A runtime architecture shows organization of system in terms of runtime entities and their interactions. Crucial for reasoning about performance, security, reliability, etc.

### Conformance view

![Diagram showing runtime architecture](https://via.placeholder.com/150)

- **How accurate is a manually generated runtime architecture?**
- **A runtime architecture**

### Static Extraction and Conformance Checking of Runtime Architectures

#### How accurately is manually generated runtime architecture?

- **Extraction**
  - Extracts as-built architecture from implementation.
  - Conformance checking compares as-built architecture to as-designed architecture.

#### Requirements on an Adoptable Solution

- **To be adoptable, approach must support**
  - Existing language, frameworks, patterns — i.e., no language extensions.
  - Analysis for object-oriented programs must deal with aliasing inheritance, etc.
  - Dynamic analysis cannot prove that program always satisfies particular property.

**Static checking is ideal if can be achieved and is sound**

- **Soundness**
  - Reveals all entities and relations that may exist at runtime.
  - After-the-fact checking does not assume code generation, monitoring of changes, etc.

#### Ownership Domain

- **An ownership domain groups related objects into a logical cluster**
  - Each object is in exactly one domain.
  - One object can be in one or more domains.
    - Domain name conveys design intent.
    - Domain can represent a runtime tier.
    - E.g., Model, View, Controller.
  - References cross domain boundaries only if there is a domain link between the two domains.

#### Implementation

- **Uses Java annotations.**
- **Does not require language extensions.**
- **Uses Eclipse Java infrastructure.**

**Key insight:** Ownership domain annotations enable the extraction of sound hierarchical runtime object graphs using static analysis.

- **Rewriting Rules:** rewrite runtime graph to annotated program.
  - **Proof of Soundness:** relate entities to extracted runtime graph.

#### Conformance Checking Analysis

- **Conformance check**
  - Highlights key differences between as-built and as-designed views:
    - **Convergence:** node or edge in both as-built and in-as-designed view.
    - **Divergence:** node in as-built but not in as-designed view.
    - **Absence:** node or edge in as-designed but not in as-built view.

**Previous static analyses:**

- Show detailed interactions
- No architectural abstraction
- Do not scale to large programs
- Low-level objects at same level as important objects.
- Sometimes, do not handle aliasing.
- Not comparable to human-generated as-designed architectures.

#### Results

- **System**
  - **Size**
  - **Comments**
  - **Java HotDraw**
  - **15 KLOC**
  - **Designed by OOA&D experts.
  - **HillClimber**
  - **15 KLOC**
  - **Designed by undergraduates.
  - **Aphyds**
  - **8 KLOC**
  - **Original developer drew architecture**

**Conformance checking tool displays results**.

**View conformance**:

- **Conformance results**:
  - **Aliasing component, parallel callback**
  - Callbacks from placer to placement
d thereby connections made bi-directional.

- **Study conformance view**.
- **Investigate differences**.

**Case Studies (JHotDraw, HillClimber, Aphyds)**

- **Analysis for object-oriented programs must deal with**
  - Structural conformance
  - Allows compare as-built and as-designed views.
  - Corresponds to properties:
    - **Display results graphically on as-designed view.**
    - **Compute measure of conformance.**
    - **Trace to code unexpected conformance finding.**
    - **Fix architectural violations in code.**
    - **Adjust as-designed architecture.**

- **Conformance Checking Strategy**

- **Goal is**
  - Not to make the two views identical.
  - Additional sub-structures in as-built architecture.
  - Introspective differences, e.g., renaming.

- **As-designed view more authoritative**
  - Includes components more relevant than omitted ones.
  - Names convey architectural intent.

- **Aphyds as-designed architecture, drawn by original developer.**

**LbGrid 30 KLOC Part of 250-KLOC commercial system**

- **Previous static analyses**:
  - Show detailed interactions.
  - No architectural abstraction.
  - Do not scale to large programs.
  - Low-level objects at same level as important objects.
  - Sometimes, do not handle aliasing.
  - Not comparable to human-generated as-designed architectures.

- **Annotations**:
  - **Object manipulation**.
  - **Logical containment**.
  - **Architectural keys**.
  - **Communication purpose** (domain link).

**Copyright**:

- **Aldrich and Chambers, ECOOP’04**

**Conclusion**:

- **Static Extraction and Conformance Checking of Runtime Architectures**

**Acknowledgments**:

- **Java 1.5 annotations**
  - **Trace to code unexpected conformance finding.**
  - **Lift objects in a domain to another domain.**
  - **Convergence and divergence**
  - **Merge objects**
  - **Provide architectural abstraction by ownership hierarchy and type.**
  - **Summarization**: sum up multiple objects in runtime architecture.
  - **As-built architecture**
  - **As-designed architecture**
  - **Abstract**: abstract architecture from code.
  - **Annotate code**
  - **Soundness**
  - **Object lifting**
  - **Convergence**
  - **Merge objects**
  - **Show domains and objects in domains.**
  - **Unify objects in a domain that may alias.**
  - **Java 1.5 annotations**
  - **Extract**
  - **Abstract**
  - **Renames**
  - **Extracts**
  - **Annotate**
  - **As-designed**
  - **As-built**
  - **Annotate**

**Conclusion**:

- **Java 1.5 annotations**
  - **Trace to code unexpected conformance finding.**
  - **Lift objects in a domain to another domain.**
  - **Convergence and divergence**
  - **Merge objects**
  - **Provide architectural abstraction by ownership hierarchy and type.**
  - **Summarization**: sum up multiple objects in runtime architecture.
  - **As-built architecture**
  - **As-designed architecture**
  - **Abstract**: abstract architecture from code.
  - **Annotate code**
  - **Soundness**
  - **Object lifting**
  - **Convergence**
  - **Merge objects**
  - **Show domains and objects in domains.**
  - **Unify objects in a domain that may alias.**
  - **Java 1.5 annotations**
  - **Extract**
  - **Abstract**
  - **Renames**
  - **Extracts**
  - **Annotate**
  - **As-designed**
  - **As-built**
  - **Annotate**

**Conclusion**:

- **Java 1.5 annotations**
  - **Trace to code unexpected conformance finding.**
  - **Lift objects in a domain to another domain.**
  - **Convergence and divergence**
  - **Merge objects**
  - **Provide architectural abstraction by ownership hierarchy and type.**
  - **Summarization**: sum up multiple objects in runtime architecture.
  - **As-built architecture**
  - **As-designed architecture**
  - **Abstract**: abstract architecture from code.
  - **Annotate code**
  - **Soundness**
  - **Object lifting**
  - **Convergence**
  - **Merge objects**
  - **Show domains and objects in domains.**
  - **Unify objects in a domain that may alias.**
  - **Java 1.5 annotations**
  - **Extract**
  - **Abstract**
  - **Renames**
  - **Extracts**
  - **Annotate**
  - **As-designed**
  - **As-built**
  - **Annotate**

**Conclusion**:

- **Java 1.5 annotations**
  - **Trace to code unexpected conformance finding.**
  - **Lift objects in a domain to another domain.**
  - **Convergence and divergence**
  - **Merge objects**
  - **Provide architectural abstraction by ownership hierarchy and type.**
  - **Summarization**: sum up multiple objects in runtime architecture.
  - **As-built architecture**
  - **As-designed architecture**
  - **Abstract**: abstract architecture from code.
  - **Annotate code**
  - **Soundness**
  - **Object lifting**
  - **Convergence**
  - **Merge objects**
  - **Show domains and objects in domains.**
  - **Unify objects in a domain that may alias.**
  - **Java 1.5 annotations**
  - **Extract**
  - **Abstract**
  - **Renames**
  - **Extracts**
  - **Annotate**
  - **As-designed**
  - **As-built**
  - **Annotate**

Marwan Abi-Antoun (Advisor: Jonathan Aldrich)

Aldrich and Chambers, ECOOP ’04

Carnegie Mellon University