivery.

Exception: Currently if you want to output json you most likely have to use exit[] at the moment.

| Added Violations | 1 |
|------------------|---|
| Resolved Violations | 0 |
| All Violations | 228 |

Newly Introduced Violations
- iiFileDelivery depends on exit/die
Value: 468
Build: #68 Tests failed: 76, passed: 182
Started: 22 Apr 15 16:28
Branch: continuous_integration
Series: Dicto Added Violations
Dictō

Persistence cannot depend on Service
Service must have annotation “@Service”
System cannot contain cycles
Evaluation

Open source project
LMS - PHP (1.8M LOC)
12 service providers, 900’000+ users

Large size company
B2B - Java EE (50K LOC)
1’000 employees

Medium size company
various - Java EE / .NET
100 employees
Evaluation

**Impact**
- Process?
- Culture?

**Value**
- Cost–effectiveness?
- Quality?

**Applicability**
- Expressivity?
- Usability?
Summary

Dictō

A uniform, readable, executable DSL for specifying architectural constraints.

Andrea Caracciolo

@ scg.unibe.ch/dicto
Discussion Topics

Persistence cannot depend on Service
Service must have annotation "@Service"
System cannot contain cycles

- how to streamline/incentivize compliance monitoring?
- which are the common obstacles?

Andrea Caracciolo
@ scg.unibe.ch/dicto