1989

Issues in Language Support for Rapid Prototyping

Luqi; Berzins, Valdis

Naval Postgraduate School

Luqi and V. Berzins, "Issues in Language Support for Rapid Prototyping'', Technical Report NPS 5 2-89- 026, Computer Science Department, Naval Postgraduate School, 1989.
https://hdl.handle.net/10945/65233

NPS52-89-026

NAVAL POSTGRADUATE SCHOOL

Monterey, California

ISSUES IN LANGUAGE SUPPORT FOR RAPID PROTOTYPING

LUQI
VALDIS BERZINS

MARCH 1989

Prepared for:

Naval Postgraduate School
Monterey, CA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. C. Austin
Superintendent

H. Shull
Provost

The work reported herein was supported by the National Science Foundation, the Office of Naval Research and the Naval Postgraduate School Research Council.

This report was prepared by:

[redacted]

LUQI
Assistant Professor
of Computer Science

Reviewed by:

[redacted]

ROBERT B. MCGHEE
Chairman
Department of Computer Science

Released by:

[redacted]

KNEALE T. MARSHALL
Dean of Information
and Policy Science

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPS52-89-026	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable) 52	7a. NAME OF MONITORING ORGANIZATION National Science Foundation & ONR Sponsored Navy Direct Funding
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b. ADDRESS (City, State, and ZIP Code) Washington, D. C. 20550
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Postgraduate School 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NSF CCR-8710737 O&MN, Direct Funding
8c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

ISSUES IN LANGUAGE SUPPORT FOR RAPID PROTOTYPING (U)

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Progress	FROM Sept 88 TO Mar 89	1989 March	15

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Darpa/ISTO is seeking to develop designs for a new language for rapid prototyping. The language is seen as part of longer subsequent efforts to develop a comprehensive prototyping system that will provide additional tools realizing a high-productivity software design and prototyping environment. This report presents the concepts of a prototyping language and relations to Expert Systems.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL LUQI	22b. TELEPHONE (Include Area Code) 408-646-2735 22c. OFFICE SYMBOL 52Lq

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-24.

UNCLASSIFIED

Issues in Language Support for Rapid Prototyping

Luqi
Valdis Berzins

Computer Science Department
Naval Postgraduate School
Monterey, CA 93953

1. Introduction

DARPA/ISTO has made an extremely important decision in developing designs for a rapid prototyping language. In this paper we discuss the important principles of language support for rapid prototyping, based on our experience. We have designed a prototyping language and carried out a feasibility study for its implementation over the past five years.

The purpose of the new Common Prototyping Language is to aid in the development of large Ada systems. The language is intended to support a comprehensive set of tools for computer-aided software design and prototyping. The goals for the language and tool set are:

(1) Rapid construction and adaptation of software,

(2) Enabling the development of more powerful systems,

(3) Checking if specified systems are acceptable to users,

(4) Checking internal consistency of proposed designs, and

(5) Ensuring that implementations conform to specifications.

The scope of the language is intended to include:

(1) parallel systems,

(2) distributed systems,

(3) real-time systems, and

(4) knowledge-based systems.

This paper contains a discussion of some of the basic issues involved in such a project.

2. Requirements for the Common Prototyping Language

To meet the goals of the project, the Common Prototyping Language should have the following properties:

(1) Simplicity. To make it easy to learn, understand, and process, the language must have a clear and simple structure and semantics. This implies uniform structure, a small number of orthogonal constructs, and general interpretations without special cases or restrictions. The language should support a user interface with graphical summary views and English paraphrasing for communication with untrained people. The language should have an abstract syntax for mechanical

processing.

(2) Expressiveness. To make it easy to use in describing systems, the language must be concise and clear. This implies support for abstractions, timing constraints, concurrency, synchronization, uniform communication, logical inference, incomplete descriptions, and automated design completion. The language should be at a specification and design level rather than at a programming level: the constructs of the language should correspond directly to decisions made by the designer, rather than to operations performed by the processor. This will make prototype descriptions self-documenting and easy to change.

(3) Formality. To support automated tools, the language must have an unambiguous and precisely defined meaning. The underlying model should have a mathematical basis to support execution, analysis, verification, and trusted transformations.

(4) Locality. To support system evolution and parallel execution, the language must have mechanisms for localizing design decisions in the description and localizing interactions between system components.

(5) Tracing. To support validation by users and system evolution, the language should support tracing design decisions to requirements.

(6) Specification. The language should include a facility for recording black-box specifications to document the intent of each component, support verification via proofs and automated testing, and to form queries for retrieving reusable components. The specifications should also form the basis for automated synthesis capabilities, inheritance of common properties and constraints, and consistency checking.

(7) Design. The language should include facilities for describing interconnections of available components, dependencies between components, and explanations of design justifications.

(8) Reuse. The language must support the description and retrieval of reusable software components. This implies facilities for adapting components to new uses and making small perturbation on their behavior without examining the details of the internal implementation of the components.

(9) Refinement. To support high productivity, the language should support the construction of efficient implementations by augmenting the prototype description with annotations describing lower level design decisions rather than requiring a complete re-formulation of the entire system description.

The rapid construction of software prototypes depends on simplifying the view of the system through which the specifiers and designers do their work, and providing automated means for bridging the gap between this simplified view and the detailed programming level description currently needed to make a software system efficiently executable. This automated support should include mechanisms for execution, preparation of input data, reporting and analyzing results, and diagnosing ill-formed descriptions and departures from desired behavior to allow the specifiers and designers to work entirely within the simplified view, at least during the construction of the initial prototype. This requires a consistent and simple semantic model rich enough to support all of these functions. Finding a suitable underlying model is the key to the project.

3. Modeling Issues

The models underlying the language provide the common ground for the associated set of tools. The semantic model for the language provides the basis for automated analysis, while the computational model provides the basis for execution. One of the main challenges in this project is to find a model that can coherently span the range of applications required.

There is no single commonly accepted model for representing real-time constraints. Some approaches that have been explored include temporal logic, state machines, mode charts, augmented data flow diagrams, Petri nets, and I/O automata. The model for the Common Prototyping Language should be chosen to enhance the application of recent results in logic, graph theory, and combinatorics to link the semantic model to an effective execution mechanism.

Other unsolved problems include effective models for real-time databases and real-time communications networks. In both of these areas, the problems of providing service within guaranteed worst-case time bounds are largely unexplored.

4. Language Issues

One tradeoff to be considered is the level of formality in the language. Informal techniques are generally easy to learn and use, but difficult to automate. Formal techniques support higher levels of automation, but are more difficult to learn and apply.

The language should allow the designer to specify attributes he cares about, but should not force the designer to specify attributes for all components. This implies automatically supplying reasonable default values for all attributes needed for execution.

5. Tool Issues

The connection between the Common Prototyping Language and Ada raises several issues that must be considered. Ordinary compiler technology is insufficient for execution of the prototyping language. Conventional translation techniques must be coupled with facilities for scheduling to meet real-time constraints and with transformations to allow the execution of incompletely specified processes.

Ada provides relatively weak guarantees about the scheduling of tasks, and limits programmer control over scheduling to statically specified priorities. Since this is somewhat removed from the level of support needed for implementing hard real-time systems, the execution support system for the prototyping language will have to provide higher level facilities for scheduling real-time operations. Such facilities can be classified as on-line (done at run-time) and off-line (done prior to execution). There is no universally accepted approach to real-time scheduling. Optimal scheduling algorithms are very time consuming, and generally cannot be carried out on-line, while off-line approaches are inflexible and do not handle overload situations very well. There are many different scheduling algorithms, and choosing the best one for a given application is a difficult problem.

Transformations are needed to execute incompletely specified components. Such transformations should supply reasonable default values for attributes necessary for execution if the designer does not explicitly specify them. Such attributes can be explicitly specified to produce a more accurate model of the system or to improve its performance.

One example of such attributes is the assignment of tasks to physical processors. Sometimes the assignment of particular critical tasks to particular processors is necessary to meet tight timing constraints by avoiding the overhead of some interprocessor communication. However, the designer usually does not care about the placement of all tasks, and would like the system to assign reasonable default locations to all of the tasks that do not have explicit processor assignments.

The tools should provide facilities for analyzing the consistency of a prototype design. Some of the checks that should be performed include:

(1) Type consistency.

(2) Feasibility of timing constraints.

(3) Consistency between the levels of a hierarchical description.

(4) Preconditions on input parameters and generic parameters.

(5) Constraints on relative rates of producer and consumer processes.

(6) Absence of deadlocks in distributed and parallel systems.

(7) Absence of unhandled exceptions.

In addition to providing facilities for constructing and checking the internal consistency of a prototype, the tool set should provide facilities for generating input data and evaluating the results of prototype execution at the in terms of the same semantic model used for the design of the prototype.

The tool set must also provide a design database for maintaining the design history in terms of a set of versions of the system and the alternative designs that were considered. This database should also be capable of recording and maintaining constraints on the system. A related issue that should be considered in the design of the language is the relation between the language, which is used for describing the design objects in the database, and the notations for describing the attributes, relationships, and constraints among those objects that are used by the tools in the associated environment.

6. Knowledge Base Issues

The supporting environment for the language should provide knowledge base support for the following:

(1) Managing reusable components. The environment should contain a large software base with reusable components. This software base should be coupled with a set of rules for tailoring and combining available components to fulfill queries that do not exactly match any of the components explicitly stored in the software base.

(2) High level debugging. Errors and failures during prototype execution should be mapped from the programming language level to level of the prototyping language, to allow the designer to work entirely in terms of the semantic model associated with the prototyping language.

(3) Optimization. The transformations for optimizing a prototype version of a system to produce a production version should be performed with minimum interaction with the designer. This implies keeping track of the decisions made by the designer in optimizing previous versions of the system, determining which of

those decisions are still valid for later versions, and automatically applying the ones that are found to be still valid.

(4) Explanations. Justifications for decisions made automatically should be available to provide feedback to the designer in cases where automated design completion procedures fail. This requires an expert system with a substantial knowledge base.

7. Conclusions

The Common Prototyping Language project has an ambitious set of goals that raises many interesting research problems. The language is a key component of a larger project for creating a comprehensive tool set for prototyping because it must tie everything together. Solutions to these problems are essential for achieving significant improvements in the quality and productivity of the software development process.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Office of Naval Research Office of the Chief of Naval Research Attn. CDR Michael Gehl, Code 1224 800 N. Quincy Street Arlington, Virginia 22217-5000	1
4.	Space and Naval Warfare Systems Command Attn. Dr. Knudsen, Code PD 50 Washington, D.C. 20363-5100	1
5.	Ada Joint Program Office OUSDRE(R&AT) Pentagon Washington, D.C. 20301	1
6.	Naval Sea Systems Command Attn. CAPT Joel Crandall National Center #2, Suite 7N06 Washington, D.C. 22202	1
7.	Office of the Secretary of Defense Attn. CDR Barber STARS Program Office Washington, D.C. 20301	1
8.	Office of the Secretary of Defense Attn. Mr. Joel Trimble STARS Program Office Washington, D.C. 20301	1
9.	Commanding Officer Naval Research Laboratory Code 5150 Attn. Dr. Elizabeth Wald Washington, D.C. 20375-5000	1

10. Navy Ocean System Center 1
Attn. Linwood Sutton, Code 423
San Diego, California 92152-500

11. National Science Foundation 1
Attn. Dr. William Wulf
Washington, D.C. 20550

12. National Science Foundation 1
Division of Computer and Computation Research
Attn. Dr. Peter Freeman
Washington, D.C. 20550

13. National Science Foundation 1
Director, PYI Program
Attn. Dr. C. Tan
Washington, D.C. 20550

14. Office of Naval Research 1
Computer Science Division, Code 1133
Attn. Dr. Van Tilborg
800 N. Quincy Street
Arlington, Virginia 22217-5000

15. Office of Naval Research 1
Applied Mathematics and Computer Science, Code 1211
Attn: Dr. James Smith
800 N. Quincy Street
Arlington, Virginia 22217-5000

16. New Jersey Institute of Technology 1
Computer Science Department
Attn. Dr. Peter Ng
Newark, New Jersey 07102

17. Southern Methodist University 1
Computer Science Department
Attn. Dr. Murat Tanik
Dallas, Texas 75275

18. Editor-in-Chief, IEEE Software 1
Attn. Dr. Ted Lewis
Oregon State University
Computer Science Department
Corvallis, Oregon 97331

19. University of Texas at Austin 1
Computer Science Department
Attn. Dr. Al Mok
Austin, Texas 78712

20. University of Maryland 1
College of Business Management
Tydings Hall, Room 0137
Attn. Dr. Alan Hevner
College Park, Maryland 20742

21. University of California at Berkeley 1
Department of Electrical Engineering and Computer Science
Computer Science Division
Attn. Dr. C.V. Ramamoorthy
Berkeley, California 94720

22. University of California at Los Angeles 1
School of Engineering and Applied Science
Computer Science Department
Attn. Dr. Daniel Berry
Los Angeles, California 90024

23. University of Maryland 1
Computer Science Department
Attn. Dr. Y. H. Chu
College Park, Maryland 20742

24. University of Maryland 1
Computer Science Department
Attn. Dr. N. Roussapoulos
College Park, Maryland 20742

25. Kestrel Institute 1
Attn. Dr. C. Green
1801 Page Mill Road
Palo Alto, California 94304

26. Massachusetts Institute of Technology 1
Department of Electrical Engineering and Computer Science
545 Tech Square
Attn. Dr. B. Liskov
Cambridge, Massachusetts 02139

27. Massachusetts Institute of Technology 1
Department of Electrical Engineering and Computer Science
545 Tech Square
Attn. Dr. J. Guttag
Cambridge, Massachusetts 02139

28. University of Minnesota 1
Computer Science Department
136 Lind Hall
207 Church Street SE
Attn. Dr. J. Ben Rosen
Minneapolis, Minnesota 55455

29. International Software Systems Inc. 1
12710 Research Boulevard, Suite 301
Attn. Dr. R. T. Yeh
Austin, Texas 78759

30. Software Group, MCC 1
9430 Research Boulevard
Attn. Dr. L. Belady
Austin, Texas 78759

31. Carnegie Mellon University 1
Software Engineering Institute
Department of Computer Science
Attn. Dr. Lui Sha
Pittsburgh, Pennsylvania 15260

32. IBM T. J. Watson Research Center 1
Attn. Dr. A. Stoyenko
P.O. Box 704
Yorktown Heights, New York 10598

33. The Ohio State University 1
Department of Computer and Information Science
Attn. Dr. Ming Liu
2036 Neil Ave Mall
Columbus, Ohio 43210-1277

34. University of Illinois 1
Department of Computer Science
Attn. Dr. Jane W. S. Liu
Urbana Champaign, Illinois 61801

35. University of Massachusetts 1
Department of Computer and Information Science
Attn. Dr. John A. Stankovic
Amherst, Massachusetts 01003

36. University of Pittsburgh 1
Department of Computer Science
Attn. Dr. Alfs Berztiss
Pittsburgh, Pennsylvania 15260

37. Defense Advanced Research Projects Agency (DARPA) 1
Integrated Strategic Technology Office (ISTO)
Attn. Dr. Jacob Schwartz
1400 Wilson Boulevard
Arlington, Virginia 22209-2308

38. Defense Advanced Research Projects Agency (DARPA) 1
Integrated Strategic Technology Office (ISTO)
Attn. Dr. Squires
1400 Wilson Boulevard
Arlington, Virginia 22209-2308

39. Defense Advanced Research Projects Agency (DARPA) 1
Integrated Strategic Technology Office (ISTO)
Attn. MAJ Mark Pullen, USAF
1400 Wilson Boulevard
Arlington, Virginia 22209-2308

40. Defense Advanced Research Projects Agency (DARPA) 1
Director, Naval Technology Office
1400 Wilson Boulevard
Arlington, Virginia 2209-2308

41. Defense Advanced Research Projects Agency (DARPA) 1
Director, Strategic Technology Office
1400 Wilson Boulevard
Arlington, Virginia 2209-2308

42. Defense Advanced Research Projects Agency (DARPA) 1
Director, Prototype Projects Office
1400 Wilson Boulevard
Arlington, Virginia 2209-2308

43. Defense Advanced Research Projects Agency (DARPA) 1
Director, Tactical Technology Office
1400 Wilson Boulevard
Arlington, Virginia 2209-2308

44. MCC AI Laboratory 1
Attn. Dr. Michael Gray
3500 West Balcones Center Drive
Austin, Texas 78759

45. COL C. Cox, USAF 1
JCS (J-8)
Nuclear Force Analysis Division
Pentagon
Washington, D.C. 20318-8000

46. LTCOL Kirk Lewis, USA 1
JCS (J-8)
Nuclear Force Analysis Division
Pentagon
Washington, D.C. 20318-8000

47. University of California at San Diego 1
Department of Computer Science
Attn. Dr. William Howden
La Jolla, California 92093

48. University of California at Irvine 1
Department of Computer and Information Science
Attn. Dr. Nancy Levenson
Irvine, California 92717

49. University of California at Irvine 1
Department of Computer and Information Science
Attn. Dr. L. Osterweil
Irvine, California 92717

50. University of Colorado at Boulder 1
Department of Computer Science
Attn. Dr. Lloyd Fosdick
Boulder, Colorado 80309-0430

51. Santa Clara University 1
Department of Electrical Engineering and Computer Science
Attn. Dr. M. Ketabchi
Santa Clara, California 95053

52. Oregon Graduate Center 1
Portland (Beaverton)
Attn. Dr. R. Kieburtz
Portland, Oregon 97005

53. Dr. Wolfgang Halang 1
Bayer AG
Ingenieurbereich Progessleittechnik
D-4047
Dormagen, West Germany

54. Dr. Bernd Kraemer 1
GMD Postfach 1240
Schloss Birlinghaven
D-5205
Sankt Augustin 1, West Germany

55. Dr. Aimram Yuhudai 1
Tel Aviv University
School of Mathematical Sciences
Department of Computer Science
Tel Aviv, Israel 69978

56. Dr. Robert M. Balzer
USC-Information Sciences Institute
4676 Admiralty Way
Suite 1001
Marina del Ray, California 90292-6695 — 1

57. U.S. Air Force Systems Command
Rome Air Development Center
RADC/COE
Attn. Mr. Samuel A. DiNitto, Jr.
Griffis Air Force Base, New York 13441-5700 — 1

58. U.S. Air Force Systems Command
Rome Air Development Center
RADC/COE
Attn. Mr. William E. Rzepka
Griffis Air Force Base, New York 13441-5700 — 1

59 LuQi
Code 52Lq
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5100 — 50

60. Research Administration
Code: 012
Naval Postgraduate School
Monterey, CA. 93943 — 1