Final Report

ONR Contract N00014-85-K-0414
Theoretical and Pragmatic Issues in Software Engineering

1 April 1985 – 30 September 1990

Martin Davis and Elaine Weyuker

This document has been approved for public release and sale; its distribution is unlimited.
Although software engineering evolved pragmatically from efforts to develop practical paradigms for the development of reliable large-scale software systems, our work was based on the belief that significant progress would result from the kind of fundamental understanding that could only be obtained from deeper understanding of the underlying theoretical issues using models of computation fundamentally different from those studied in the now classical theory of computation.

Our research has included relatively abstract theoretical work, empirical studies, and system implementation. Our underlying belief was that software engineering research should follow the standards typical of most fields of science and engineering, in which researchers develop theories based on accepted models in the field and practical observations which are then validated by empirical studies. We have seen our work as an integrated whole in which experiments and implementations can suggest theoretical developments which in turn lead to further empirical work.

Beginning at the theoretical end of the spectrum, continuing previous investigations of formal notions of criteria for adequacy of software test data, we studied such a notion based on measures of distance between programs satisfying the axioms for a metric space. We found analogues of the concept of critical point emerging naturally from our work. We were also able to prove a new version of our earlier dichotomy theorem. In addition, we investigated the formalization of what properties are desirable for any software test data adequacy criterion. These properties were then used to compare several well-known testing criteria, and strengths and weaknesses were noted. In a similar vein, properties of good software complexity measures were also proposed, and an assessment of well-known complexity measures performed. Again, the strengths and weaknesses were noted.

Substantial progress was made in our ongoing work on data flow testing. In particular, a prototype tool, ASSET, was designed and implemented, and experimentation began to assess the usefulness of these proposed techniques. New theoretical work was begun to consider the effect of unexecutable program segments on the use and applicability of data flow testing. Interesting and surprising results showed that unexecutable paths can substantially change the relative effectiveness of various well-known testing strategies. A large empirical assessment of the cost of using data flow testing was performed during this period. We found that even the most demanding of our data flow strategies was surprisingly inexpensive to use, when assessment was based on the number of test cases needed to satisfy the selected data flow criterion. We also found, however, that the number of unexecutable definition use associations could be quite large, and that it can be costly and time consuming to make this determination. Finally, we developed algorithms to use data flow information as the basis for selecting regression test suites.

In a totally different vein, we also designed and implemented a specification-based
testing tool to help manage both the development of test cases, and their use in regression testing for large software systems.

Other work included a new domain-based notion of software reliability that had the unique property that it incorporated a notion of criticality of failure into the measure.

A different area of research included an analytic comparison of partition testing strategies and random testing. Our conclusions were somewhat different from those of earlier work done by others that used simulations to make the comparisons.

**PUBLICATIONS**


