Ultimate Architecture Enforcement

Custom checks enforced at code-commit time

CBSoft 2013 – Brasília
October 1st, 2013
Paulo Merson

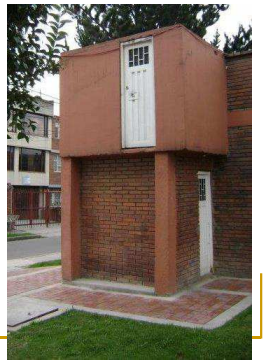
TRIBUNAL DE CONTAS DA UNIÃO

A fictitious, grim story

- A software architecture was carefully defined based on:
  - Architecture and design patterns
  - Experience with similar design problems
  - Reviews with stakeholders
- The architecture was incrementally implemented
  - Code became the main artifact
- Development & maintenance:
  - Involved tens of people
  - Spanned several years

The end of the story

ACTUAL ARCHITECTURE != INTENDED ARCHITECTURE

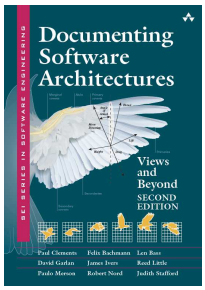


3

Consequences of lack of conformance

- Maintainability decreases due to introduction of undesirable dependencies
  - Code becomes brittle, hard to understand and change
- Possible negative effect:
  - On reliability, security, portability, performance, interoperability and other qualities
  - Caused by deviation from design decisions that addressed quality attribute requirements

How to avoid code and architecture disparity?



1. Communicating the architecture to developers
   - Not the focus of this presentation
2. Automated architecture conformance analysis
   - Often using static code analysis tools

FORTIFY An HP Company
pmd DON'T SHOOT THE MESSENGER
LATTIX
sonarqube
LINT
Checkstyle
FindBugs

TCU

5

Two limitations of static analysis tools as commonly used

1. Many people just enable built-in checks, which
- Have configurable thresholds but are still generic
- Are oblivious to custom design constraints and project-specific named layers and other elements

2. Developers can ignore reported violations
- There can be thousands of violations ☹
- There's no time to "clean the house" ☹
- There may be no accountability for adding violations ☹

TCU

Paulo Merson
October 2013

6

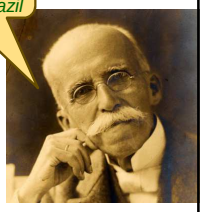
How did we overcome limitation #1

- Checkstyle custom checks
  - Use the checkstyle API
  - Have full access to a Java file AST
  - Follow the Visitor design pattern

Custom check example

```
/** Classes prefixed by Dao can't be used outside com.mycompany.mysystem.service.*     */
public class CheckNonServiceUsesDao extends Check {
  private boolean inServiceLayer;
  @Override
  public int[] getDefaultTokens() {
    return new int[] {TokenTypes.PACKAGE_DEF, TokenTypes.IDENT};
  }
  @Override
  public void visitToken(DetailAST aAST) {
    if (aAST.getType() == TokenTypes.PACKAGE_DEF) {
      inServiceLayer = false;
      String packageName = fullyQualifiedPackage(aAST);
      if (packageName != null &&
          packageName.startsWith("com.mycompany.mysystem.service")) {
        inServiceLayer = true;
      }
    } else if (aAST.getType() == TokenTypes.IDENT && !inServiceLayer) {
      if (aAST.getText().startsWith("Dao")) {
        log(aAST.getLineNo(),
          "Classes outside the service layer can't call Dao classes");
      }
    }
  }
}
```

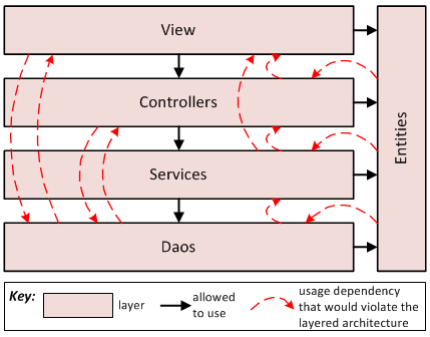
Custom checks at TCU



- We've created 43 custom checks
  - 22 checks for architecture conformance
  - 12 checks for coding guidelines
  - 9 checks for application security

Architecture conformance checks

- Enforce:
  - The layered architecture
  - Mandated generalizations
  - Placement of UI, business, data access logic
  - Naming of software elements
  - Use of infrastructure/util elements



Coding guideline checks

- Check proper coding of
  - Exception handling
  - Resource release
  - Thread programming
  - JUnit tests

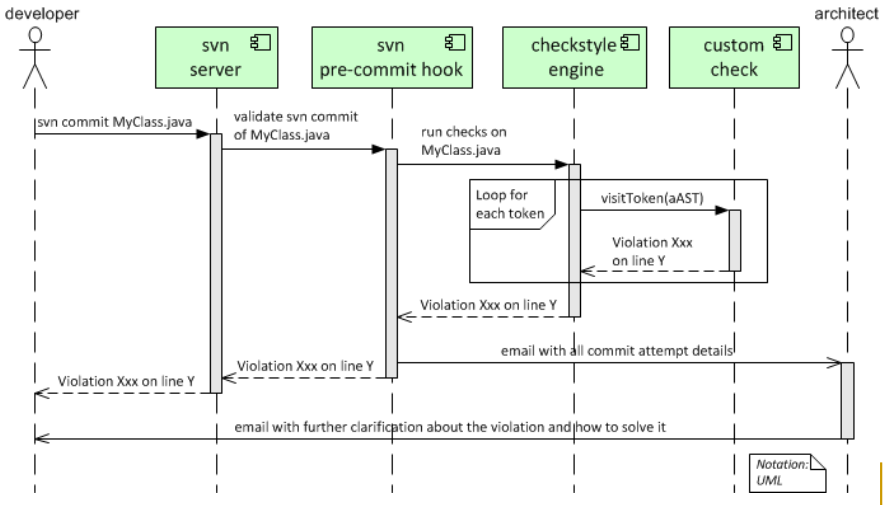
Application security checks

- Prevent vulnerabilities, such as:
  - SQL injection (JDBC and Hibernate)
  - Execution of external programs in web apps
  - Hard-coded passwords
  - Subclassing/overriding security critical classes or methods
  - Lack of authentication/authorization at entry points

How did we overcome limitation #2

- Upon code commit a subversion *pre-commit hook*:
  - Runs custom checks on each Java file
  - If any check detects a violation
    - The commit operation is denied
    - User sees an error message informing
      - File name,
      - Line number,
      - Short description of the violation and how to fix it
    - Architecture team is notified of the failed commit attempt

Failed code commit attempt



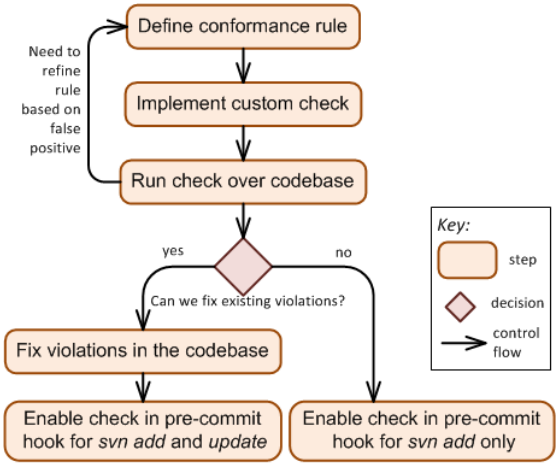
Lessons learned (1)

- Architecture team should not be the “architecture police” but the “architecture mentors”
- Adopt an *architecturally-evident coding style*
  - Naming conventions or annotations for each type of element
- Let developers know about the custom checks and open space for suggestions
- Let all architects understand the potential of custom checks

Lessons learned (2)

- Create a mechanism for exceptions to the rules
- Track the violation count per check over time
- Enable custom checks on the IDE and continuous integration
- Create unit tests for custom checks

Process to define, implement, and enable a custom check



Questions – Now or Later

Paulo Merson
pmerson@acm.org