RL-TR-96-143
Final Technical Report
July 1996

INTERMEDIATE ARCHITECTURAL REPRESENTATION FOR THE KBSA-ADM

Florida International University

Yi Deng, Paul Attie, and Michael Evangelist

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be releasable to the general public, including foreign nations.

RL-TR-96-143 has been reviewed and is approved for publication.

APPROVED: Joseph A Carozzoni

JOSEPH CAROZZONI
Project Engineer

FOR THE COMMANDER: John A. Graniero

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify Rome Laboratory/ (C3CA), Rome NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1996	Final Sep 93 - Dec 95

4. TITLE AND SUBTITLE
INTERMEDIATE ARCHITECTURAL REPRESENTATION FOR THE KBSA-ADM

5. FUNDING NUMBERS
C - F30602-93- C-0247
PE - 62702F
PR - 5581
TA - 27
WU - 75

6. AUTHOR(S)
Yi Deng, Paul Attie, and Michael Evangelist

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Florida International University
University Park
Miami FL 33199

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory/C3CA
525 Brooks Rd
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-96-143

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Joseph Carozzoni/C3CA/(315)330-7796

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
The Knowledge-Based Software Assistant (KBSA) originally assumed automatic programming a fundamental support technology. The KBSA Concept Demonstration project and the last decade of research in the Knowledge-Based Software Engineerng (KBSE) community have demonstrated that practical automatic programming is much farther away than initially anticipated. The KBSA CDS showed that a complete functional specification of a system written in a high-level specification language can be effectively manipulated, evaluated, verified, and executed in a single process environment. It also showed that fully automatic transformation of such a specification into efficient target code running in a distributed computing environment is still a distant vision. Many architectural and non-functional constraints that are not expressed in current KBSA functional specifications must be taken into account in the process of constructing and implementing a system. This report describes the research to partition the specification-to-code transformation process into discrete, interactive (user-assisted) stages. Each stage introduces new architectural and non-functional constraints and produces a more complete and refined system model, with the last stage producing an efficient system implementation. This research focused on the domain of real-time, distributed systems.

14. SUBJECT TERMS
Parallel programming, Automatic programming, Formal methods, Knowledge-based systems

15. NUMBER OF PAGES
24

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Intermediate Architectural Representation for the KBSA ADM

U.S. Air Force Rome Lab Contract No. F30602-93-C-0247

Final Scientific and Technical Report

Yi Deng, Paul C. Attie, and Michael Evangelist
School of Computer Science
Florida International University
Miami, University Park, Miami, FL 33199
{deng,attie,wme}@fiu.edu

1 Overview

The Knowledge-Based Software Assistant (KBSA) originally assumed automatic programming as a fundamental support technology. The KBSA Concept Demo (CD) project and the last decade of research of the KBSE community have demonstrated that practical automatic programming is much farther away than initially expected. The KBSA/CD showed that a complete functional specification of a system written in a language such as ERSLa can be effectively manipulated, evaluated, verified, and executed in a uni-processor computing environment. It also showed that a fully automatic transformation of such a specification into efficient target code running in a distributed computing environment is still a distant vision. Many architectural and non-functional constraints that are not expressed in current KBSA functional specifications must be taken into account in the process of constructing and implementing a distributed system.

This project was to investigate a pragmatic and achievable approach to addressing the above problem in a time-frame that allows for transferring results to the KBSA Advanced Development Model (ADM) project [ADM92]. More specifically, our objective was to fill the architectural information gap in the KBSA approach by developing formal methods and prototype technologies for architectural specification, refinement, and analysis of progressive architectural designs. The results of the project are to serve as a basis for a more practical KBSA approach, in which the specification-to-code transformation process is based on discrete, interactive stages with knowledge-base support. Each stage will introduce new architectural and non-functional constraints, and will produce a more complete and refined system model. The target application domain for our project

is concurrent and distributed real-time systems, which are typical in critical applications, e.g. C3I and air traffic control systems, of the Air Force.

The purpose of this final scientific and technical report is to systematically document our activities and accomplishments for the entire duration of the aforementioned project from September 22, 1993 to December 31, 1995. The project was composed of two parts: *basic research* and *technology development*. The rest of the report is organized accordingly.

2 Basic Research

2.1 Syntactic Transformations of Distributed and Concurrent Programs

2.1.1 Overview

Formal program verification is widely accepted as a means of guaranteeing the correctness of concurrent programs. The practical utility of formal verification is limited by numerous factors — for example, the large amount of manual labor required, the possibility of proof errors, the lack of personnel trained in proof techniques, and so on. It is also clear that post-development verification alone does not provide a systematic software development process. These observations motivated the approach taken by the Knowledge-Based Software Assistant [GBCR83] project, which has investigated techniques for the automatic generation of code from specifications. Chief among these techniques is the use of *transformations* to successively refine specifications into code. Unfortunately, the KBSA project has concentrated on centralized, sequential systems, while most large systems today are distributed and concurrent, raising a host of previously unaddressed issues. For example, most distributed systems are nonterminating, and the *correctness* of a nonterminating system cannot, obviously, be specified as a relation between initial and final states. Instead, the ongoing, *temporal* behavior of the system must be specified. Furthermore, suitable representations for the intermediate results of the refinement process must be devised. Such representations must take into account many aspects which are absent in centralized, sequential, systems, chiefly the representation of the system architecture, and the communication behavior among the various modules of the architecture.

In this project, we have concentrated on the development of *syntactic transformations* which can be used to generate programs from specifications in an interactive, designer-assisted process.

Such transformations are mechanizable and, therefore, do not involve significant amounts of manual labor. Using this approach, the process of development may be viewed as the human-assisted high-level compilation of a specification into code. In the foreseeable future, human creativity will remain essential for choosing an appropriate transformation to apply at each stage. But verifying that a transformation preserves desired properties is unnecessary, in our approach, because this is guaranteed by the fact that the transformations are *correctness-preserving*. We have focused on the correctness property of *deadlock-freedom*; a system is deadlock-free if and only if it never reaches a state in which no process is capable of making a transition. Clearly, this is a crucial property for any nonterminating distributed system, and a pre-requisite for demonstrating that more general safety and liveness properties are preserved. We have shown that all of our transformations preserve deadlock-freedom. We expect to extend our work to show that other temporal properties are also preserved by our transformations.

Our model of concurrency is based on the **Raddle** [At87] language. In this model, a concurrent program is the parallel composition of some number of sequential processes. Each sequential process is built up from primitive "actions" using the operators "sequence" (perform actions one after the other), "nondeterministic choice" (perform either one action or another), and iteration. If two or more processes attempt to execute an action with the same name, then *all* such processes are required to participate in the execution of the action. Such a "multiparty interaction" provides the means of interprocess synchronization and communication in our model.

2.1.2 Research Accomplishments

Our research accomplishments in this project consist of designing a set of transformations for distributed systems. We have also formally proven that all of our transformations preserve the important property of deadlock-freedom. Our transformations can be classified into two categories: *sequence-introduction* and *choice-merge*.

A. Sequence Introduction Transformations [ADDE95]

Sequence introduction transformations are used to refine a program action into a sequence of "smaller" actions. Our work on sequence introduction transformations has reached a relatively complete stage, and is reported in "Automating the Refinement of Specifications for Distributed

Systems via Syntactic Transformations," [ADDE95]. In this paper, we present the following two transformations:

Right-sequence introduction transformation The right-sequence introduction transformation allows us to introduce a new action d into a concurrent program immediately after (to the "right" of) a preexisting action c. Thus the original action c is refined into a sequence of two actions, c and then d.

Left-sequence introduction transformation The left-sequence introduction transformation allows us to introduce a new action d into a concurrent program immediately before (to the "left" of) a preexisting action c. Thus the original action c is refined into a sequence of two actions, d and then c.

B. Choice Merge Transformations [AD95]

Choice merge transformations are used to take two "small" deadlock free programs (of a certain constrained syntactic form), each of which implements some well-defined part of the overall system functionality, and merge them into a "larger" deadlock free program which implements the "combined functionality" of the small programs. Thus these transformations can be used to *build* large complex programs from smaller ones.

The work on choice-merge transformations is in a more preliminary stage, and is given in "Construction of Concurrent Programs by Syntactic Merging (draft)," [AD95]. We present six transformations in this paper. Roughly speaking, these transformations take two sequential processes and merge their bodies using the "nondeterministic choice" operator. Since a concurrent program is the concurrent composition of some number of sequential processes, the transformations are extended to concurrent programs by simply applying them to some subset of the sequential processes in the concurrent programs.

2.2 Architectural Representation for Real-Time Distributed Systems: G-Nets

2.2.1 Overview

Critical modern computer systems, e.g., air traffic control and radar tracking systems, are generally not only large and complex but also concurrent in their computational behavior. Many are either

embedded in another system or required to interact heavily with other systems, often through a fault-prone communications environment. The design problem is made even more difficult when a system has all these characteristics and is, in addition, deadline-driven. *Real-time systems,* which are reactive systems defined by the last characteristic, frequently possess most of the others, as well. They pose one of the most interesting challenges to the software engineering community, because of the numerous interlaced requirements that designers must meet simultaneously.

Formal architectural modeling and analysis play an important role in developing such systems, and in ensuring the dependability and extensibility of the systems. Research on real-time system modeling were conducted along two separate tracks. The first track is formal methods, which offer promise as a foundation for real-time development. Formal representations provide rigor and analytical capability, which help reduce complexity and increase understanding; *practical* formalisms are executable and, therefore, support simulation and modeling — important when system complexity can exceed analytical feasibility. In the past two decades, researchers have extensively investigated the theory of formal methods for real-time (RT) systems but have paid less attention to developing a true engineering methodology. The second track is architectural specification and description languages, like Rapide [LUC95a] and ROOM [SEL94a], which provide syntactically well-defined high-level models and languages to describe the architecture or architectural patterns of real-time systems. These languages, however, normally do not have a solid mathematical foundation, and mathematically sound formal semantics.

A major aspect of this project is to develop a formal architectural representation that combines the strengths of the two areas of research. The starting point of our investigation is the G-Net model [DEN90a], based on which we have developed an executable formalism called NOAM (Net-based Object-oriented Architectural Model). [1] Building on the theoretical foundation of extended Petri nets, NOAM integrates a hierarchical architectural model with a generalized object model to create a coherent formal representation. It not only allows the precise specification of complex system properties and behavior but also supports the incremental, hierarchical modeling and decomposition of distributed system architectures.

[1]The early versions of NOAM are called extended G-Nets and RTG-Nets (Real-Time G-Nets).

2.2.2 Research Accomplishments

A summary of the research papers and reports generated from this project is provided below, and appropriate citations are given.

A. Transformation from G-Net specification to PrT nets [DCL94].

In this paper [DCL94] a formal transformation technique is reported, which translates a G-Net specification to a semantically equivalent PrT-net. The resultant PrT-Net can then be formally analyzed. The practical significance of this approach in fault-tolerant systems and distributed multimedia systems is discussed. The transformation technique also gives formal semantics to G-Net specifications in terms of PrT nets.

(This paper has been published in the *International Journal of Software Engineering and Knowledge Engineering*, Vol. 4, No. 4, 1994, 427-450.)

B. Use Formal Techniques to Support the Design of Concurrent Object-Oriented Systems [DL95].

As a major class of OO systems, concurrent/distributed OO systems introduce additional complexities into system design, and make it even harder to validate the design. Therefore, the need for formal and systematic support in the design becomes even more apparent. Despite its importance, few studies about using formal techniques to support concurrent OO system design can be found in the literature. This paper is an attempt to address the problem. The contribution of the paper is twofold: First, an executable specification technique called *CooDS*, which is well suited for the specification and analysis of concurrent OO systems, is presented. Second, an approach for using the technique to support systematic evolution and refinement of OO designs is proposed. An example is given to illustrate the approach.

C. An Environment for Specification, Simulation and Analysis of Distributed Object-Oriented Systems [LD95]

Executable specification provides an effective means to support software development. In addition to its well-defined semantics and analytic capability, an executable design specification can be executed to simulate system behavior, thus helping to detect and eliminate design faults early on. For this purpose, G-Nets, a Petri Net-based object-oriented formalism, has been proposed as an

executable specification technique for the design of distributed object-oriented (OO) systems. In this paper, we present a distributed environment for the specification and simulation of distributed OO systems based on G-Nets. The environment combines the OO visual formalism with a distributed execution platform to provide a realistic and user friendly tool to simulate and analyze an OO system design at an early stage of the development. The system is interactive and graphic-oriented, which helps to reduce the difficulties in constructing a system specification, and provides a visual means for the user to monitor the concurrent execution of and communication between G-Net objects. An example is provided to demonstrate the capabilities and functionalities of the system.

(This paper has been published in the *Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering*, Rockville, MD, June 22-24, 1995, 402-410.)

D. A Formalism for Architectural Modeling of Concurrent Real-Time Systems [DDAE95]

Formal methods offer promise as an approach for complex real-time systems design, because of their precision and analyzability (not to mention executability). Over the last two decades, researchers have extensively investigated the theoretical foundations of formal methods but have paid less attention to creating a true engineering methodology. This research, which focuses on the issues of scalability and architectural specification, is a step in that direction. We present an executable formalism called NOAM (Net-based Object-oriented Architectural Model) for the architectural modeling of distributed real-time systems. Building on the theory of Petri nets, NOAM integrates a hierarchical architectural framework with a generalized object model to create a coherent formal representation. The approach not only allows the precise specification of complex system properties and behavior but also supports incremental, hierarchical modeling and decomposition. An example is given to illustrate the applicability of the formalism.

E. A Formal Approach for Architectural Modeling and Prototyping of Distributed Real-Time Systems [DDE95]

The design of distributed real-time systems is one of the most difficult problems facing software engineering, because of the need to satisfy numerous complex requirements simultaneously. The research described below is part of an ongoing project to develop a suitable foundation for

an engineering methodology for building such systems. We have developed a formal model for representing, analyzing, and executing distributed real-time systems by synthesizing several well-studied approaches to design. Our goal has been to build on the strengths of these techniques while overcoming their weaknesses. Special focus is placed on using the formal model and our support environment to meet real-time constraints.

2.3 Deadlock Detection and Resolution in Distributed Systems

As a by-product of this project, we have developed two simple and efficient algorithms for detecting and resolving generalized deadlocks in distributed systems. These algorithms have a notably better performance and simpler structure than existing algorithms in the same class.

A. Efficient Algorithms for Detection and Resolution of Distributed Deadlocks [CD95]

We present a simple and efficient distributed algorithm for detecting generalized-deadlocks in distributed systems. Unlike previous algorithms, which are all based on the idea of distributed snapshots, and require multiple rounds of message transfers along the edges of the global wait-for graph (WFG), the proposed algorithm uses a novel approach that incrementally constructs an "image" of the WFG at an initiator node. The algorithm has a time complexity of $d + 1$ and a message complexity of $e + n$, where n is the number of nodes, d the diameter, and e the number of edges of the WFG. Compared with the best existing algorithm, our algorithm notably reduces both time and message complexities. Correctness proof and performance analysis for the algorithm are provided. In addition, the new approach simplifies deadlock resolution. An extension to the algorithm is presented to handle generalized-deadlock resolution with only a slight increase in the message complexity.

(An extended abstract has been published in the *Proceedings of the 7th IEEE Symposium on Parallel and Distributed Processing*, San Antonio, TX, October 25-28, 1995, 10–16.)

B. Deadlock Detection and Resolution in Distributed Systems Based on Locally Constructed Wait-For Graphs [CDA95]

A new approach for generalized deadlock detection and resolution in distributed systems is presented. First, a simple and efficient deadlock detection algorithm is presented, and its correctness is formally proved. Instead of taking a distributed snapshot of the global wait-for graph (WFG) as in

the existing algorithms, our algorithm incrementally constructs and reduces a WFG at an initiator process, which is then searched for deadlock. This new algorithm has two primary advantages: First, it has the simplicity and efficiency of a centralized algorithm, while remaining distributed in nature. Compared to the best message complexity, $4e - 2n + 2l$, of existing algorithms, our algorithm has an optimal worst case message complexity of $2n$, where n, e and l are the number of vertices, edges, and leaves in the WFG, respectively. The time complexity of our algorithm is also better than or equal to that of existing algorithms. Second, because the locally constructed WFG contains complete information about a detected distributed deadlock, it helps to simplify the task of deadlock resolution, and to reduce the number of processes that need to be aborted in order to break the deadlock. We present a simple extension to the deadlock detection algorithm to handle deadlock resolution in distributed systems with only a slight increase in message complexity. In addition, a simple modification to the deadlock detection algorithm is also presented to detect distributed AND-OR deadlocks.

(This paper has been submitted to the *IEEE Transactions on Software Engineering.* An extended abstract will appear in the *Proceedings of the 16th International Conference on Distributed Computing Systems*, Hong Kong, May 1996.)

3 Technology Development

3.1 Raddle Workbench

A Raddle workbench has been developed, which can be used to enter, edit and execute designs written in the Raddle language. This work was primarily undertaken by Andersen consulting under subcontract. The workbench provides a GUI environment, where Raddle designs are displayed graphically, and can be edited using a pointing device and keyboard. When a design is executed, the results are displayed graphically, so that interactions are highlighted when executing, and the flow of local control in each sequential process can be seen. The workbench is useful both for the prototyping of distributed systems, and also for experimenting with potential syntactic transformations. It serves as the platform for our design-via-transformations methodology. The Raddle workbench was developed in C++ and runs under the Solaris operating system.

For more details, the reader is referred to the following Raddle workbench documentation:

- **Raddle Workbench Architecture and User Interface Design Manual [AC94a]**
  Describes the design of the workbench in terms of functional modules, and provides a brief description of the functionality of each module. Also describes the graphical user interface of the workbench. The user interface can be used to either load a previous design, or to create a new design using a "drawing palette." Designs are represented graphically in a flowchart-like notation: sequential execution flows from left to right, branching represents choice-points in the execution, and a "loop" represents iteration. Designs can also be executed, with the execution being graphically displayed (e.g., program actions are "highlighted" as they are being executed).

- **Class Definitions and Implementation Issues [AC95a]**
  Describes the class structure and member functions for the most important classes used in the implementation of the workbench.

- **Interpreter Design [AC95b]**
  Discusses the algorithm used for interpreting a Raddle program, in particular, how actions are scheduled for execution, and how fair scheduling is implemented so that no enabled action is ignored forever.

- **Changes to the Syntax and Semantics of Raddle [AC94b]**
  Describes the modifications made to the version of Raddle described in [At87]. Most of these modifications result from the adoption of C++ as the underlying expression language (this choice of the underlying expression language was left open in [At87]).

3.2 The NOAM (RTG-Nets) Workbench

Based on the NOAM (RTG-Nets) architectural representation, we have developed a graphical, distributed workbench to support architectural specification, decomposition, simulation and prototyping of real-time distributed systems. As our research on the NOAM formalism has evolved, the design and implementation of the workbench has undergone several iterations of refinement.

In addition to its template-driven graphical interface for model building, the system supports graphics-oriented interactive prototyping. It allows a user to observe and control the behavior

of a NOAM specification through token movements on graphical NOAM object structures, and to interactively refine the design. The following is a summary of the main characteristics of the workbench. For more details, please refer to the cited documentation.

1. The graphics-oriented user interface provides a friendly environment for the user to use icons and pop-up templates to construct NOAM specifications. This significantly reduces the difficulty associated with formal model building. Furthermore, a similar interface is also provided for the runtime distributed simulators, using which a user can observe and monitor the execution of the architectural specification of a system modeled using NOAM.

2. The workbench is distributed on a set of workstations connected by an Ethernet LAN, which helps to improve the efficiency of complex system simulation and prototyping. A system configuration constructor is provided, which maps automatically a logical architectural NOAM specification to a specific networked system configuration as defined by the user. Consequently, there is no need to embed simulation specific information in a logical NOAM specification, and the same specification can be easily mapped to different simulation configurations

3. Object-oriented design and implementation are used throughout the development of the workbench. Every NOAM object in the specification corresponds to a graphic structure in the specification editor, which is stored as a composite object composed of place, transition, and arc classes. At runtime, a graphical simulator is created serving as the interface to the NOAM object. The encapsulated nature of NOAM objects supports multiparadigm specification and prototyping, where a system model can be described by a combination of the formal NOAM representation and other languages, e.g., C/C++. In addition, each simulator object can have its own methods, and therefore, multiple execution policies can be easily supported.

4. The simulation control mechanism of the workbench is transparent to the users, and independent of the overall structure of the workbench. Consequently, a change to the existing simulation control mechanism, which is determined by the semantics of NOAM, has little affect on the structure of the workbench, and vice versa. Therefore, a change made to the NOAM syntax and semantics will not significantly disrupt the structure, or require reimplementation of the system.

For more details, the reader is referred to the following NOAM workbench documentation:

- **NOAM Workbench Design Report [Lu95a]**
  Discusses the features and design issues of the workbench, overall system architecture, class structures for both off-line and runtime systems, and the control mechanism of the system.
- **NOAM Workbench Specification Editor Source Code [Lu95b]**
  Contains the documented source code for the offline system of the workbench.
- **NOAM Workbench Simulator Source Code [Lu95c]**
  Contains the documented source code for NOAM object simulators.
- **NOAM Workbench Simulation Engine Source Code [Lu95d]**
  Contains the documented source code for the simulation engine of the workbench.
- **NOAM Workbench User Guide [Lu95e]**

References

[ADM92] RFP F30602-92-R-0039, Andersen Consulting, "Knowledge-Based Software Assistant, Advanced Development Model, Vol ume III: Technical Proposal," July 1992.

[At87] P.C. Attie, "A Guide to Raddle87 Semantics," MCC technical report STP-340-87, Microelectronics and Computer Technology Corp, January 11, 1987.

[AC94a] Andersen Consulting, "Raddle Workbench Architecture and UI Design Manual," Technical Report, Andersen Consulting, 1994.

[AC94b] Andersen Consulting, "Changes to the Syntax and Semantics of Raddle," Technical Report, Andersen Consulting, 1994.

[AC95a] Andersen Consulting, "Class Definitions and Implementation Issues," Technical Report, Andersen Consulting, 1995.

[AC95b] Andersen Consulting, "Interpreter Design," Technical Report, Andersen Consulting, 1995.

[ADDE95] P.C. Attie, C. Das, Y. Deng, and M. Evangelist, "Automating the Refinement of Specifications via Syntactic Transformations," Technical Report, School of Computer Science, Florida International University, 1995.

[AD95] P.C. Attie and C. Das, "Construction of Concurrent Programs by Syntactic Merging," in preparation.

[ADEEK94] P.C. Attie, Y. Deng, M. Evangelist, A. Engberts, and V. Kozaczynski, "Intermediate Architectural Representations for the KBSA Advanced Development Model," Technical Report, School of Computer Science, Florida International University, 1994.

[CD95] S. Chen and Y. Deng, "Efficient Algorithms for Detection and Resolution of Distributed Deadlocks," Technical Report, School of Computer Science, Florida International University, 1995.

[CDA95] S. Chen, Y. Deng, and P. Attie, "Deadlock Detection and Resolution in Distributed Systems Based on Locally Constructed Wait-For Graphs," Technical Report, School of Computer Science, Florida International University, 1995. Extended abstract to appear in Proceedings of the *16th International Conference on Distributed Computing Systems*, Hong Kong, May 1996.

[DEN90a] Y. Deng and S. K. Chang, "A G-Net Model for Knowledge Representation and Reasoning," *IEEE Transactions on Data and Knowledge Engineering,* Vol. 2, No. 3, September 1990, 295-310.

[DCL94] Y. Deng, S. Chang, and X. Lin, "Executable Specification and Analysis for the Design of Concurrent Object-Oriented Systems," *International Journal of Software Engineering and Knowledge Engineering,* Vol. 4, No. 4: 427–450 (1994).

[DDE95] Y. Deng, W. Du, and M. Evangelist, "A Formal Approach for Architectural Modeling and Prototyping of Distributed Real-Time Systems," Technical Report, School of Computer Science, Florida International University, 1995.

[DDAE95] Y. Deng, W. Du, P. Attie, and M. Evangelist, "A Formalism for Architectural Modeling of Concurrent Real-Time Systems." Proceedings of the *the 8th International Conference on Software Engineering and Knowledge Engineering*, to appear.

[DL95] Y. Deng and X. Lin, "Use Formal Technique to Support the Design of Concurrent Object-Oriented Systems," Technical Report, School of Computer Science, Florida International University, 1995.

[GBCR83] C. Green, D. Luckham, R. Balzer, I. Cheatham, and C. Rich, "Report on a Knowledge Based Software Assistant," prepared for Rome Air Development Center, Griffiss AFB, New York 13441, June 15, 1983.

[Lu95a] S. Lu, "NOAM Workbench Design Report," Technical Report, School of Computer Science, Florida International University, 1995.

[Lu95b] S. Lu, "NOAM Workbench Specification Editor Source Code," Technical Report, School of Computer Science, Florida International University, 1995.

[Lu95c] S. Lu, "NOAM Workbench Simulator Source Code," Technical Report, School of Computer Science, Florida International University, 1995.

[Lu95d] S. Lu, " NOAM Workbench Simulation Engine Source Code," Technical Report, School of Computer Science, Florida International University, 1995.

[Lu95e] S. Lu, "NOAM Workbench User Guide," Technical Report, School of Computer Science, Florida International University, 1995.

[LD95] S. Lu and Y. Deng, "An Environment for the Specification, Simulation, and Analysis of Distributed Object Oriented Systems," Technical Report, School of Computer Science, Florida International University, 1995.

[LUC95a] D.C. Luckham, J.J. Kenney, L. Augustin, J. Vera, D. Bryan, and W. Mann, "Specification and Analysis of System Architecture Using Rapide," *IEEE Transactions on Software Engineering*, Vol. 21, No. 4, April 1995, 336-355.

[SEL94a] B. Selic, G. Gullekson and P. Ward, "Real-Time Object-Oriented Modeling," John Wiley & Sons, Inc., 1994.

☆U.S. GOVERNMENT PRINTING OFFICE: 1996-710-126-47033

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.