1989

Software Analysis and Testing through Prototyping

Luqi; Kraemer, B.; Berzins, V.

Naval Postgraduate School

Luqi, B. Kraemer, and V. Berzins, "Software Analysis and Testing through Prototyping'', Technical Report NPS 52-89-044, Computer Science Department, Naval Postgraduate School, 1989.
https://hdl.handle.net/10945/65282

NPS52-89-044

NAVAL POSTGRADUATE SCHOOL

Monterey, California

SOFTWARE ANALYSIS AND TESTING THROUGH PROTOTYPING

LUQI
BERND KRAEMER
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

This report was prepared in conjunction with research conducted for the National Science Foundation and funded by the Naval Postgraduate School.

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) NPS52-89-044	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 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)

SOFTWARE ANALYSIS AND TESTING THROUGH PROTOTYPING (U)

12. PERSONAL AUTHOR(S)
LUQI, BERND KRAEMER, VALDIS BERZINS

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	19

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Computer Aided Software Engineering, Rapid Prototyping, Specification, real-time Software, Embedded Systems, Software Design, Reusability

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
Prototyping is such a complementary approach, which allows many of the traditional kinds of software analysis and testing to be applied at earlier stages. The prototyping process helps to establish relatively static concepts of correctness, which can be used as a meaningful basis for later verification efforts. The execution of software prototypes is similar to traditional validation, except that the developer is explicitly concerned with the length of time during which the proposed system will continue to meet customer needs, rather than just ensuring the system will meet currently perceived needs. This paper also discusses three levels of analysis and testing that are important for real-time systems in rapid prototyping.

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

Software Analysis and Testing through Prototyping

Luqi
B. Kraemer
V. Berzins

Computer Science Department
Naval Postgraduate School
Monterey, CA 93943

ABSTRACT

Prototyping is such a complementary approach, which allows many of the traditional kinds of software analysis and testing to be applied at earlier stages. The prototyping process helps to establish relatively static concepts of correctness, which can be used as a meaningful basis for later verification efforts. The execution of software prototypes is similar to traditional validation, except that the developer is explicitly concerned with the length of time during which the proposed system will continue to meet customer needs, rather than just ensuring the system will meet currently perceived needs. This paper also discusses three levels of analysis and testing that are important for real-time systems in rapid prototyping.

1. Introduction

Robustness, reliability, and correctness of operation are quality aspects of software products that gain increasing importance. This is particularly true for critical systems whose malfunction may result in loss of human life, compromise of national security, or massive loss of property [3]. Software components of hard real-time systems often are among this class of critical system as they typically control production processes, transport systems, communication systems, chemical or power plants, etc.

The techniques for certifying such properties range from formal program *verification* and software *testing* to formal and informal *analysis* techniques [2, 5] such as data flow analysis, loop invariant detection, deadlock detection, software inspection, documentation evaluation, and walkthroughs. Overview information and references to verification can be found in *Software Engineering Notes*, August 1985, information about current approaches to testing are surveyed in [4], and other validation techniques are reported in [1] and *IEEE Software*, May 1989. All of these techniques have their specific merits but also show limitations which require using a combination of validation and verification techniques for building quality software.

Testing has a long tradition in programming and software engineering. The traditional approach to software development, however, suffers from the fact that the results of testing operational code become available close to the end of the development process, so that design errors detected during system testing require an immense redesign and reimplementation effort, and are likely to cause project delays if they occur. Thus an effective quality assurance strategy should combine testing with other approaches that can detect requirements and design errors earlier in the cycle, when they are less expensive to correct and have less external impact on the project. Prototyping is such a complementary approach, which allows many of the the traditional kinds of software testing to be applied at earlier stages.

2. The Role of Prototyping in System Validation

The faults in software systems with the largest impact are requirements and specification errors, since such errors tend to affect large portions of the system and can be very expensive to correct. Requirements are often uncertain at the early stages, because the customers do not have a complete understanding of their problems or how proposed software systems will affect their daily operations and their understanding of the application domain. Experience with a software system usually changes the customers' perceptions of their problems, and opens up new possibilities which lead to new requirements. This makes the customer's problem a *moving target*. Educating customers and developers about the problem is as much a part of the process as building a system, so that the traditional concepts of what constitutes an error do not quite match the reality of the early parts of the development process: a system that is "correct" at one point in time may become "incorrect" later without any changes in system behavior.

The purpose of the iterative prototyping process illustrated below is to help stabilize the effects of a proposed system on the customer's perceptions and requirements before the system is constructed on a full scale.

```
+--------------+                        +-----------+
| determine    |     requirements       | construct |
| requirements |----------------------->| prototype |
+--------------+                        +-----------+
      ^                                       |
      | requirements adjustment               | prototype
      |                                       |
+--------------+                              v +---------------+
| demonstrate  |<-----------------------------+->| system        |
| prototype    |-------------------------------->| implementation|
+--------------+        requirements OK          +---------------+
```

This process helps to establish relatively static concepts of correctness, which can be used as a meaningful basis for later verification efforts. The goal of prototyping is similar to traditional validation, except that the developer is explicitly concerned with the length of time during which the proposed system will continue to meet customer needs, rather than just ensuring the system will meet currently perceived needs. Automated tools are necessary to carry out this process with reasonable speed and cost [1, 6].

3. Multiple Levels of Analysis and Testing

There are at least three levels of analysis and testing that are important for real-time systems at the prototyping stage:

(1) Checking whether proposed timing requirements are sufficient to meet the higher level functional requirements that motivate the timing requirements. Common examples of such functional requirements include ensuring that software estimates of the state of a real world system are maintained to a given accuracy, or that the state of the real-world system is controlled to remain within some desirable region. A concrete example is an aircraft control system, whose purpose is to prevent mid-air collisions. Testing is essential for this part of the problem, because it involves the relationship between a formally described abstract object (the software prototype) and an informally described concrete object (the physical system to be controlled). Analysis of formal models of the software and interacting physical systems should be coupled with testing to check the correspondence between the formal models and the real world.

(2) Checking whether a proposed design meets its requirements, given that the individual components meet their specifications. Most large software systems are designed using modular decompositions. The essential question at the design stage is: will a proposed design work correctly if the implementations of the specified subcomponents are carried out with perfect accuracy? A specification-based prototyping approach can help answer this question before much effort has been spent on the detailed implementation of the components. This part of the problem is subject to formal verification techniques, which are easier than prov-

ing correctness of low level code, because only the correspondence between two sets of specifications in the same language is at issue.

(3) Checking whether a component meets its specifications and timing constraints. This process can be addressed at the prototyping stage by testing and instrumentation that monitors the behavior of an executing prototype with respect to its specifications. This part of the problem has both symbolic and testing aspects, particularly with respect to the real-time behavior of the proposed implementation, which again depends to some extent on the physical properties of the hardware systems involved in the implementation. To establish some confidence that a proposed system will provide guaranteed service within a deadline, the interactions between the software with users, hardware, and other physical components must be tested.

4. How It Can Be Done

We propose a rapid prototyping approach comprising a language, called RPL, [8] and an integrated tool set supporting iterative prototyping of complex software systems [6, 7, 10]. RPL covers a wide range of applications, including real-time, parallel, distributed, and knowledge-based systems. It combines second-order logic specifications supporting verification with an augmented dataflow representation for design and interconnection of prototype components. The design graph is augmented with special pre/post conditions to express real-time constraints and adjust component behavior to each application context [9]. Execution is based on automatically generating code which links reusable software components or simulates component behavior via an executable subset of the specification logic. Real-time constraints are guaranteed by automatically constructed schedules. Iterative modifications of prototypes are supported by localized information in RPL and its computational model, component behavior modification via logical constraints, and facilities of the tool set for code and design reuse, requirements tracing, and static analysis. The logic and the proposed computational model provide the basis for integrating these facilities into a coherent language and tool structure. Among others the tool set will support execution and dynamic debugging, optimization and transformation to final implementation, as well as formal analysis and proofs of correctness.

The prototyping approach allows requirements and desirable features of the intended system to be clarified while the system is incrementally implemented by mapping designs to reusable and executable software components. Design alternatives can be evaluated by observing the behavior of prototypes under real-time conditions. Test data generation is simplified due to the the separation of concerns emphasized by RPL and its formal semantics. Predicted performance can be verified by executing the prototype under real-time conditions reflecting best and worst case assumptions. In particular, static analysis can be combined with testing to verify the assumptions of the timing properties of the software components on which the design is based, with special attention to the paths with the longest expected execution times. This can lead to greater confidence by decoupling the empirical estimation of the execution times for individual machine instructions from the static analysis which determines the sequence of instructions along the longest execution path.

5. Conclusion

The interaction between testing and prototyping should be explored from several points of view. Since prototypes are embedded in a computer-aided prototyping system for execution, they provide a greater degree of flexibility, observability, and control than a production implementation, enabling new testing techniques that check some of the critical decisions made in the early stages of software development, and provide a means for coupling testing with simplified formal analysis with respect to high level specifications. As in Monte Carlo simulations, the use of partial formal analysis to reduce the variability of the unknown aspects of the problem can lead to more accurate conclusions based on fewer test cases. Demonstrations to customers also provide a means for using testing techniques to do requirements validation, and provide error detection and location earlier in the development process, when it can have a much larger beneficial effect. These possibilities open up a new and important area for future research and development.

1. V. Berzins and Luqi, *Software Engineering with Abstractions: An Integrated Approach to Software Development using Ada*, Addison-Wesley, 1989.

2. M. Christ-Neumann, B. Kraemer, H. H. Nieters and H. W. Schmidt, "The GRASPIN Environment on the Lisp Machine - User's Guide", GRASPIN Technical Paper GMD37/1, GMD, Sankt Augustin, Mar 1989.

3. J. Goguen, "OBJ as a Theorem Prover with Applications to Hardware Verification", SRI-CSL-88-4R2, SRI International, Menlo Park, California, August 1988 .

4. W. Howden, *Functional Program Testing and Analysis*, McGraw-Hill, New York, 1987.

5. B. Kraemer, *Concepts, Syntax and Semantics of SEGRAS - A Specification Language for Distributed Systems*, Oldenbourg Verlag,, Muenchen-Wien, 1989.

6. Luqi, "Rapid Prototyping for Large Software System Design", Ph. D. Thesis, University of Minnesota, 1986.

7. Luqi and V. Berzins, "Rapidly Prototyping Real-Time Systems", *IEEE Software*, Sep. 1988, 25-36.

8. Luqi, V. Berzins, B. Kraemer and L. White, "A Proposed Design for a Rapid Prototyping Language", NPS52-89-45, Computer Science Department, Naval Postgraduate School, 1989.

9. Luqi, "Handling Timing Constraints in Rapid Prototyping", in *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, Jan. 1989, 417-424.

10. W. Swartout and R. Balzer, "On The Inevitable Intertwining of Specification and Implementation", *Comm. of the ACM 25*, 7 (July 1982), 438-440.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, Virginia 22304-6145

2. Library, Code 0142 — 2
   Naval Postgraduate School
   Monterey, California 93943-5002

3. Office of Naval Research — 1
   Office of the Chief of Naval Research
   Attn. CDR Michael Gehl, Code 1224
   800 N. Quincy Street
   Arlington, Virginia 22217-5000

4. Space and Naval Warfare Systems Command — 1
   Attn. Dr. Knudsen, Code PD 50
   Washington, D.C. 20363-5100

5. Ada Joint Program Office — 1
   OUSDRE(R&AT)
   Pentagon
   Washington, D.C. 20301

6. Naval Sea Systems Command — 1
   Attn. CAPT Joel Crandall
   National Center #2, Suite 7N06
   Washington, D.C. 22202

7. Office of the Secretary of Defense — 1
   Attn. CDR Barber
   STARS Program Office
   Washington, D.C. 20301

8. Office of the Secretary of Defense — 1
   Attn. Mr. Joel Trimble
   STARS Program Office
   Washington, D.C. 20301

9. Commanding Officer — 1
   Naval Research Laboratory
   Code 5150
   Attn. Dr. Elizabeth Wald
   Washington, D.C. 20375-5000

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. Tom Keenan
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. Fox
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. University of Maryland 1
Attn. Dr. Basili
Computer Science Department
College Park, MD 20742

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 1
USC-Information Sciences Institute
4676 Admiralty Way
Suite 1001
Marina del Ray, California 90292-6695

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

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

59 **LuQi** 100
Code 52Lq
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5100

60 Steve Huseth 1
Honeywell Systems & Research Center
3660 Technology Dr
Mels, MN 55418

61 Research Administration 1
Code: 012
Naval Postgraduate School
Monterey, CA 93940