Trinity: A Language for Multi-View Architecture Description and Control

Maddie Kirwin
kirwinma@grinnell.edu

Selva Samuel
ssamuel@cs.cmu.edu

Jonathan Aldrich
Jonathan.Aldrich@cs.cmu.edu

Software Architecture

the "fundamental organization of a system embodied in its components, their relations to each other, and the environment"

Architecture Views

Module or Code

Modules: principal units of implementation

- Used to explain system functionality + structure of code base
- A blueprint for code construction and incremental development
- Analysis of code dependency

Component and Connector (CnC)

Components: elements that have some runtime presence (processes, objects, clients, servers).
Connectors: components' pathways of interaction (protocols, information flows).

- Show how the system works
- Guide development around structure & behavior of runtime elements
- To reason about performance and reliability

Deployment

Deployment View: a mapping between software and nonsoftware elements in the former's environment.

- Analyzing actual runtime performance, reliability, and security
- SW elements: CnC elements
- Environmental elements: hardware, network elements, and their capabilities

The Problem

It is hard to determine whether the logical relationships between entities in architecture diagrams are present in system implementations.

Previous Solutions

Architecture Description Languages (ADLs)

- (-) *Description:* Inferred by the name, ADLs only describe software architectures; they do not prescribe, or **enforce conformance** to them
- (+) *Analysis:* ADLs are focused on system analyses
- (+) *Formal Notation:* Currently, ADLs are the most formal mainstream architecture tools available

ArchJava Java extension unifying SWA and implementation

- (-) *Application:* Does not do anything interesting with SWA (i.e. checks)
- (-) *Distributed Systems:* No support for distributed systems
- (-) *Multiple SWA Views:* Lacks support for multiple architecture views; focuses only on Component-and-Connector view.

Trinity's Approach

- Makes software architecture a **"live" component** of Trinity systems
- Trinity enforced **architecture conformance** complements ADL analyses
- Directly translate the conceptual entities from SWA views into **code-enforced constructs**
- Support for **all three software architecture views** (module or code, CnC, and deployment).
- Support for architecture conformance in **distributed systems**.

Implementation Concepts

Architecture concepts are runtime entities in Trinity

<This is an explanation of the different architecture entities in Trinity, not specific to an example>
Component
Connector
entryPoints(?)

Design

Demonstrated Principles

- Readability
- Reuse/adaptability
- Communication integrity, especially in distributed systems

<INSERT EXAMPLE TRINITY CODE OF EXAMPLE ARCH.>
* describe each component

<INSERT SOFTWARE ARCHITECTURE DIAGRAM OF EXAMPLE>

Example