

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

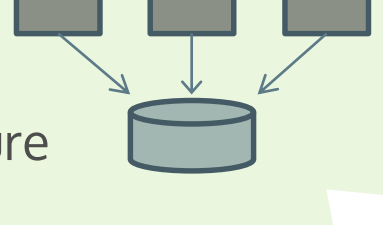
- Used to explain system functionality + structure of code base
- A blueprint for code construction and incremental development
- Analysis of code dependency



Component and Connector (CnC)

**Components:** elements that have some runtime presence (processes, objects, clients, servers).  
**Connectors:** components' pathways of interaction (protocols, information flows).  
**Ports:** component interfaces that define possible interaction with components.


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



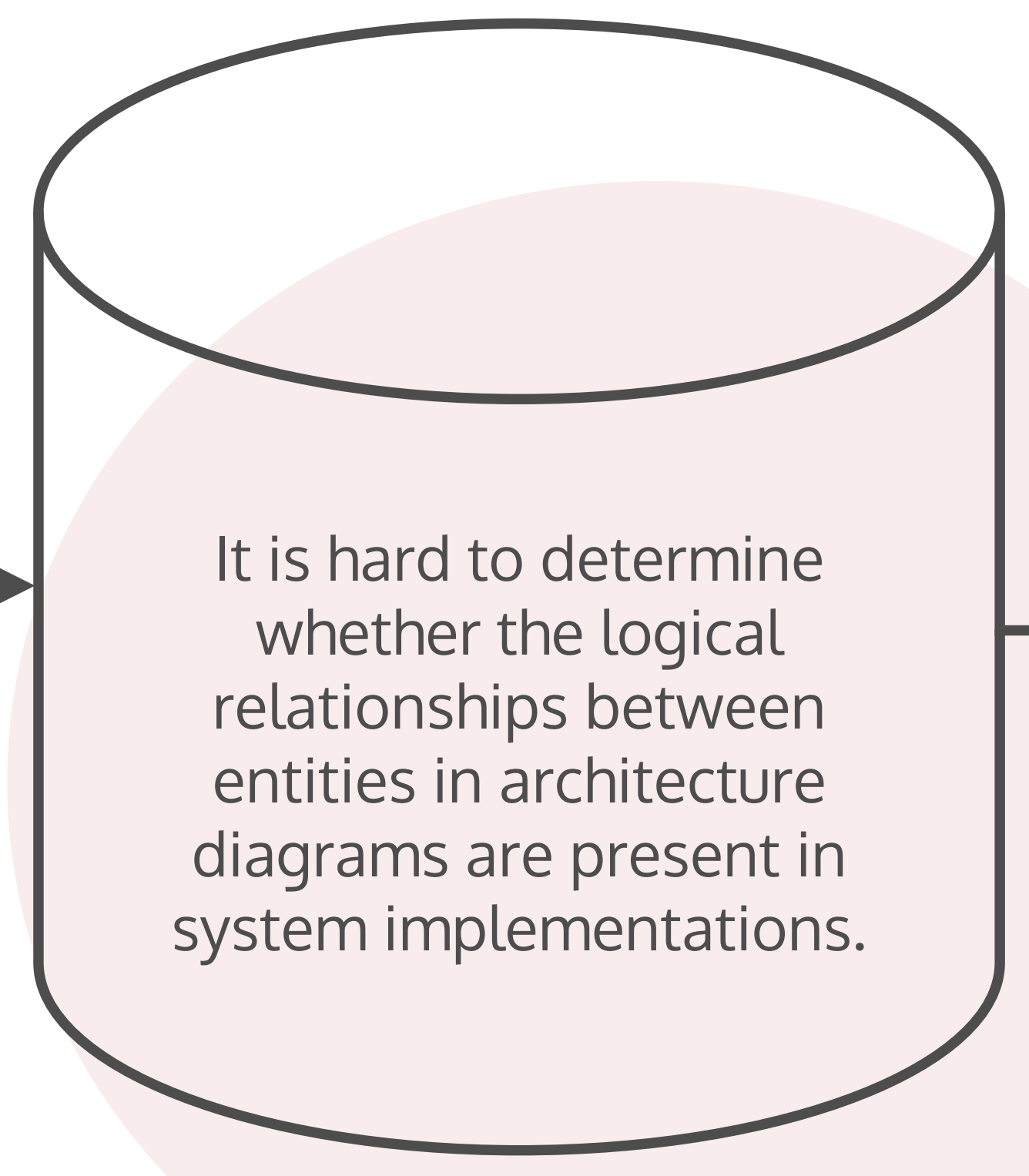
Deployment

**Deployment View:** a mapping between software and non-software 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



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

- (+) *Conformance:* Checks for architecture conformity
- (-) *Distributed Systems:* No conformance checks in distributed systems (ArchJava supports multiple systems via custom connectors, but does not enforce conformity)
- (-) *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**  
System architecture is contained in a single file and is prescriptive, uniting design and implementation.
- Reuse and Adaptability**  
Compatibility checking and code generation make switching, adding, and removing architecture elements easier and more secure.
- Communication Integrity in Distributed Systems**  
jfdlkasjfdklasfjs

Example

```
component Client
  port getInfo: requires CSInterface

component Server
  port sendInfo: provides CSInterface

external component DB
  port dbInterface: target DBModule

connector JDBCContr
  val connectionString: String

architecture
  components
    RequestHandler rh
    DB db

  connectors
    JDBCContr jdbcCtr

  attachments
    connect rh.dbInterface and db.dbInterface
    with jdbcCtr

  bindings
    sendInfo is rh.sendInfo
```

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

<INSERT SOFTWARE ARCHITECTURE DIAGRAM OF EXAMPLE>

```
architecture
  components
    Client client
    Server server

  connectors
    JSONCtr jsonCtr

  attachments
    Connect client.getInfo and
    server.sendInfo with jsonCtr

  entryPoints
    Client: start
```