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

**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**

#### Modules: principal units of implementation Or Used to explain system functionality + Module structure of code base A blueprint for code construction and incremental development Analysis of code dependency Components: elements that have some runtime presence (processes, objects, clients, servers). Component **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 View:** a mapping between eployment software and nonsoftware elements in the former's environment.

Analyzing actual runtime performance,

network elements, and their capabilities

reliability, and security

SW elements: CnC elements

Environmental elements: hardware,

## 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 architecture and implementation

- (-) Application: Does not check for conformity to architecture
- (-) Distributed Systems: No support for distributed systems
- (-) Multiple Views: Lacks support for multiple architecture views; focuses only on Component-and-Connector view

## Trinity's Approach

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

# Design

## **Implementation Concepts**

Architecture concepts are translated into runtime entities in Trinity

#### **Trinity Architecture Components**

**Component:** a runtime entity that may interact with other components through ports.

**Connector:** interaction pathways that join two compatible component ports.

**Port:** component access points that enable interaction with other components.

entryPoints: a program starting point that enables execution.

# Demonstrated Principles

Trinity's design demonstrates the following principles:

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

Example

# <INSERT EXAMPLE TRINITY CODE OF EXAMPLE ARCH.>

< INSERT SOFTWARE ARCHITECTURE DIAGRAM OF EXAMPLE>

\* describe each component

component Client port getInfo: requires CSIface

component Server port sendInfo: provides CSIface

external component DB port dblface: target DBModule

connector JDBCCtr val connectionString: String

> architecture components RequestHandler ch DB db

> > connectors JDBCCtr jdbcCtr

attachments connect rh.dblface and db.dblface with jdbcCtr

bindings sendInfo is rh.sendInfo architecture components Client client Server server

> connectors JSONCtr jsonCtr attachments Connect client.getInfo and server.sendInfo with jsonCtr

entryPoints Client: start