Static Extraction and Conformance Checking of Object-Oriented Runtime Architectures

Marwan Abi-Antoun
& Jonathan Aldrich
Carnegie Mellon Univ.

Conformance Checking Strategy

We follow the **extract-abstract-check** strategy:

- **Document as-designed** architecture
- **Abstract as-built** architecture from code
  - **Annotate** code to clarify architectural intent
  - **Extract** sound approximation of runtime object graphs
  - **Abstract** into as-built runtime architecture
- **Check** and **measure** structural conformance
  - Structurally **compare as-built and as-designed** views
  - **Trace to code** unexpected conformance finding

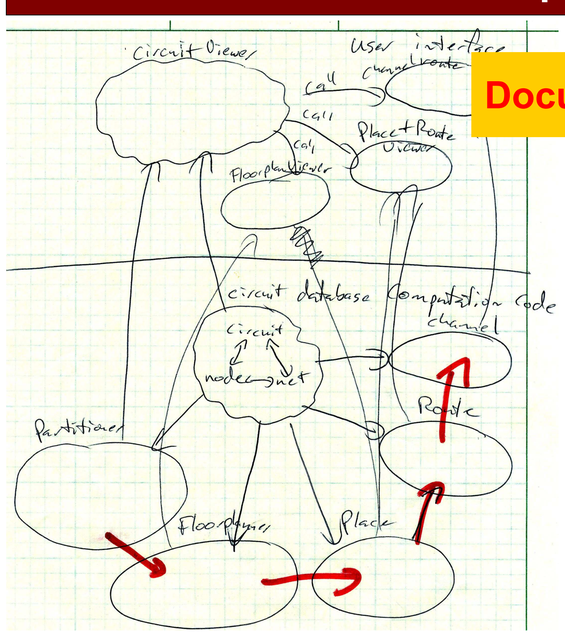
Conformance Checking Analysis

- Consider as-designed **view more authoritative**
- Allow as-built view to contain low-level details
- Account **for all communication** in as-built view that is not in as-designed view
- Include **transitive communication** through elided objects

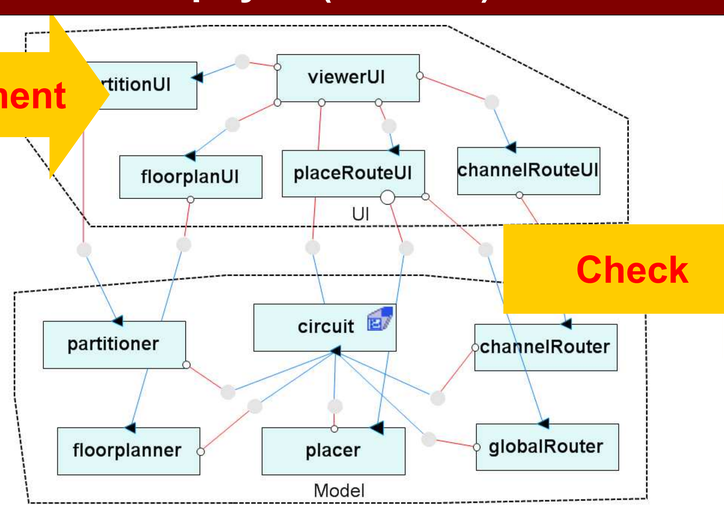
Conformance check highlights **key differences**:

- **Convergence**: node or edge **in both** as-built and in as-designed view ✓
- **Divergence**: node or edge **in as-built** but **not in as-designed** view +
- **Absence**: node or edge **in as-designed** but **not in as- built** view ✗

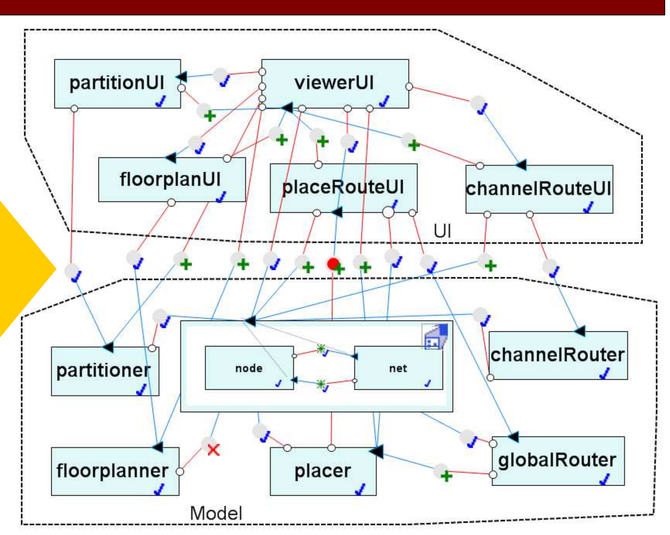
Illustration of End-To-End Approach on Aphyds (8-KLOC)



Aphyds as-designed architecture, drawn by original developer.



In Eclipse AcmeStudio perspective, document as-designed architecture in architecture description language.

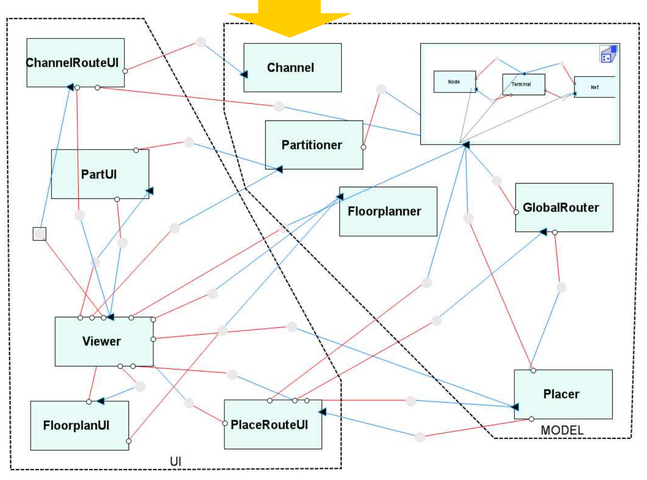


Display conformance results graphically and using metrics using ArchConf conformance checking tool. Study conformance view. Investigate differences.

```
1  class Circuit {
2    public domain DB; // Declare public domain DB
3    domain OWNED; // Declare private domain OWNED
4    DB Node node; // Declare Node reference in DB
5    // Outer annotation is for container; inner one for its elements
6    OWNED Hashtable<String, DB Node> nodes;
7  }
8  class Viewer<M> { // Declare domain parameter M
9    M Circuit circuit; // Declare Circuit reference in M
10 }
11 class Main {
12   domain MODEL, UI; // Declare top−level domains
13   MODEL Circuit circuit;
14   // Bind domain parameter M to actual domain MODEL
15   UI Viewer<MODEL> viewer;
16 }
```

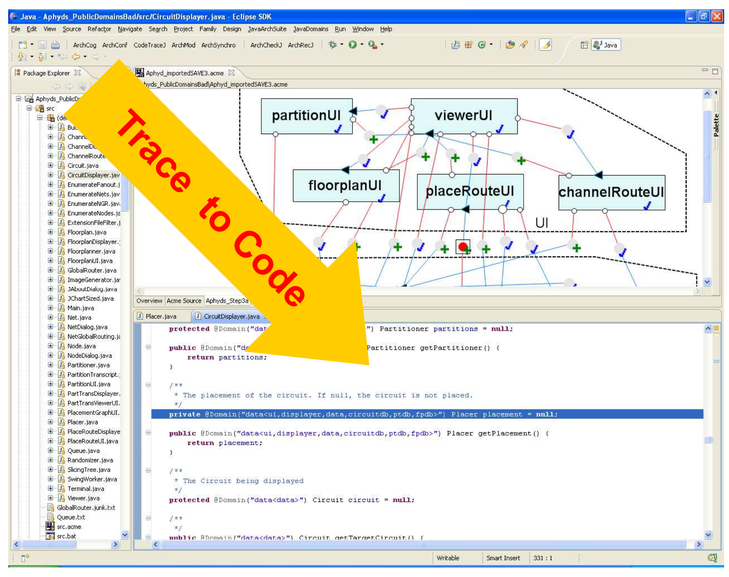
Ownership Domain annotations express:
- object encapsulation
- logical containment
- architectural tiers
- communication permissions

Compare Structurally



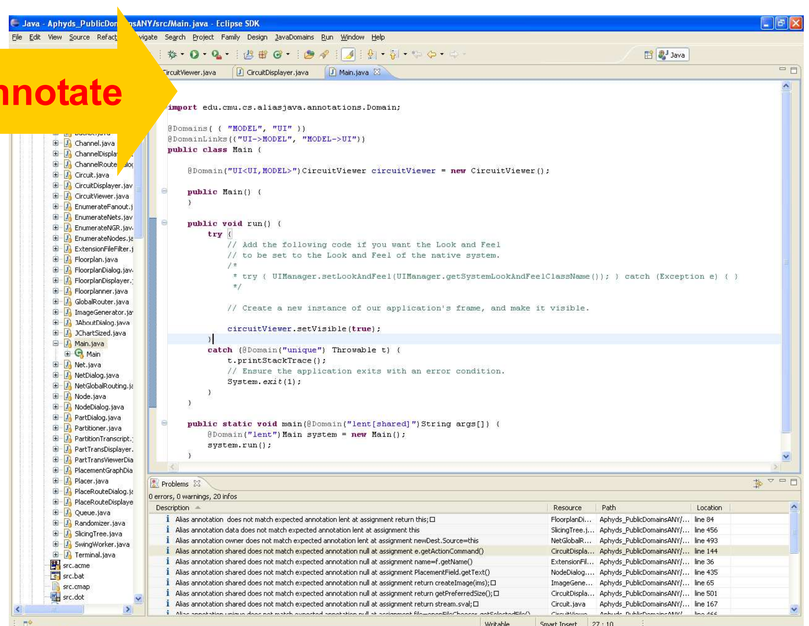
Abstract as-built object graph into an as-built component-and-connector architecture using ArchCog abstraction tool.

Trace to Code



Relate architectural elements to code. Fix serious architectural violations. Or refine as-designed architecture.

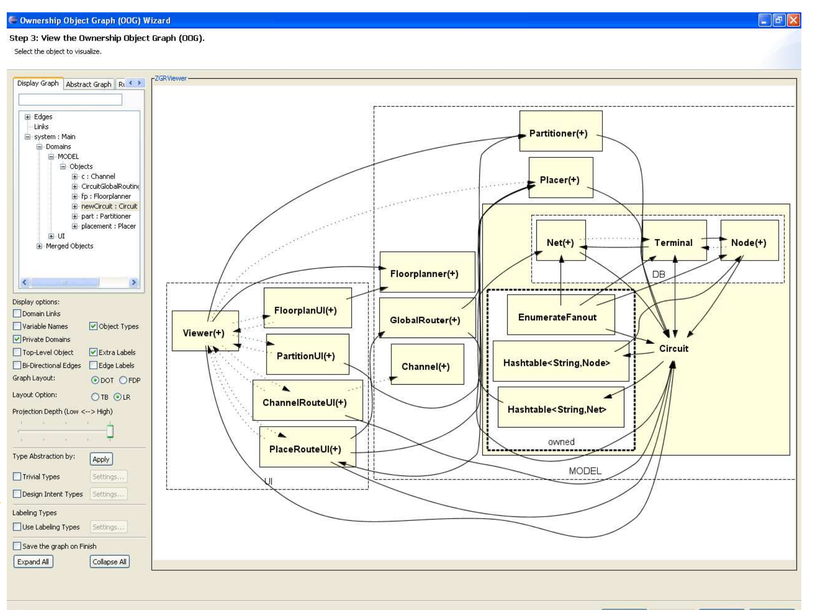
Annotate



In Eclipse Java development perspective, add ownership domains as Java 1.5 annotations. Check using ArchCheckJ typechecking tool. Address warnings in Eclipse problem window.

Abstract

Extract



Extract sound hierarchical object graph from annotated program using ArchRecJ architectural extraction tool.