Why Architecture Conformance Matters for Software Systems

John Klein  jklein@sei.cmu.edu
Robert Nord
Forrest Shull

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA  15213
Why Architecture Conformance Matters

Terminology
• Architecture
• Conformance

Problem Definition

Improving Your Practices

Automation Challenges

Research – Early Results
Architecture

A system’s architecture is defined by its significant design decisions, where in my experience, “significant” is measured by the cost of change.

- Grady Booch

A system’s ability to meet its desired (or required) quality attributes is substantially determined by its architecture. If you remember nothing else from this book, remember that.

- Len Bass, Paul Clements, and Rick Kazman

Implications for the System

Architecture can only permit, not guarantee, any quality attribute.

For the implementation to exhibit the quality attributes engineered at the architectural level, it must conform to the architecture.
Typical Nonconformance Scenario

- **Design**
  - Extensible, Composable
- **Implementation**
  - Baseline Delivery

**Months or Years**

- **Planned V.2 Release**
  - Depends on Extensibility and Composability
- **Actual V.2 Release**
  - Remediate...

- **Discover Nonconformance**
Challenges in Software Conformance

Modular Open Systems Approach (MOSA)
- technical and business strategy
- affordable and adaptable systems

Technical Standards, e.g., FACE
- conformance verification matrix
  - 487 items, 194 are inspection of design
- component-level standard

FACE-conformant systems may encounter integration problems.

Improve Conformance of Implementations to Architectures

An automated design conformance checker integrated into a continuous integration (CI) workflow

• exposes nonconformances at time of commit instead of months later
• promotes conversation whether code or architecture needs to change
• allows remediation before violations become fixed in the implementation
• enables program managers to hold developers accountable
State of the Art in Software Conformance

Tools based on static analysis

- E.g., Understand™, Structure101, Lattix

Code quality metrics are weakly correlated to nonconformance, but may be indicative

Dependency structure analysis can detect nonconformance for some architecture styles, e.g., Layers

Effective use requires tuning of out-of-the-box configuration

System-specific ad hoc tests

- E.g., fitness functions

Implement using unit test framework like CppUnit or JUnit

Tests could be developed to check conformance to any architecture style

As the architecture and/or the implementation evolves, the tests related to those changes may become brittle over time and need to be maintained

https://www.scitools.com
https://structure101.com
https://www.lattix.com

Goal – Automatically Inferring Design from Code

Need to bridge the model-code gap
Essential challenges to detecting design constructs
- imprecise definitions of abstractions
- variation in implementation
- limits of fact gathering analyses
- alignment or correlation of extracted design fragment to intended design fragment

Steps You Can Take Today to Improve Conformance

Capture (parts of) the intended design – structures and observable principles that allow the system to satisfy the high priority quality attribute requirements

Adopt (and refactor to) an architecturally-evident coding style to bridge the model-code gap in the parts of the intended design that you have captured

Check conformance to the key parts of your intended architecture at commit time or incremental release or major release or gate review…but do it!

- Manual checks – in peer reviews
- Automated checks – tailor rules in off-the-shelf tools and/or ad hoc tests


Towards Generalized Automation: Prototype Conformance Checker

Developed at SEI in 2022
What Is Involved in Applying the Approach

We have learned how to customize the approach for a particular framework-based system and architecture communication style.
What Practical Problems Does the Approach Solve?

<table>
<thead>
<tr>
<th>Design concepts</th>
<th>Code Quality</th>
<th>Architecture Quality</th>
<th>Design Conformance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Classes, packages, files</td>
<td>Modules, dependencies</td>
<td>Architecture communication styles</td>
</tr>
<tr>
<td>Bridging code and design</td>
<td>Logical and physical element composition</td>
<td>Dependency clusters (semi-automated)</td>
<td>Automated rules (framework-based systems)</td>
</tr>
<tr>
<td>Conformance</td>
<td>ISO standards, maintainability</td>
<td>Modularity, dependencies, design paradigms</td>
<td>Intended architecture and canonical design knowledge</td>
</tr>
</tbody>
</table>

Conformance checking is **feasible** today using a **rules-based** approach to extract design information from **framework-based** systems.

The approach recovers a broader range of architecture views and supports checking a broader range of criteria under conformance.
Project Team Members

John Klein
Principal Member of the Technical Staff, CMU / SEI

Robert Nord
Principal Member of the Technical Staff (Ret.), CMU / SEI

Forrest Shull
Lead for Defense Software Acquisition Policy Research, CMU / SEI

James Ivers
Principal Engineer, CMU / SEI

Lena Pons
Software Architecture and AI Researcher, CMU / SEI

Chris Seifried
Associate Engineer, CMU / SEI

Josh Fallon
Defense Network Analyst, CMU / SEI