

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

Subtitle Text, if any \*

Name1  
Affiliation1  
Email1

Name2    Name3  
Affiliation2/3  
Email2/3

## Abstract

This is the text of the abstract.

## 1. Introduction

## 2. Design

Trinity is designed to unify software architecture design and implementation for not just a single architecture view, but in all three (module, component-and-connector, and deployment).

To demonstrate how Trinity makes software architecture live in Wyvern systems, we have implemented a simple 3-tier web application whose abridged code is shown in Figure 1. In overview, a database is accessed by a server that handles requests from a client. The example architecture contains two components, the client and server. As in more theoretical software architecture, components are runtime entities that may have ports that act as access points to interact with other components. Our example server and client each have complementary ports, `sendInfo` and `getInfo` respectively, that enable interact; here, the server can send information to the client using a JSON connector responsible for serialization and deserialization. Note that Trinity connectors enable the joining two compatible component ports, analogous to ports in software architecture. The architectures attachments section actually/physically connects the client and the server using their matching ports and the JSON connector.

The client and server have their own component-specific architectures given before the general architecture code.

Both components have corresponding ports that depend on a client-server interface, denoted by the `requires/provides CSIFace` types of each port. This interface is required of a server by the client, as stated by the `requires CSIFace` type, and the server fulfills it, as shown by the `provides CSIFace` server port type. The database is an external component of the server, which differs from a component in that the programmer does not provide its source code. `port dbIface`: requires `DBModule` The database and server are connected by a JDBC connector. This is used within a sub-architecture of the server that connects a specified request handler and the database. Finally, the bindings section of the servers sub-architecture specifies that the client-facing port, `sendInfo`, of the server component is indeed the same `sendInfo` port of the request handler.

## A. Appendix Title

This is the text of the appendix, if you need one.

## Acknowledgments

Acknowledgments, if needed.

## References

P. Q. Smith, and X. Y. Jones. ...reference text...

---

\* with optional subtitle note