

Architecture Rules Enforcement and Governance Using Aspects

Srini Penchikala
SATURN 2009

About the Speaker

- ❑ Enterprise Architect
- ❑ Writer, Speaker, Editor (InfoQ)
- ❑ Detroit Java User Group Leader
- ❑ Working with Java since 1996, JEE (2000), SOA (2006), & PowerPoint since Sep 2008
- ❑ Current: Agile Architectures, Domain-Driven Design, Architecture Enforcement, Model Driven Development
- ❑ Future: Role of DSL's in Architecture Enforcement



Goals for this Presentation

- ❑ Overview of Reference Architecture (RA) and its significance in EA
- ❑ How Aspects and AOP can help to enforce RA and manage Architecture Governance

3

Format

- ❑ Interactive
- ❑ Demos
- ❑ Duration: ~60 minutes
- ❑ Q & A
- ❑ Prerequisite: Familiarity with AOP and Aspects

4

Before we start...

- ❑ How many currently have some kind of Reference Architecture in place?
- ❑ How many actually use RA to enforce architecture?

5

Agenda

- ❑ Reference Architecture & Enforcement
- ❑ Architecture Rules Categories
- ❑ Architecture Enforcement Approaches
- ❑ Aspect-oriented Programming
- ❑ Rules Demo(s)
- ❑ CI Process Changes
- ❑ Case Study
- ❑ Architecture Rules Aspects - Open Source Project

6

Agenda

- ▣ Reference Architecture & Enforcement
- ▣ Architecture Rules Categories
- ▣ Architecture Enforcement Approaches
- ▣ Aspect-oriented Programming
- ▣ Rules Demo(s)
- ▣ CI Process Changes
- ▣ Case Study
- ▣ Architecture Rules Aspects - Open Source Project

7

Reference Architecture

- ▣ A high-level system design which consists of:
 - description of system components
 - definitions of relationships between components
 - definitions of relationships between system components & elements external to the system

8

Architecture Enforcement

□ Architecture Rules:

- Rules (compile time as well as run time) used to assert architecture.

□ Architecture Governance:

- Construct an Architecture Contract to govern overall implementation & deployment process.
- Perform appropriate governance functions while system is being implemented and deployed.
- Ensure conformance with defined architecture by implementation projects

9

Architecture Enforcement Maturity Model

- Level 0: No RA
- Level 1: RA is described in a Word/HTML/PDF document
- Level 2: Semi-automated checking mechanism for architecture enforcement
- Level 3: Fully automated checking mechanism

10

Why Architecture Enforcement?

- ❑ Checking mechanism to enforce Reference Architecture
- ❑ Match Requirements (Architecture) to Implementation (Code)
- ❑ Detect and prevent structural complexity
- ❑ Promotes consistency and modularity in the System
- ❑ Aids in Architecture, Design, and Code refactoring efforts

11

Architecture Rules

- ❑ Business v. Architecture Rules
 - Architecture rules are as important as business rules
- ❑ Code Quality By Design
 - Unit tests, TDD & BDD only help with code quality from functional requirements stand-point
 - Need for a process to ensure code quality from design stand-point
- ❑ Technical Credit
 - Good Design, Testable and Integrated Code
 - Code that complies with Architecture/Design Policies

12

Types of Enforcement

- ❑ Layered architecture and application module dependencies
- ❑ API usage (internal and 3rd party)
- ❑ Coding Policies
- ❑ Code-Naming Conventions
- ❑ Design Patterns and Best Practices

13

Setup Types

- ❑ Static analysis
 - Allows for easy visual representation
 - Techniques
 - AOP, AspectJ
 - Tools
 - Structure 101
 - SonarJ
 - Lattix
- ❑ Dynamic analysis
 - Based on the program flow
 - Rule-set usable at runtime
 - Techniques: AspectJ/Spring AOP

14

Agenda

- ▣ Reference Architecture & Enforcement
- ▣ **Architecture Rules Categories**
- ▣ Architecture Enforcement Approaches
- ▣ Aspect-oriented Programming
- ▣ Rules Demo(s)
- ▣ CI Process Changes
- ▣ Case Study
- ▣ Architecture Rules Aspects - Open Source Project

15

Rule Categories

- ▣ Layered Architecture
- ▣ Separation Of Concerns
- ▣ Domain-Driven Design
- ▣ Infrastructure
- ▣ Miscellaneous

16

Rules - Layered Architecture

- ❑ Presentation layer should not use DAO classes directly
- ❑ Service layer never calls web layer
- ❑ DAO (Persistence) layer depends on no other layer except domain

17

Rules - Separation of Concerns

- ❑ No transaction management in DAO classes
- ❑ Only DAO's should reference javax.sql classes
- ❑ Only Controller classes are aware of Web/HTTP objects
- ❑ Service layer should be transactional
- ❑ External systems can not access internal implementation classes (*Impl classes)

18

Rules - Domain-Driven Design

- ❑ Service object creation via a Factory (i.e. no “new Service()” code)
- ❑ No direct access to DAO's except from Domain classes
- ❑ Business service that fails with a concurrency related failure can be retried

19

Rules - Infrastructure

- ❑ No direct use of API:
 - Log4J or Commons Logging (use Framework Logger)
 - JavaMail (use MailService component)
 - FTP (use FileTransferService component)
- ❑ No System.out.println statement anywhere in the code

20

Rules - Miscellaneous

- ❑ All primary key classes (*PK.java) should implement Serializable interface
- ❑ All test classes should be named with the suffix "Test"

21

Persistence

- ❑ Hibernate session close only allowed in EJB's and Servlet Filters
- ❑ DAO's accessing DB Views use JDBC (no Hibernate)
- ❑ Use SpringJdbcTemplate for JDBC data access

22

Other Domain Concerns

▣ Auditing

- Domain state change tracking

▣ Monitoring

- Aspects to implement MonitorMBean interface
- Expose Results as an MBean
- Publish results using RSS feeds

23

Agenda

- ▣ Reference Architecture & Enforcement
- ▣ Architecture Rules Categories
- ▣ Architecture Enforcement Approaches
- ▣ Aspect-oriented Programming
- ▣ Rules Demo(s)
- ▣ CI Process Changes
- ▣ Case Study
- ▣ Architecture Rules Aspects - Open Source Project

24

Approaches

- ❑ Code Inspections
- ❑ Architecture Reconstruction
- ❑ Model Driven Architecture (MDA)
- ❑ Code Generation
- ❑ Byte Code Instrumentation
- ❑ Enforcement Tools
- ❑ Aspect-oriented Programming (AOP)

25

Architecture Analysis Tools

- ❑ Macker
- ❑ ArchitectureRules
- ❑ Classycle
- ❑ PatternTesting
- ❑ Contract4J
- ❑ Structure 101, Lattix, SonarJ

26

Architecture Analysis Tools

- ❑ Structure 101
 - Slicing / Rules
- ❑ SonarJ
 - Logical architecture definition & Physical mapping of architecture to Java code
- ❑ Lattix
 - Dependency Structure Matrix showing the Desired vs. the Realized architecture implementation

27

Agenda

- ❑ Reference Architecture & Enforcement
- ❑ Architecture Rules Categories
- ❑ Architecture Enforcement Approaches
- ❑ Aspect-oriented Programming
- ❑ Rules Demo(s)
- ❑ CI Process Changes
- ❑ Case Study
- ❑ Architecture Rules Aspects - Open Source Project

28

Aspect-oriented Programming

- ❑ Add behavior to objects in a non-obtrusive manner through static and dynamic crosscutting
- ❑ Code cross-cutting concerns in separate modules and apply them in a declarative way

29

AOP Use Cases

- ❑ Spring Framework Built-In Aspects:
 - Transaction Management
 - Security
- ❑ Custom Aspects:
 - Profiling
 - Caching
 - Architecture Rules
 - Contract Enforcement

30

AOP in Architecture Enforcement

- ❑ Provides the necessary abstraction to enforce rules
- ❑ Customized compile-time error or warning messages
- ❑ Less intrusive
- ❑ Target application code is not modified in any form
- ❑ Rules can be turned on or off any time

31

Enforcement: Aspects or Tools?

- ❑ Question:
 - AOP or Tools?

32

Enforcement: Aspects or Tools?

□ Tools:

- +Better enforcement options
- -Licensing costs

□ Aspects

- +More flexibility for customization
- -Relatively intrusive
- -Not backed by an architecture meta model

33

Enforcement: Aspects or Tools?

□ Question:

- AOP or Tools?

□ Answer:

- It depends..
- Choose the right tool to do the right job

34

Agenda

- ▣ Reference Architecture & Enforcement
- ▣ Architecture Rules Categories
- ▣ Architecture Enforcement Approaches
- ▣ Aspect-oriented Programming
- ▣ Rules Demo(s)
- ▣ CI Process Changes
- ▣ Case Study
- ▣ Architecture Rules Aspects - Open Source Project

35

Architecture Rules Enforcement

- ▣ IDE (Eclipse, AJDT)
- ▣ Build Process Changes (Ant, CruiseControl)
- ▣ RSS Feeds (Rome)

36

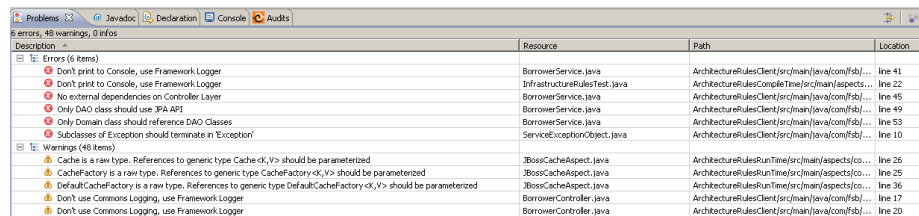
Sample Application

Tools:

- Eclipse
- AspectJ
- AJDT
- Spring AOP
- Ant

37

Sample Application - Eclipse



Description	Resource	Path	Location
Errors (6 items)			
Don't print to Console, use Framework Logger	BorrowerService.java	ArchitectureRulesClient/src/main/java/conf/sbj/...	line 41
Don't print to Console, use Framework Logger	InfrastructureRulesTest.java	ArchitectureRulesClient/src/main/aspectj/...	line 22
No external dependencies on Controller Layer	BorrowerService.java	ArchitectureRulesClient/src/main/java/conf/sbj/...	line 45
Only DAO class should use JPA API	BorrowerService.java	ArchitectureRulesClient/src/main/java/conf/sbj/...	line 49
Only Domain class should reference DAO Classes	BorrowerService.java	ArchitectureRulesClient/src/main/java/conf/sbj/...	line 53
Subclasses of Exception should terminate in 'Exception'	ServiceExceptionObject.java	ArchitectureRulesClient/src/main/java/conf/sbj/...	line 10
Warnings (40 items)			
Cache is a raw type. References to generic type Cache<K, V> should be parameterized	JBossCacheAspect.java	ArchitectureRulesRunTime/src/main/aspectj/co...	line 26
CacheFactory is a raw type. References to generic type CacheFactory<K, V> should be parameterized	JBossCacheAspect.java	ArchitectureRulesRunTime/src/main/aspectj/co...	line 25
DefaultCacheFactory is a raw type. References to generic type DefaultCacheFactory<K, V> should be parameterized	JBossCacheAspect.java	ArchitectureRulesRunTime/src/main/aspectj/co...	line 36
Don't use Commons Logging, use Framework Logger	BorrowerController.java	ArchitectureRulesClient/src/main/java/conf/sbj/...	line 17
Don't use Commons Logging, use Framework Logger	BorrowerController.java	ArchitectureRulesClient/src/main/java/conf/sbj/...	line 20

38

Layered Architecture Rules

- ▣ Service classes should not depend on Application layer classes

39

-
- ▣ DEMO

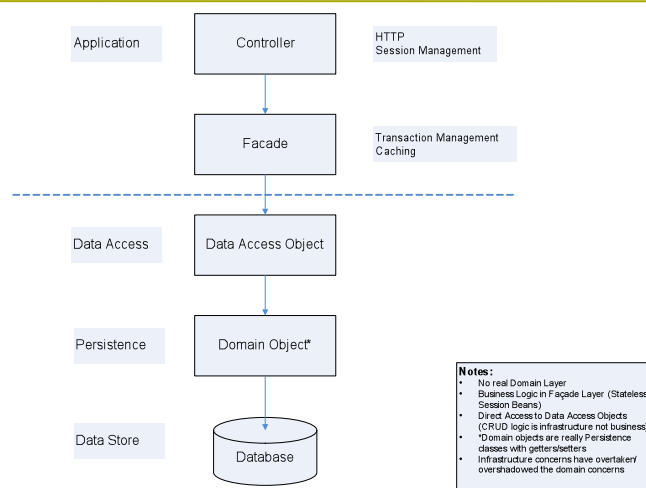
40

Domain-Driven Design Rules

- ❑ No direct access to DAO's except from Domain classes (go through Domain or Repository objects)

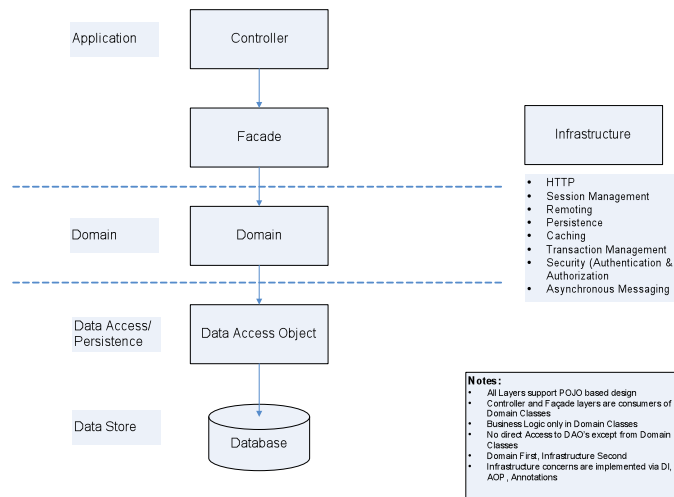
41

Typical J2EE Architecture Model



42

Domain Driven Architecture



43

□ DEMO

44

DDD Rules Demo 2

- ▣ Business service that fails with a concurrency related failure can be retried
- ▣ Caching example
 - Spring AOP and EHCACHE
 - Requirement: To cache specific data (objects) using a custom Annotation
 - Annotation: @CacheEntity

45

▣ DEMO

46

Infrastructure Rules

- ▣ No direct use of API:
 - Log4J or Commons Logging (use Framework Logger)

47

-
- ▣ DEMO

48

Agenda

- ▣ Reference Architecture & Enforcement
- ▣ Architecture Rules Categories
- ▣ Architecture Enforcement Approaches
- ▣ Aspect-oriented Programming
- ▣ Rules Demo(s)
- ▣ CI Process Changes
- ▣ Case Study
- ▣ Architecture Rules Aspects - Open Source Project

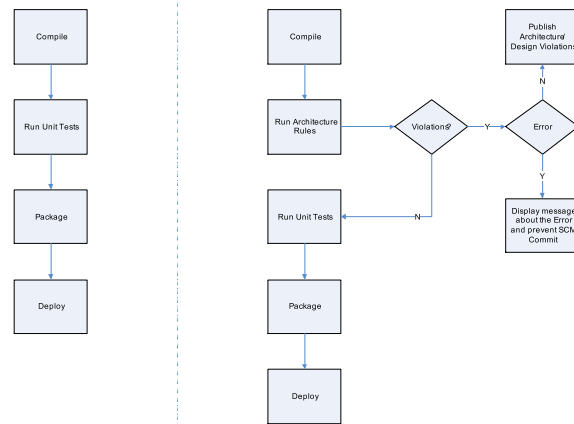
49

CI Process Changes

- ▣ Build process changes to include Architecture rules task
- ▣ Part of build process (in Local and Integration environments)
- ▣ Deviations are published nightly as RSS feeds
- ▣ Identified deviations are used in conducting design and code reviews

50

Build Process Changes



51

Sample Application - Ant Build

```

C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient>
C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient>ant -buildfile build/arch-rules-build.xml run.arch.rules
Buildfile: build/arch-rules-build.xml

run.arch.rules:
[jav] warning at private static log log = LoggerFactory.getLogger(BorrowerService.class);
[jav] [C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient\src\main\java\com\fb\pa\controller\BorrowerController.java:17:
0:0] Don't use Commons Logging, use Framework Logger
[jav] see also: C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesCompileTime\bin\com\fb\pa\Framework\aspects\Archrules\I
nfrastructure\InfrastructureRulesAspect.class:30:0
[jav] warning at log.debug("execute() called.");
[jav] [C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient\src\main\java\com\fb\pa\controller\BorrowerController.java:20:
0:0] Don't use Commons Logging, use Framework Logger
[jav] see also: C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesCompileTime\bin\com\fb\pa\Framework\aspects\Archrules\I
nfrastructure\InfrastructureRulesAspect.class:30:0
[jav] warning at private static log log = LoggerFactory.getLogger(FrameworkUtil.class);
[jav] [C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient\src\main\java\com\fb\pa\Framework\util\FrameworkUtil.java:19:0
0:0] Don't use Commons Logging, use Framework Logger
[jav] see also: C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesCompileTime\bin\com\fb\pa\Framework\aspects\Archrules\I
nfrastructure\InfrastructureRulesAspect.class:30:0
[jav] warning at private static log log = LoggerFactory.getLogger(BorrowerService.class);
[jav] [C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient\src\main\java\com\fb\pa\service\BorrowerService.java:23:0:0] D
on't use Commons Logging, use Framework Logger
[jav] see also: C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesCompileTime\bin\com\fb\pa\Framework\aspects\Archrules\I
nfrastructure\InfrastructureRulesAspect.class:30:0
[jav] warning at log.debug("updateBorrowerDetails() called.");
[jav] [C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient\src\main\java\com\fb\pa\service\BorrowerService.java:35:0:0] D
on't use Commons Logging, use Framework Logger
[jav] see also: C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesCompileTime\bin\com\fb\pa\Framework\aspects\Archrules\I
nfrastructure\InfrastructureRulesAspect.class:30:0
[jav] warning at log.debug("updateBorrowerDetails() called.");
[jav] [C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient\src\main\java\com\fb\pa\service\BorrowerService.java:41:0:0] D
on't print to Console, use Framework Logger
[jav] see also: C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesCompileTime\bin\com\fb\pa\Framework\aspects\Archrules\I
nfrastructure\InfrastructureRulesAspect.class:30:0
[jav] error at BorrowerController.execute();
[jav] [C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesClient\src\main\java\com\fb\pa\service\BorrowerService.java:45:0:0] M
ethods are not allowed to be overridden
[jav] see also: C:\dev\projects\ResearchLabs\ReferenceArchitecture\ArchitectureRulesCompileTime\bin\com\fb\pa\Framework\aspects\Archrules\I
nfrastructure\InfrastructureRulesAspect.class:30:0
[jav] error at an.isOpen();

```

52

Agenda

- ▣ Reference Architecture & Enforcement
- ▣ Architecture Rules Categories
- ▣ Architecture Enforcement Approaches
- ▣ Aspect-oriented Programming
- ▣ Rules Demo(s)
- ▣ CI Process Changes
- ▣ Case Study
- ▣ Architecture Rules Aspects - Open Source Project

53

Case Study

- ▣ Implementation Plan:
 - Phase 1 (Day 1): Enable rules but not enforce them (yet)
 - Phase 2 (Day 16): Enable rules and enforce only critical Errors (No Warnings)
 - Phase 3: Enable rules and enforce all deviations (Errors & Warnings)
 - Phase 4: Enable rules and enforce all Errors and Warnings in all applications in the enterprise

54

Architecture Rules Aspects

- ❑ Open source Google project
 - architecture-rules-aspects (<http://code.google.com/p/architecture-rules-aspects/>)
- ❑ Future road-map:
 - DSL to define architecture rules (e.g. Groovy)
 - Define rules in a Rules Engine (JBossRules)

55

Implementation Road Map

- ❑ Publish rules along with Reference Architecture (RA) and Standard Technology/Framework Stack
- ❑ Pick a pilot application to enforce architecture rules as part of the build process (and Eclipse IDE)
- ❑ Incorporate architecture enforcement into other applications
- ❑ Training/Mentoring as part of an on-going Architecture Governance effort

56

Conclusions

- ❑ Make architecture rules part of build process to detect deviations earlier in SDLC
- ❑ Use enforcement check results to improve architecture/design/code (via refactoring efforts)
- ❑ Capture design patterns and best practices as policy enforcement Aspects
- ❑ Refine & refactor enforcement rules

57

AOP Tools

- ❑ AJDT
- ❑ Spring AOP
- ❑ MaintainJ: Sequence Diagram Using AspectJ
- ❑ PointcutDoctor
- ❑ Jexin

58

References (1/2)

- ❑ Architecture Rules Aspects Google Project (<http://code.google.com/p/architecture-rules-aspects/>)
- ❑ AspectJ In Action (2nd Edition)
- ❑ Building Software As Solid As The Pyramids (Ramnivas Laddad & Alef Arendsen)
- ❑ Using Aspect-Oriented Programming to Enforce Architecture (Paulo Merson)
- ❑ AOP In the Enterprise (Adrian Colyer)
- ❑ Aspects & policy injection - clean up your code
- ❑ Policy Injection Application Block

59

References (2/2)

- ❑ Structure 101
- ❑ SonarJ (<http://www.hello2morrow.com>)
- ❑ Lattix
- ❑ Contract4J (<http://www.contract4j.org/contract4j>)
- ❑ Macker (<http://innig.net/macker/>)
- ❑ Rome - RSS Feeds Framework (<https://rome.dev.java.net/>)

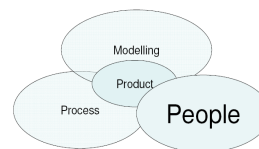
60

Contact Information

- ▣ Domain-Driven Design and Enterprise Architecture articles on InfoQ
- ▣ InfoQ website (<http://www.infoq.com>)
- ▣ E-Mail: srinipenchikala@gmail.com
- ▣ Blog: <http://srinip2007.blogspot.com>

61

▣ Q & A



Non-Linear First-Order Components
Alistair Cockburn

62

Thank You

- ▣ Thank you for your attention
- ▣ Feedback survey