

Software Testing as an Integral Part of Educating for Network-Centric Warfare (NCW) and Information Assurance (IA)



by Dr. J. Bret Michael

As the U.S. Navy moves forward with its goal of achieving information superiority by affixing network-centric warfare (NCW) capabilities at the tip of the United States' spear, Naval officers will become more reliant on software-intensive systems to carry out their missions. These systems will provide advanced warfighting capabilities such as engage on remote (EOR), in which track data from external sensors, in the absence of local sensor data, is passed to the fire control component of a weapon system. The system uses this data to calculate launch parameters, fire the interceptor, and provide in-flight target updates to the interceptor, with the local weapon command center retaining control and responsibility for the engagement.

Systems that provide for cooperative engagement, and other NCW capabilities, will need to be of high quality—meaning that the system will have as few defects as possible. For example, a tactical action officer (TAO) expects an EOR system to be highly dependable, in terms of its availability, reliability, and ability to tolerate faults, in addition to meeting correctness criteria such as the system reaching its desired states given specific events and guard conditions. It is not acceptable for the system to become unavailable, because for instance, security flaws in the shipboard communication software permitted an adversary to modify the behavior of the system. Some software testing must be performed to reveal flaws that can cause the system to behave incorrectly, along with “off-nominal” testing to gauge the effects of inputs from the environment that could affect properties of the system such as its survivability or security; some inputs may result in desired systems behavior, while others may result in undesired or unknown system behavior. [1] The testing results, in addition to actual experience with the operation of the system, form the basis on which the TAO and other stakeholders develop their trust in the system.

There are a numerous reasons that the quality of these systems, in terms of their capabilities and nonfunctional properties (e.g., testability, security), can be difficult to assess. For instance, EOR takes place in a system-of-systems context, for which one must assess the emerging proper-

ties of the composite system rather than those of the individual subsystems; this can be especially problematic when the prime contractors and subcontractors working on the same weapon system do not fully exchange information about the subsystems with one another, but rather treat information for each subsystem as being company-proprietary. Another challenge is that DoD relies on capability-based acquisition, in which Government personnel only specify the capabilities of a system, while the contractor provides the customer with a statement of work as to how the capabilities and nonfunctional properties will be achieved and assessed. Another significant challenge is that such systems are largely comprised of software. Software can be complex—such as in terms of its logic, semantics, and dependencies between units of software—making it hard to uncover software defects. Moreover, the software units are often acquired as commercial-off-the-shelf (COTS) products—and, not all vendors provide detailed information about the internal workings or quality of their COTS products.

Among the many efforts underway at the Naval Postgraduate School (NPS) to support NCW initiatives, the faculty of the Department of Computer Science have created specialty courses (e.g., Engineering of Network-Centric Systems) and specialty tracks (e.g., the computer security track with an emphasis on developing EAL7 high-assurance systems), in addition to redesigning some of their existing courses to help prepare Naval officers for the task of acquiring high-quality software-intensive systems. This article, discusses the recent redesign of our course titled “Software Testing” to reinforce the materials the students learn in courses on NCW and related topics such as information assurance (IA).

Overview of the software testing course

This course is offered in the department of computer science curriculum for software engineering. It covers test planning, execution, and analysis. In addition to a thorough treatment of the theoretical underpinnings of software testing, the former is covered in the textbook by Binder on testing object-oriented software, that we rely on the textbook by

Education in Information Assurance (IA)



Friedman and Voas to introduce the students to software-testing theory. [2, 3] We supplement textbook material with a set of scholarly articles that discuss the latest thinking on how to improve both the quality of software and the effectiveness of software testing—these readings serve as the basis for in-class discussions.

The course is delivered simultaneously to both in-residence and distance-learning students, with the latter participating via interactive video teleconferencing. The lecture material and homework assignments are organized into learning modules that can be accessed via the Web-based Blackboard system—the School standardized on Blackboard for Web-enabled and fully Web-based delivery of courses. Our adoption of the Blackboard system for presenting the course material is in sharp contrast to the approach taken by Ramakrishnan, which involved developing a custom Web-based interactive environment called LIGHTVIEWS for teaching software testing. [4] The course on software testing takes advantage of two of the interactive features of Blackboard that support asynchronous learning—

- Quizzes that automatically provide feedback to students regarding their mastery of key concepts.
- Discussion forums on which the students post their thoughts on topics posed by the instructor and their classmates.

A major component of the existing course is a team-based project in which the students obtain hands-on experience developing a test plan, executing the plan, analyzing the test results, and presenting the results and lessons learned to their classmates. We subscribe to the approach described by Carrington of providing students with an existing software system to test (Carrington found that if students test a system that they have developed, they tend not to be motivated to try to uncover defects in their system). [5] Another advantage of Carrington's approach is that students learn firsthand about challenges such as the need to become knowledgeable about the application domain and contexts in which the software system will be used. Such a project is also important, as pointed out by

Brought and Reed for permitting students to experiment—using scientific methods—with different strategies and techniques for testing software systems. [6] In the past, we have supplied the students with the software for a simple discrete-event simulation of the operation of a Carrier-Sense Multiple-Access with Collision Detection (CSMA/CD) local area network, for which the requirements specification, design, and code are given. [7]

Redesign of the course

To better meet the educational needs of the students at NPS, we have redesigned the course on software testing by introducing a case study of a system that exemplifies, to some extent, the concept of NCW and the linkages between software testing and IA—the Ballistic Missile Defense System (BMDS), which is a system-of-systems comprised of Naval assets (e.g., the Aegis and Spy-1 systems) along with those of other services and agencies. The motivation for the case study is to demonstrate to the students the benefits, challenges, and limitations associated with software testing in the context of NCW and IA. Our approach of integrating the subject matter from other courses into the course on software testing is just the inverse of the proposal made by Jones to integrate software testing into other computer science courses. [8]

The case study is now an integral part of the lecture material and discussion topics. For instance, we have created learning modules and discussion forums that cover issues associated with the software testability of system-of-systems. Examples of discussion questions are—

- How does one ensure that laboratory results hold in the operational environment given that a system-of-systems' configurations are dynamic?
- If our test results are only valid for a specific configuration and particular set of variables, then how robust is our testing approach with respect to future system behavior in the operational world?

...continued on page 16

"Software Testing as an Integral Part of Education in NCW and IA"

- What guarantees, if any, can be made that the desired system behavior will be maintained in the software as patches and modifications are made after the system is fielded?"

In addition, we have developed team projects around the case study that are to be used to emphasize, for instance, the difference between testing techniques for achieving software quality (e.g., those for module or class testing), those for assessing software quality (e.g., system-level testing), and between feasibility testing (i.e., can the system provide the capability?) and operational capability testing (i.e., can the system provide the capability in an operational context?). The projects also emphasize important tasks associated with testing system-of-systems, such as distinguishing between controllable and uncontrollable system variables, with the aim of minimizing the negative impact on the system of those variables that can be controlled and characterizing the impact of external influences on the system that are outside the engineering-design space. Many of the students who take the software-testing course become software acquisition officers rather than software developers, so we also cover software acquisition topics as they relate to, for instance, testability.

As previously stated, the criticality of BMDS to our nation's security dictates that such safety-critical systems be of high quality. In security courses, the topic of assurance as it pertains to NCW is typically discussed in terms of penetration analysis and formal verification of security kernels. We revised the course on software testing to provide students with lecture material that clearly delineates differences among penetration analysis, formal verification, and software testing. Likewise, we created experiments for the students to conduct. This allows them to discover firsthand some of the pros and cons associated with applying security-specific and off-nominal testing (e.g., fault injection) techniques to reveal security flaws. For example, the fact that fault injection permits the testing of COTS components for which the source code is not available, and the weaknesses of penetration analysis, one of which as pointed out by Du and Mathur is that the tester must either know a priori the types of flaws that exist in BMDS or be able to postulate what those flaws might be. [9] We have also added to the supplementary reading list articles that discuss ways of improving the testing for security flaws, such as the techniques described by Jiwnani and Zelkowitz to direct the application of scarce testing resources based on the distribution and prioritization of security vulnerabilities. [10] In this course, we also discuss such a prioritization of resources from the perspective of safety, reliability, and availability.

Lastly, we plan to invite personnel from combatant commands, Government agencies, and the private sector to give guest lectures. However, often our students have prior experience in conducting network-centric warfare, managing information assurance, or performing software testing—their expertise helps bring to light for their classmates real-world challenges faced by the user, software-systems engineer, and software-acquisition professional.

Technology transfer

We are assisting faculty affiliated with the federally funded National Institute for Systems Test and Productivity (NISTP), located at the University of South Florida, to introduce DoD-specific content into their graduate-level course on software testing. In addition, we are studying the lessons learned reported by others from their experience in teaching software testing to graduate students. For example, we might be able to apply certain aspects of the approach reported by Hoffman, Strooper, and Walsh to improve upon our current design of the learning module on the subject of automated testing. [11] Our efforts are being funded by research grants from the Space and Naval Warfare Systems Command and Missile Defense Agency. ■

About the Author

Dr. J. Bret Michael

Dr. Michael has been an Associate Professor of Computer Science at the Naval Postgraduate School since 1998. His research on information assurance and information operations covers many aspects of distributed computing. Dr. Michael is a member of the IATAC Steering Committee. He may be reached at bmichael@nps.navy.mil.

References

1. Ghosh, A. K. & Voas, J. M., Inoculating software for survivability, *Comm. ACM* 42, 7 (July 1999), pp. 38–44.
2. Binder, R. V., *Testing Object-Oriented Systems: Models, Patterns, and Tools.*, Reading, MA: Addison-Wesley, 2000.
3. Friedman, M. A. & Voas, J. M., *Software Assessment: Reliability, Safety, Testability*, New York, NY: John Wiley & Sons, 1995.
4. Ramakrishnan, S. "LIGHTVIEWS—Visual interactive Internet environment for learning OO software testing." In *Proceedings Int. Conf. on Software Eng.*, IEEE (Limerick, Ire., June 2000), pp. 692–695.
5. Carrington, D., "Teaching software testing." In *Proceedings Second Austral. Conf. on Computer Sci. Educ.*, ACM (Melbourne, Aust., July 1996), pp. 59–64.
6. Braught, G. & Reed, D., "Disequilibration for teaching the scientific method in computer science." In *Proceedings Thirty-third SIGCSE Tech. Symposium on Computer Sci. Educ.*, ACM (Covington, KY, 2002), pp. 106–110.
7. Sadiku, M. N. O. & Ilyas, M., *Simulation of Local Area Networks*, Boca Raton, FL: CRC Press, 1994.
8. Jones, E. L., "Software testing in the computer science curriculum—a holistic approach." In *Proceedings Austral. Conf. on Computing Educ.*, ACM (Melbourne, Aust., Dec. 2000), pp. 153–157.
9. Du, W. & Mathur, A. P., "Testing for software vulnerability using environment perturbation." In *Proceedings Int. Conf. on Dependable Systems and Networks*, IEEE (New York, NY, June 2002), pp. 603–612.
10. Jiwnani, K. & Zelkowitz, M., "Maintaining software with a security perspective." In *Proceedings Int. Conf. on Software Maint.*, IEEE (Montreal, Can., Oct. 2002), pp. 194–203.
11. Hoffman, D., Strooper, P., & Walsh, P., "Teaching and testing." In *Proceedings Ninth Conf. on Software Eng. Educ.*, IEEE (Daytona Beach, FL, Apr. 1996), pp. 248–258.