

Position Statement: Testing Complex Systems

P. Grabow

Computer Science Department
P.O. Box 97356
Baylor University
Waco, TX 76798 U.S.A.
grabow@cs.baylor.edu

Testing has become more difficult since the early days of computing. However, the nature of the difficulty may not be so obvious. Therefore it is helpful to consider the questions that the testing process has traditionally addressed: 1) What constitutes the system? 2) How should the system behave? and 3) How will the environment interact with the system? For some of today's systems, however, it may not be possible to adequately answer one or more of them -- greatly complicating the testing process.

In the early days of mainframe computing, usually the system boundary was obvious and the environment was constrained, e.g., confined to a room. And the system developer was not far removed from the user. Testing -- although not always easy --- was at least manageable. Today, though, the boundary of a complex system (e.g., wireless telecommunications) may change over time or it may not always be identifiable. Consequently, the system specification itself becomes problematic. How can you adequately specify the behavior of something that does not have definite boundaries?

The quality of the testing process has traditionally been tied to the definition of the system and its environment. In a broad sense, testing compares the implemented system to its specification (i.e., verification) and to the expectations of the environment/user (i.e., validation) [1]. If an adequate specification does not exist -- or a sufficient description of the environment is not possible -- how can these comparisons be made?

Suppose, however, that the system boundary is identifiable. Even here, though, an adequate specification may not be feasible. The behavior of the system may be so complicated that its specification is too difficult to understand -- beyond the abilities of the developer. For example, it may be impossible to determine if the specification is internally consistent and/or reasonably complete. This complexity may be unintentional -- increasing over time via a myriad of modifica-

tions and requirements that are beyond the control of the developer. The current air traffic control system is an example [2]. How can a sufficient testing strategy be constructed based on a specification that may be internally inconsistent and incomplete?

The system specification is also a problem when the "system" utilizes existing systems. For instance, thousands of financial systems use the Global Positioning System (GPS) for their time calculations because they require the accuracy of the GPS to calculate interest on multi-billion dollar loans (often to the nearest millisecond) [3].

Also, a system developer may not know or understand how the environment will eventually interact with the system. However, user expectations (or environmental demands) often determine the success of a system (even if the users are unrealistic or naive). For example, many of PageNet's 10.4 million customers were surprised when the Galaxy IV satellite failed in May 1998 [4] -- even though the price that they pay may not justify the expected reliability.

Therefore, the claim is this: Problems with testing are really problems with specifications (and designs).

References

- [1] Boehm, B.W., *Software Engineering Economics*, Prentice-Hall, 1981.
- [2] Perry, T.S., "Special Report: In Search of the Future of Air Traffic Control", *IEEE Spectrum*, Vol. 34, No. 8, 1997.
- [3] E. Yourdon and J. Yourdon, *Time Bomb 2000*, Prentice-Hall, New Jersey, 1998.
- [4] "PanAmSat Satellite Outage Interrupts Pager, Television Service in the U.S.", *The Wall Street Journal Interactive Edition*, May 20, 1998.