Automated Design Conformance During Continuous Integration

NOVEMBER 14–16, 2022

Robert Nord
Principal Researcher

©2022 [DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
Software architecture enables our ability to innovate through extensible design. The end goal—to build systems that provide timely and cost-effective capability to users—is achieved only if the code conforms to the architecture.

This project developed an automated conformance checker prototype that can be used in a continuous integration workflow to discover nonconformances within minutes, instead of the months or years it takes today.

This work helps teams detect problems as they are introduced, allowing faster and more economical realignment of code and architecture and increasing confidence that sustainable code is being delivered.
Automated Design Conformance during Continuous Integration

Why Conformance Matters for Open Software Systems
Software architecture is an abstraction that helps organizations satisfy mission goals and capability needs.

For the implementation to exhibit the desired system qualities, it must conform to the architecture.

Lifecycle View of Software Acquisition https://aaf.dau.edu/aaf/software/
Challenges in Conformance Checking

Conformance Checks
- Inter-construct communication relations originate from a construct and end at infrastructure.
- The intended specification allows communication between construct A and construct B.
- The implemented design has all constructs listed in the intended specification.

State of the Practice
- Component-level manual inspection
- ISO code quality standards, maintainability
- Modularity, dependencies, design paradigms

Challenges
- Automated inspection checks
- System-level checks: constructs and relations
- Conformance to architecture styles

**Automated Conformance Checking during CI**

Conformance is the practice of keeping the architecture and code aligned. Development teams check during **continuous integration** that implementation and architecture are aligned.

Automating conformance checking and feeding back updates to maintain alignment.
Automated Design Conformance during Continuous Integration

Automating Conformance Checking
Infer Design from Code

The key to this work is new research inferring **design** information from source code. Detecting design constructs is challenging due to

- imprecise definitions of abstractions
- variation in implementation
- limits of fact gathering analyses
Using Frameworks to Infer Design

Choosing a framework to realize an architecture style

- constrains code to use framework’s interfaces to realize the style
- supplies structure for implementing the styles chosen for an application

<table>
<thead>
<tr>
<th>Concept</th>
<th>FACE</th>
<th>ROS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish</td>
<td><code>FACE::Create_Connection</code> (*name, pattern, direction, conn_id);</td>
<td><code>NodeHandle::advertise &lt;msg_type&gt;(topic)</code></td>
</tr>
<tr>
<td>Intention to generate messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update</td>
<td><code>FACE::Send_Message</code> (conn_id, data);</td>
<td><code>M_statusPub.publish</code></td>
</tr>
<tr>
<td>Dissemination of messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subscribe</td>
<td><code>FACE::Create_Connection</code> (*name, pattern, direction, conn_id);</td>
<td><code>NodeHandle::subscribe (topic)</code></td>
</tr>
<tr>
<td>Interest receiving messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reflect</td>
<td><code>FACE::Receive_Message</code> (conn_id, data);</td>
<td><code>not explicit in code</code></td>
</tr>
<tr>
<td>Reception of messages</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prototype Conformance Checker

Source Code

Intended Design

Ingest → Augment → Chunk → Infer Constructs → Build Fragments → Naming → Check Conformance

Graph Database

Construct labels

Design fragment graph

Construct names

Nonconformance violations

KEY

Fact extraction
Inference
Synthesis

Raw code graph

Style annotations
Example Input: Source Code and Intended Design

Source Code
- AutoRally Project
- Software for AutoRally Platform
- ~200K C++ code lines
- ROS-framework

Intended Design
- instances of publisher and subscriber design constructs
- message publication and subscription
Example Output: Nonconformances Found

```c
if(transmitPositionRate > 0)
{
    m_transmitPositionTimer = m_nh.createTimer(
        ros::Duration(1.0/transmitPositionRate),
        &XbeeNode::transmitPosition,
        this);

    m_poseSubscriber = nh.subscribe("pose_estimate", 1,
        &XbeeNode::odomCallback,
        this);
}
```

Nonconformances Found

- [INTENDED] Construct CameraAutoBalance does not exist in the as implemented fragment.
- [INTENDED] Construct StatusMonitor does not exist in the as implemented fragment.
- [INTENDED] Construct Runstop does not have the correct type of <EntityName:Subscriber>.
- [INTENDED] Construct ConstantSpeedController does not exist in the as implemented fragment.
- [INTENDED] F_PUBLISHES relation from CameraAutoBalance to Infrastructure is not in the as implemented fragment.
- [INTENDED] F_SUBSCRIBES relation from StatusMonitor to Infrastructure is not in the as implemented fragment.
- [INTENDED] F_SUBSCRIBES relation from Runstop to Infrastructure is not in the as implemented fragment.
- [INTENDED] F_PUBLISHES relation from ConstantSpeedController to Infrastructure is not in the as implemented fragment.
- [INTENDED] F_SUBSCRIBES relation from ConstantSpeedController to Infrastructure is not in the as implemented fragment.
- [INTENDED] F_SUBSCRIBES relation from XbeeNode to Infrastructure is not allowed.

10 nonconformances found.

Tracing Nonconformance to Code

Tracing Nonconformance to Intended Design
Automated Design Conformance during Continuous Integration

Looking Forward
What Practical Problems Does the Approach Solve?

<table>
<thead>
<tr>
<th>State of Practice</th>
<th>Design Conformance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Code Quality</strong></td>
<td><strong>Architecture Quality</strong></td>
</tr>
<tr>
<td>Design concepts</td>
<td>Modules, dependencies</td>
</tr>
<tr>
<td><strong>Bridging code and design</strong></td>
<td>Dependency clusters (semi-automated)</td>
</tr>
<tr>
<td><strong>Conformance</strong></td>
<td>Modularity, dependencies, design paradigms</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.
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**.

**度 of change to prototype**
- reusable
- easier
- harder

**Type of Change to Prototype**
1. New system for known framework
2. New framework for known style
3. New style

<table>
<thead>
<tr>
<th>Stage</th>
<th>Type of Change to Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>New system for known framework</td>
</tr>
<tr>
<td>2</td>
<td>New framework for known style</td>
</tr>
<tr>
<td>3</td>
<td>New style</td>
</tr>
</tbody>
</table>

**What Is Involved in Applying the Approach**

- Ingest
- Augment
- Chunk
- Infer Constructs
- Build Fragments
- Naming
- Check Conformance

**Graph Database**

Nonconformance Violations
Improve Conformance of Implementations to Architectures

An automated design conformance checker integrated into a 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

DevOps practices: continuous feedback and continuous integration and deployment
Project Team Members

Robert Nord  
Principal Member of the Technical Staff, CMU / SEI

James Ivers  
Principal Engineer, CMU / SEI

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

Lena Pons  
Software Architecture and AI Researcher, CMU / SEI

Chris Seifried  
Associate Engineer, CMU / SEI

Josh Fallon  
Defense Network Analyst, CMU / SEI