# Perpetual Testing Research at Purdue Quarterly Technical Report Vol 1, Number 3

Michal Young
Department of Computer Sciences
Purdue University[1]
West Lafayette, Indiana 47907-1398
michal@cs.uoregon.edu

For the period: 1 August 1997 through 30 October 1997

Cooperative agreement: F30602-97-2-0034

Prepared for: U.S. Air Force, Air Force Materiel Command, Rome Laboratory

1. The principal investigator moved to University of Oregon on 15 August 1997. This report period includes work at both Purdue University and University of Oregon.

## Abstract

The Perpetual Testing project goal is to develop technologies to support seamless, perpetual analysis and testing of software through deployment and evolution. Whereas the current dominant paradigm treats testing as a phase that succeeds development and precedes delivery, the perpetual testing projects is building a foundation for treating analysis and testing as on-going activities to improve quality assurance without pause through several generations of product, in the development environment as well as the deployed environment. Improvements to existing technologies focus largely on scalability and incrementality for large evolving systems, attaining and maintaining adequate adherence of all software artifacts to relations captured by a rich web of hypercode links, including dependence relations among software components and among properties and analysis techniques.

The perpetual testing project is a collaboration of researchers at the University of Massachusetts, the University of California at Irvine, and Purdue University. Perpetual testing is part of the High Assurance cluster of DARPA-sponsored EDCS research.

## 1 Key Project Events During Quarter

The primary event of this period was moving the project from Purdue University to University of Oregon. No relevant DARPA-sponsored meetings were held during the period.

**Other presentations**

Michal Young presented an invited "State-of-the-art" report on software analysis and testing at the International Conference on Engineering of Complex Computer Systems in Como, Italy, 10 September 1997. Attendees at the session included personnel of the U.S. Federal Aviation Administration and Lawrence Livermore Laboratories.

## 2 Results

**Two-phase testing.** Two-phase testing refers to a combination of static analysis of a logical design model coupled with dynamic testing to demonstrate code conformance. It differs from directly testing an implementation for desired properties (which is being pursued in other sub-projects of Perpetual Testing) in the "indirect" approach of statically verifying a model and then using the model to generate test oracles, thus the term "two-phase." Direct and two-phase testing are not exclusive options, but are suitable for different properties. Two-phase testing is aimed particularly at properties that are dependent on non-deterministic and timing-dependent execution, such as race conditions and potential deadlock in concurrent software.

The immediately prior period (up to July 1997) was largely devoted to preparing a tool for the EDCS Demo Days. During the current period, we took stock of what had been learned and began a complete re-design of the prototype tool. Some of the redesign included incorporation of interfaces to work pre-

viously completed by other members of the project team, but omitted from the July 97 prototype in the interest of stability. In particular, the re-designed tool will replace the aging home-grown CATS/Pal analysis back end with an external tool, using a standard interchange format for process graphs. In a similar vein, the home-grown textual representation of process hierarchies will be replaced by a facility for importing and exporting architecture descriptions in the EDCS-standard ACME representation.

In addition to changes aimed primarily at interoperability, the major new functionality planned for the next development cycle (up to the next Demo Days) is increased automation of interface definition. In the current (July 97) prototype tool, processes (corresponding to "components" in ACME) can be split, combined, or introduced, and the user is prompted for descriptions of the interfaces ("connectors" in ACME terminology) that are affected by these changes. In the new design, many of these interfaces will be automatically derived.

Aside from work on the tool, graduate research assistant Yung-Pin Cheng has begun work on using architecture refactoring as an aid in inductive analysis of systems with many identical processes. This is initially pencil-and-paper work on small examples. It will be followed by small-scale experiments with existing tools, then development of tool support.

**Structure recovery** J While ideally the relation between architectural design and implementation would be maintained at every point in software development and evolution, in practice it is often necessary to "recover" the relation from legacy systems. The "fanasci" tool takes an architectural design description and an implementation structure (derived from source code), together with a partial mapping from implementation modules to architecture-level components, and "fills in" additional relations between implementation and architecture that are implied by architectural design constraints. We have previously reported a fanasci prototype that built up "uses" or "layers" hierarchies and a tool for extracting Ada task call graphs (call graphs restricted to calling paths between tasks). The graduate research assistant for this part of the project graduated at the end of summer, and this sub-project has been placed temporarily "on hold" pending recruitment of a suitable replacement.

**Residual test coverage monitoring.** "Residual" testing is testing that takes place after deployment. Work to date at Purdue has focused on monitoring of dynamic test thoroughness, as indicated by structural test coverage criteria, in deployed software. Residual coverage monitoring should indicate whether any test obligations which were not fulfilled in the development environment correspond to execution behaviors that occur in actual use. This could occur, for example, if an execution path that has been presumed infeasible by developers is in fact feasible. The primary question addressed by the current work is whether such monitoring can have sufficiently low impact on execution characteristics, particularly efficiency and responsiveness, to be acceptable to users.

At the end of summer, Christina Pavlopoulou finished her M.S. at Purdue and transferred to the Electrical Engineering department for Ph.D. study. Christina is continuing with some empirical studies using the existing tool, but new tool development is "on hold" for the time being.

**Other Activities.** Michal Young is technical program chair for the ACM International Symposium on Software Testing and Analysis, to be held in March 1998. Technical paper submissions, reviewing, and the program committee meeting were held during the current period.

## 3 Conclusions

This has a period for taking stock of results so far and planning for the next major round of development and demonstration. The main technical work during the period was a thorough redesign of tool support for two-phase testing. Some other sub-projects have been placed temporarily "on hold," as the students carrying on those projects graduated and the principal investigator moved from Purdue University to University of Oregon.

Some disruption from the move of the project was inevitable. The largest impact came from delaying hiring of a professional support programmer, pending completion of a subcontract arrangement between Purdue and University of Oregon. The level of graduate and undergraduate research assistant staffing has also been reduced. Some equipment purchases have also been deferred, although a few essentials (including one Sun workstation) have been purchased with "borrowed" funds at University of Oregon in anticipation of properly charging the project when financial arrangements are in place.