4

AD-A278 161

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE 06/93	3. REPORT TYPE AND DATES COVERED Final 02/01/89-3/31/93

TITLE AND SUBTITLE
Fautless Software Project

5. FUNDING NUMBERS
G
N0014-89-J1751

AUTHOR(S)
William Howden

PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Regents of the University of California
University of California, San Diego
Dept. of Computer Science & Engineering
9500 Gilman Dr., La Jolla, CA 92093-0114

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
James G. Smith, Code 1211 --Scientific Officer
800 North Quincy Street
Arlington, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
(b)

11. SUPPLEMENTARY NOTES

DTIC
ELECTE
APR 11 1994
S G D

12a. DISTRIBUTION/AVAILABILITY STATEMENT
No Limitations

10pg

94-10875

13. ABSTRACT (*Maximum 200 words*)

Research has focused in three areas. The first is the development of a general paradigm for informal program verification and understanding. The second is the development of practical methods for the reconstruction of functional specifications from code. The third is the development of formal, statistical models for the evaluation of testing methods.

[illegible]

14. SUBJECT TERMS
real programs and systems, modularization, selectivity, abstraction, QDA, Ada informal verification system, faultless software

15. NUMBER OF PAGES
9

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	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

94 4 8 093

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723
Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

Numerical productivity measures

Refereed papers submitted but not yet published	2
Refereed papers published	3
Unrefereed papers and articles	2
Books or parts thereof submitted but not yet published	0
Books or parts thereof published	0
Patents filed but not yet granted	0
Patents granted	0
Invited presentations	2
Contributed presentations	2
Honors received (program committee ISSTA)	1
Prizes or rewards received	0
Promotions obtained	0
Graduate students supported	3

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723
Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

Summary of technical results

Research has focused in three areas. The first is the development of a general paradigm for informal program verification and understanding. The second is the development of practical methods for the reconstruction of functional specifications from code. The third is the development of formal, statistical models for the evaluation of testing methods.

Informal analysis paradigms

In the first area of research we have identified three dimensions along which it is possible to simplify program analysis, and make it applicable to real programs and systems. They are modularization, selectivity and abstraction. The first is the familiar technique of "divide and conquer" in which analysis is carried out on a module-by-module basis, and will not be discussed here.

Selectivity refers to the restriction of formal analysis methods to those properties of a program, or steps in analysis, that are not obvious. Selectivity can result in informal analysis if steps or properties are accepted as valid "by inspection". In the extreme case, as in code reading and inspection, the entire process is informal. What is needed is a paradigm in which the obvious parts are informal, and more formal analysis is given to the less obvious parts.

Abstraction refers to the replacement of a detailed description of some property of a program with a more abstract one in which the abstraction stands for the more detailed property. In the formal approach, such abstractions would have some exogenous formal definition. In an informal approach, undefined abstractions, in the form of undefined predicates and relationships, are used. When these abstractions are associated with code, they can be thought of as being informally defined by the code itself.

An informal approach to analysis has been developed in which selectivity is implemented through the documentation of "working hypotheses". Abstraction is implemented through the use of different kinds of abstract object properties and the use of "abstraction rules". The latter describe relationships between objects and properties at different levels of abstraction.

When informal analysis is used, an obvious question to consider concerns the circumstances in which it may be invalid, resulting in an unsound proof of some property. We have distinguished between two possibilities. One is that in which an abstract, informal representation is internally inconsistent. In this case the abstraction is said to be invalid. Steps can be taken to ensure that an informal abstraction is valid. The QDA analysis system for the AV-8B avionics is used as an example in which abstractions are prevented from being invalid through restrictions on the language for creating the abstraction. The other way in which an unsoundness problem can occur is that in which an abstraction is valid, but that there were errors in the abstraction creation process. We

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723

Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

identify features of an approach called "fail-safe" documentation that will avoid these problems. In this approach, an analysis is carried out which may produce "false negatives" but avoids "false positives", i.e. it may falsely report that some property is invalid.

These results provide a good basis for our continued theoretical work on informal analysis as well as for continued development of informal analysis technology.

Reengineering of functional specifications

The main idea in our approach to functional specifications (f-specs) generation is the use of symbolic evaluation. It can be used to generate f-specs for program paths that consist of a condition part that describes the input constraints for a path, and a function part that describes the computations carried out on that path. An f-spec for an individual path is called an f-spec component. The collection of components for a program is called its program f-spec. Symbolic evaluation is an idea that has periodically reappeared since its first appearance as a test data generation method in (W.E. Howden, Methodology for the generation of program test data, *IEEE Transactions on Computers*, 1975). The method is appealing, but practical considerations have prevented it from being of general use.

We have adopted a generalized approach to symbolic evaluation in which we generate f-specs with respect to "operations of interest". These are functions or procedures that can occur at any point in a program. Symbolic evaluation generates a description of the context in which the operation occurs, and the symbolic values of its input parameters.

We have studied a set of programs in a full programming language to see if the practicality problems of symbolic evaluation could be solved. Our results have been positive, and we have developed a set of techniques that can be used to generate useful symbolic evaluation functional specifications. The methods are: projection, simplification, suppression, and modularization.

Projection is closely related to program slicing. In this approach we identify groups of variables (or output data operations) that have the property that if we develop a functional specification for them separately there will be a significant decrease in the total number of functional specification components. We identify patterns in which projection will have a large positive payoff and those in which the payoff could be negative. Heuristics are suggested for choosing sub-optimal, but minimizing projections.

Simplification can take different forms. One kind of simplification is ordinary expression simplification in which an f-spec would be generated and then re-write rules used to simplify awkward or redundant expressions. Another approach that we have investigated is "structural simplification". In this approach we look for certain common structural patterns that allow us to simplify functional specifications before they are generated, rather than after.

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723
Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

Infeasible path detection can also be considered a simplification method. We incorporate methods in our approach to prevent the generation of f-spec components having inconsistent condition parts.

We consider the use of "clichés" in program reengineering to be a simplification method. Cliché oriented methods attempt to simplify by recognizing common programming design fragments such as "update a variable" or "sort a file". In some cases these ideas correspond to what we have called structural simplifications. In other cases recognition is more complicated. We recognize the difference between intenstional and extensional patterns. The former, like structural simplification patterns, are well defined. The latter are known primarily through example, and attempts to build algorithmic recognition procedures have had modest results. Our approach is to generate simple, readable, functional specifications that make the recognition of extensional patterns (by the programmer/user) as easy as possible.

Projection involves looking at a program from different points of view. Simplification involves methods to produce simpler, but equivalent, representations. Suppression involves eliminating information from a representation with the goal of producing a clearer picture of the remaining information. Several suppression methods have been identified. One is index suppression. Suppose that a program contains a sequence of program substructures that contain writes to a sequential file. We will not be able to generate independent separate f-specs for the writes, because they are related through their manipulation and use of the common file pointer. If the use of this pointer is suppressed in the generation of the f-specs, then separate f-specs can be generated for each write, showing the conditions under which it is executed independently of all the other writes.

Other forms of suppression that have been developed include format suppression, data item suppression and data and operator discrimination suppression. All are useful in generating abstract specifications in which some aspect of a program's details are eliminated.

Modularization is critical to the control of complexity in an f-spec. One approach that has been suggested is to try to recognize "functional patterns" in code, and to identify summarizing abstract functions that can then be used in place of that code, resulting in simpler f-specs. Our approach has been, instead, to figure out how to used modularizing abstractions already present in the code, i.e. a program's procedures and functions. The idea is to identify those procedures that occur in the call chain to some "operation of interest", such as a write operation, and then to also treat them as operations of interest in generating higher level specs. The higher level f-specs show the conditions under, and data with which, the operations are called. The approach has been successful, although it required the development of additional methods for dealing with special modularization problems.

One of the major problems in the use of symbolic evaluation has been to find some way of generating specifications for programs that involve loops. We experimented with several different possible methods, including the generation of closed forms for simple

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723
Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

cases, and came to the following conclusion. In the case of well structured programs where loops are encapsulated in a programming construct, the best way is to treat loops as procedures that are called in the context for the loop, and to use our modularization procedures for representing the code in which the loop occurs. A modified form of our specifications generation procedure is used to describe the behavior of the code in the loop.

In the case of poorly structured loops we have developed a method in which an abstract procedure or function is generated that stands for the computations performed between the first and last iterations of the loop. This abstract procedure is then used, during symbolic evaluation, to describe the effects of the loop.

One of the difficulties in carrying out research on symbolic evaluation has been the need to develop a set of methods that can work together in an integrated fashion. Our approach has been very empirical, involving the generation of dozens of examples of applications of different ideas to real programs. It is only recently that we have formulated the approach that is described here, and for which the results indicate that the project will be successful. We are now preparing a report describing the above methods in more detail, which will contain actual examples of the successful use of the methods.

Statistical models for evaluation of testing methods

A variety of different methods for testing have been proposed but there has been limited success in discovering a formal basis for comparing their effectiveness. One approach involves the use of variations on failure rate models to try to compare random and partition oriented testing.

We re-examined the failure rate model and showed that it does give good results in characterizing certain kinds of situations in which one would expect a partition oriented strategy to be an improvement over simple random testing. We then distinguished between what we call statistical and deterministic random testing. This distinction makes it possible to prove that certain kinds of idealized testing patterns correspond to situations in which one of random or partition testing would be superior to the other. We also formalized several other results in the area.

Previous work involved the development of alternative statistical models. We developed a fault oriented model in which the "testability" of a piece of software rather than the failure rate properties of its domain is used as the basis for analysis. This model makes it possible to take into account testing situations in which the programmer is assumed to be able to accurately choose data that will discover faults in certain kinds of fault classes. We developed the fault rate model, and used it to characterize situations in which it is effective to use a testing strategy that involves the decomposition of a program. The concept of "decomposition" is general in our model, and can be both functional or logical. A functional decomposition, for example, might isolate normal from boundary cases. A logical decomposition might isolate the character string processing aspects of a program. The basic idea in the testability approach is to use conditional probability models for

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723

Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

predicting test effectiveness. This research is at an intermediate stage, and current activities involve the application of the testability model to the evaluation of a variety of actual testing methods.

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723
Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

Publications, presentations and reports

Pefereed conference proceedings

W.E. Howden and Sue Pak, Problem Domain, Structural and Logical Abstractions in Reverse Engineering, *Proceedings, Conference on Software Maintenance*, IEEE, November, 1992, Orlando Florida.

W.E. Howden and Cheron Vail, An informal verification of a critical system, *Proceedings, Fifth International Conference on Software Engineering its Applications*, December, 1992, Toulouse, France.

W.E. Howden, Foundational issues in software testing and analysis, *Proceedings, Fifth International Conference on Software Quality*, June, 1993.

Presentations

W.E. Howden, An informal approach to program verification and understanding, Laboratoire d'Automaticque et d'Analyse des Systems, January, 1993, Toulouse, France.

Submitted papers

W.E. Howden and Bruce Wieand, A method for informal program analysis and its application to a critical system, UCSD T.R., Computer Science and Engineering, September, 1993.

W.E. Howden and Y. Huang, Analysis of testing methods using failure rate and testability models, UCSD T.R. Computer Science and Engineering, June, 1993.

Technical Reports

W.E. Howden and G.M. Shi, Temporal event analysis and program understanding, UCSD T.R., Computer Science and Engineering, April 1993.

W.E. Howden and Sue Pak, Structural abstraction methods for software reengineering, UCSD T.R., Computer Science and Engineering, March, 1993.

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723
Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

Research transitions and DOD interactions

A tool implementing the QDA methodology that was developed for the Naval Air Warfare Center, and the results of its application to the AV-8B avionics, provided the empirical data upon which the development of a paradigm and general methodology for informal program analysis was based.

Principal investigator: William E. Howden
Institution: University of California at San Diego
Phone: 619 534 2723
Contract title: Faultless software project
Contract number: N0014-89-J1751
Reporting period: 1 Oct 92 - 30 Sep 93

Software and hardware prototypes

The paradigm and general methodology of the QDA approach is being used as the basis for the development of an Ada informal verification system.