# Maintaining Architecture Consistency


**Jacek Rosik**

| | |
|---|---|
| **Supervised by** | Jim Buckley |
| **Project Leader** | SPL 2: Muhammad Ali Babar |
| **Project Title** | Conformance Analysis in SPL |


## Architecture Consistency

Software Architecture has become recognized as an important asset in the design of software systems. However, if the implementation does not conform to the designed architecture, then the capabilities of the designed architecture may be lost. Such a loss of consistency is referred to as *Architectural Drift*.

### Goals

- Improve the quality and maintainability of software systems through maintaining consistency between design and implementation;
- Increase architecture awareness of developers to inhibit Architectural Drift.

## Research Questions

- **Is Architectural Drift a real-life issue?** – Characterize Architectural Drift in a real-life software development scenario (extent, causes and implications).
- **Can Architectural Drift be prevented?** – Where can we improve on existing approaches? Can we show we have improved them?
- **What is different within the domain of SPL?** – What are the specific architecture consistency issues unique to SPL? How can they be addressed?

## A Longitudinal case study into Architectural Drift

### Goals

- Empirically characterize architectural drift in a real-life scenario;
- Assess the approach derived from 'best practice' in the literature;
- Empirically derive requirements for architecture consistency tool.

### Approach (provisional)

- Based on a proven design-recovery technique (Reflexion Models);
- The process follows generic template in Fig. 1
  - Design abstraction created before implementation commences;
  - Systematic, iterative checks to verify consistency over the system's development and evolution.

### Results

- **Persistent violations** - Not all architecture inconsistencies introduced were removed;
- **Implementation prompted architecture changes** - Some of violations, were introduced into the design. However rationale for the inclusion was lost over time;
- **Shadowing of violations** - Some of the desired architectural dependencies shadowed undesired dependencies (Architectural Drift)

## Existing approaches to Architecture Consistency

- Mostly based on static, structural analysis techniques;
- Used to analyze module level dependencies;
- Limited information on the processes and tools used;
- Evaluations:
  - Usually done as one-off exercises only;
  - Little insight is given as to whether the discovered violations were removed;
  - Usually performed outside of the organization or on an academic system.

### Typical Process

In most of the approaches consistency checks are performed after a change is implemented or on a finished version.

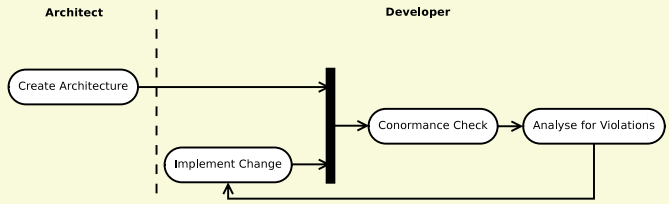


Fig. 1 Typical Architecture Conformance process

1. Create an abstraction of the design, as-designed architecture.
2. Extract the actual, as-implemented architecture.
3. Compare the as-designed and as-implemented architectures.
4. Analyze the resulting view for architectural violations.

## Requirements for Architecture Consistency

- Static, structural evaluation technique as it showed good potential;
- Real-Time architectural feedback and consistency checking during system implementation to inhibit violations before they're introduced;
- Extended notation to better model architecture addressing the shadowing issue.
- Allow for explicit, implementation prompted architecture changes;
- Improve support for SPL:
  - Explicit modeling of product line members;
  - Facilitate consistency checks across the product line members;

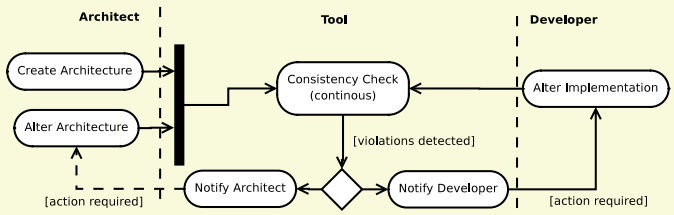


Fig. 2 Revised Architecture Consistency process

## Future Work

- Finalize the requirements for the tool;
- Tool design and implementation;
- More in-depth empirical evaluation;
- Investigate other uses of Architecture Consistency;