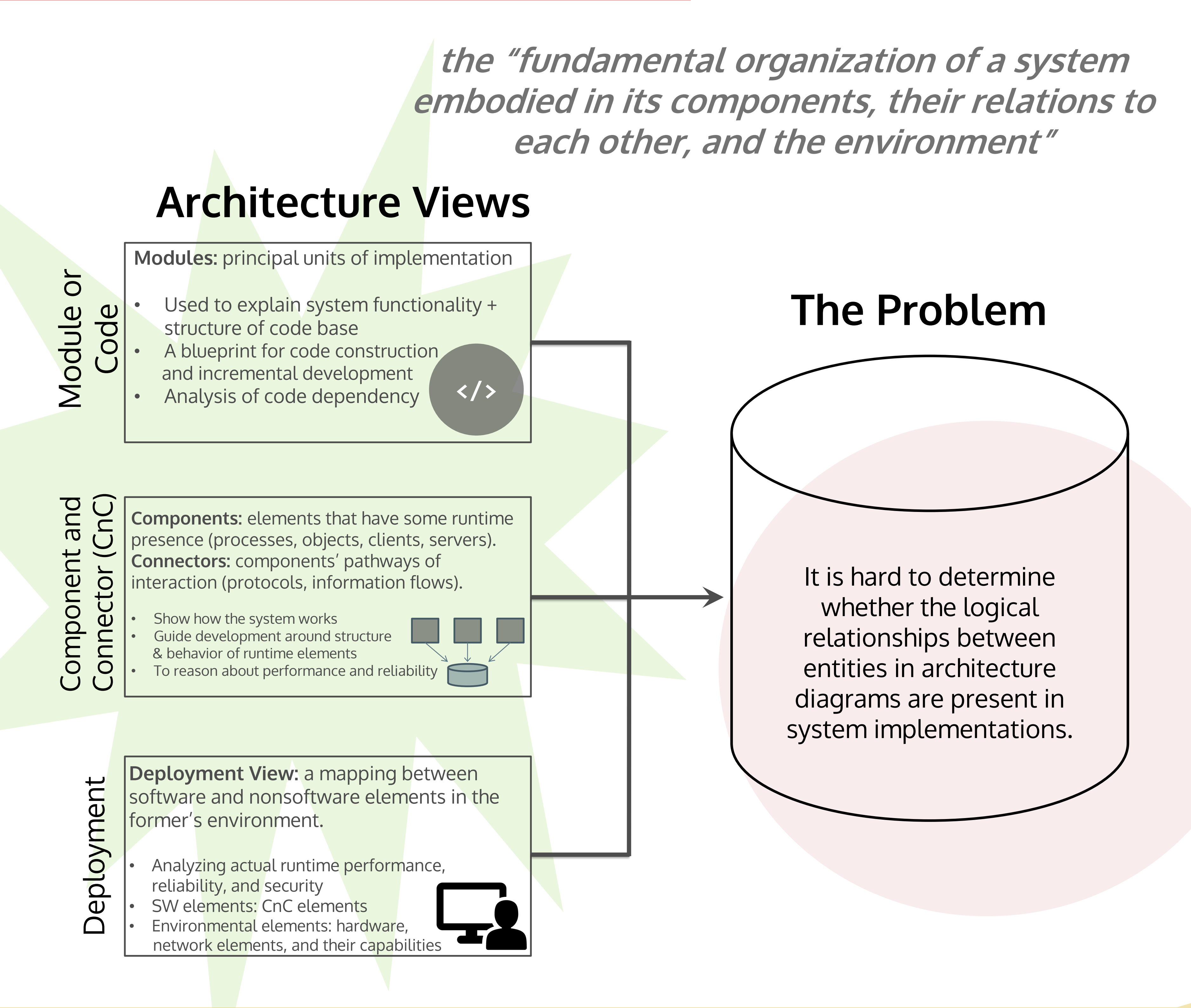


Software Architecture



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(?)

Demonstrated Principles

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

Design

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

<INSERT SOFTWARE ARCHITECTURE DIAGRAM OF EXAMPLE>

Example