



Component-based software development relies fundamentally on the quality of the components of which a system is composed and their configuration, yet little technology exists for component-based quality assurance. To predictably and reliably build complex systems by composing components, components must be analyzed not only independently but also in the context of their connection to other components. Analysis should be coordinated, therefore, at a higher level of abstraction — e.g., at the software architecture level, where components, connectors, and their configuration are better understood and intellectually tractable.

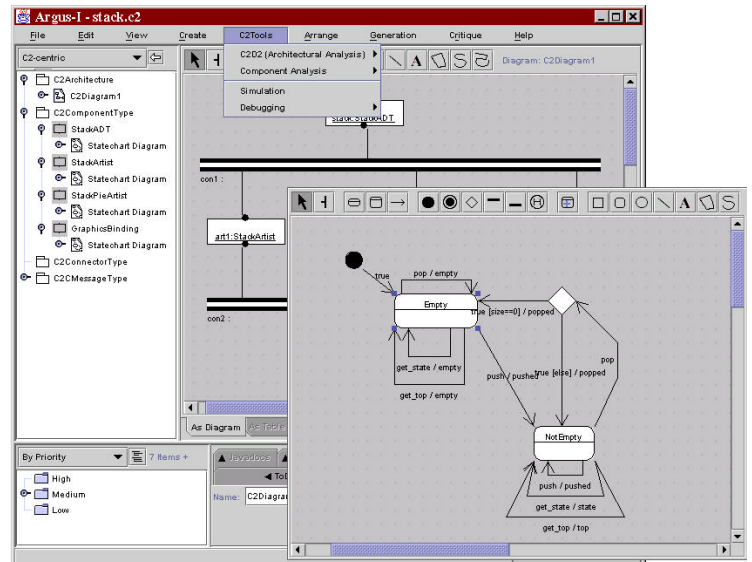
ARGUS-I is a comprehensive set of specification-based analysis tools focusing on both the component and architecture levels. ARGUS-I facilitates iterative and evolvable analysis during architectural specification and implementation. Both structural and behavioral analyses are accomplished by a synergistic combination of static and dynamic techniques. While static analysis can, for instance, detect incompatibility between the data exchanged between components and verify architectural adherence to design heuristics and style rules, dynamic analysis may be required for revealing defects in the dynamic component interaction and communication behavior between components.

The current version of ARGUS-I works with software architectures in the *C2-style*¹ but augments the C2 architecture description with component behavior specification described by Statecharts. Some of the most important analysis capabilities of ARGUS-I are briefly described hereon.

Component Specification Analysis

Component-level specification analyses are performed statically (verifying interface consistency and the relationship to the statechart behavior specification) and dynamically (evaluating component behavior through statechart simulation).

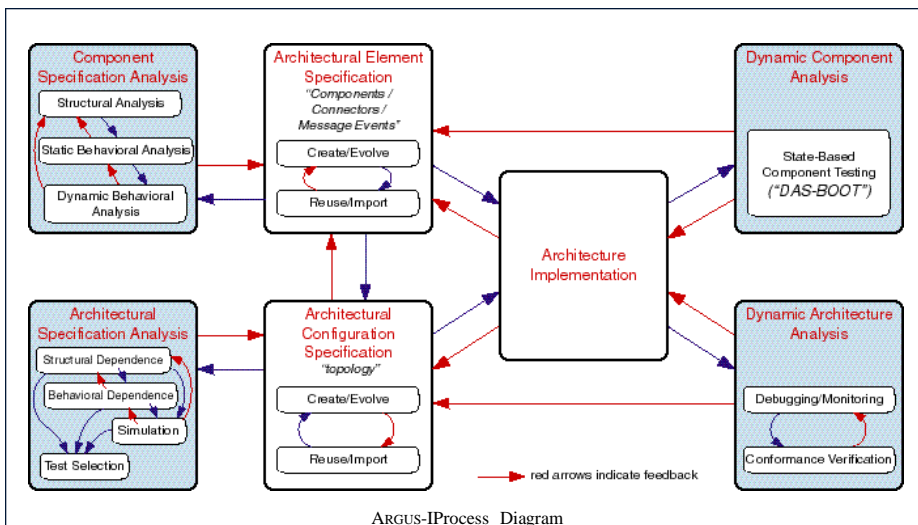
Static structural analysis performs type checking and internal consistency checking over the component interface specification, revealing inconsistencies such as interfaces not mapped to an operation (and vice-versa) and parameter mapping mismatches.



Architecture and Component Specification

ARGUS-I, the “all-seeing”, had one hundred eyes, at least some of which were always awake and watching. ARGUS-I was known for having killed the remarkable bull that ravaged Arcadia. After ARGUS-I was slain, Hera placed his eyes on the feathers of the peacock.

Static behavioral analysis over each component’s statechart checks not only the statechart syntax and semantics, but also state reachability and consistency between the component’s interface and the statechart. Moreover, *dynamic behavioral analysis* enables execution of the component’s statechart based upon event traces defined by the software architect, facilitating early discovery of problems in the component’s behavioral specification.



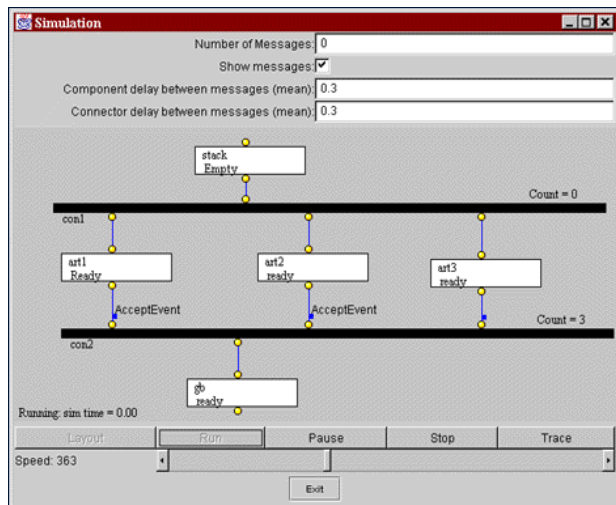
ARGUS-I Process Diagram

Architecture Specification Analysis

Architecture-level specification analyses are performed statically (verifying structural and behavioral dependencies among components) and dynamically (evaluating the architecture configuration through simulations).

Dependence analysis of the architecture specification is performed across the component interfaces, determining the relationships between components and revealing interface mismatches. The results can be presented in textual and graphical representations.

Simulation of the architecture specification serves widely variant objectives, such as discovering design errors, optimizing or fine-tuning a system, or simply better understanding a complex system through analysis. In ARGUS-I, the main purpose of simulating the architecture specification is to detect problems early in the architecture design and to evaluate its performance. Simulations are based on the components' behavior as specified by their statecharts and the architecture configuration. Traces can be visualized both graphically and textually. Statistical measures can be visualized and distinct simulations can be compared.



Architecture Simulation

Dynamic Component Analysis

Component-level dynamic analysis is performed by testing components to reveal inconsistencies between the implemented component and its statechart specification.

Dynamic behavioral analysis is accomplished by state-based component testing using DAS-BOOT². With DAS-BOOT, the developer/tester can verify the behavior of a Java class based on and against its statechart description, which models the class behavior. Test drivers are generated, which execute the code and compare test results to the statechart behavior.

Dynamic Architecture Analysis

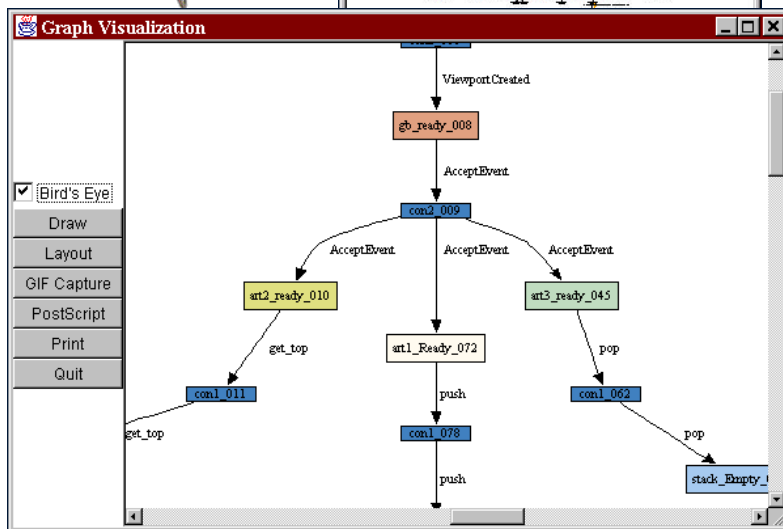
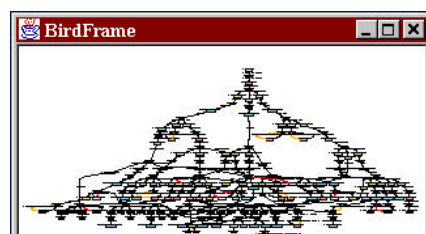
Architecture-level dynamic analysis is important to verify and analyze whether all components behave as specified when integrated. In ARGUS-I, dynamic architecture analysis is based on *architecture debugging*, *monitoring*, and *dynamic conformance verification* between specification and implementation.

While *debugging* and *monitoring*, the developer can visualize event traces at both architectural and component levels, as well as setting breakpoints at the component ports and removing/adding/modifying message events. Statistical measures about message traffic are made available.

Conformance verification between architecture specification (configuration) and implementation (integrated components) is performed in relation to both component interface and behavior. During execution, *interface conformance* is verified by comparing messages and parameters, while *behavioral conformance* is verified by comparing system behavior to a parallel execution of the state machines specified for the components.

¹ C2: <http://www.ics.uci.edu/pub/arch/C2.html>

² DAS-BOOT: <http://www.ics.uci.edu/pub/rosatea/das-boot.htm>



Simulation Trace



The **ROSATEA** tools are research prototypes, some of which have been successfully transitioned into use on real projects, but not yet as commercial systems. These tools were developed by the Research Organization for Specification- and Architecture-based Testing E (&) Analysis at the University of California at Irvine. The work was done in conjunction with the Perpetual Testing Projects sponsored by the DARPA's EDCS program.



The **Argus-I** research and development effort is sponsored in part by: the Air Force Materiel Command, Air Force Research Laboratory, and the Defense Advanced Research Projects Agency under agreement number #F30602-97-2-0033. The views and conclusions contained herein are those of the researchers and should not be interpreted as representing the official position or policy, either expressed or implied, of the U.S. Government, AFMC, Rome Laboratory, DARPA, or the University of California, and no official endorsement should be inferred.

Freely Available Software

Information about UC Irvine's *Research Organization for Specification- and Architecture-based Testing E (&) Analysis* (ROSATEA), as well as its software, is available at: <http://www.ics.uci.edu/~rosatea/>

Contact Information

Professor Debra J. Richardson
Information and Computer Science
University of California
Irvine, California 92697-3425

url: <http://www.ics.uci.edu/~djr>
email: djr@ics.uci.edu
voice: 949-824-7353
fax: 949-824-1715