Accelerated Stress & Reliability Testing for Software and Cyber-Physical Systems

Jeremy Straub
Department of Computer Science
University of North Dakota
Grand Forks, ND, USA
jeremy.straub@und.edu

***Abstract*—This paper considers the topic of accelerated testing of cyber-physical systems and software. It compares the testing of these systems to each other and to classical mechanical/electro-mechanical systems. The impact of the software (and combined hardware-software element) on accelerated testing is also considered. A case study is used to examine and illustrate key aspects of cyber-physical system testing. This case study relates to a near worst-case example: a long-duration autonomous spacecraft mission. The paper then discusses one prospective approach for testing software and cyber-physical systems, the use of an autonomous testing system, and discusses the benefits of this approach (including being able to incorporate elements of this system and its knowledge base into an onboard maintenance system) before concluding.**

I. Introduction

The accelerated testing of software is significantly different from the testing of mechanical, electrical and combined electromechanical systems. While software testing may be able to run in 'virtual time', simulating days, weeks, months or more of operation in as little as minutes or hours, this approach fails to fully simulate the long-term operation of the software. Additionally, quantity-based testing (i.e., where the expected number of manipulations of an item over its lifetime is projected and this quantity is tested) falls far short of testing the long-term operations of software. Further complicating matters is the testing of a cyber-physical system, where the long-term combined operations of both electromechanical hardware and software must be projected. This paper considers the accelerated testing of long-running software. It utilizes a near worst-case scenario (due to the lack of physical human access) of a long-term deep space mission to illustrate the particular needs of automated testing of software and cyber-physical systems. It proposes the use of autonomous software for the analysis and testing of such a system and describes how the results of this testing could be used to inform the creation of an onboard maintenance / reliability assurance system. The paper concludes with a discussion of the efficacy of accelerated testing for each system type and considerations for future work in this area.

II. Background

This section provides an overview of prior work in several relevant areas that informs the current work. It beings with a discussion of cyber-physical systems and their testing. Then, the testing of software (including, in particular, artificial and computational intelligence software) is considered. Finally, the problems of the long-term operations of cyber-physical systems are reviewed.

A. Cyber-physical Systems and their Testing

Rajkumar, et al. [1] suggest that cyber-physical systems represent a new "computing revolution". These systems combine hardware actuation, sensing and other capabilities with the decision making, data storage and other benefits of a robust software system. Cyber-physical systems are found performing numerous functions ranging from automotive [2] and aerial [3] transportation to supporting power grids [1], healthcare [4], scientific discovery [1] and emergency response [1].

Because of their interaction with the real world, cyber-physical systems have considerations beyond those of many other computing systems. Testing takes on a level of particular importance in the context of systems that can injure humans, cause physical damage or otherwise negatively impact the real world through their failure. The difference of cyber-physical systems from typical data-processing computing applications have caused some to suggest that current "computing foundations" may be inadequate for the challenge [5]. Systems must be validated not only to perform with expected inputs (and under typical operating conditions) but also to deal with the unexpected gracefully.

Cyber-physical system validation activities, thus, must cover a wide range of prospective areas of concern. Conformance testing [6, 7] validates whether the system meets key identified operations criteria including both functional and non-functional requirements. Integration and interoperability testing [8] ensures that the system will function in conjunction with other systems and that internal components work together adequately. For many systems (e.g., mission and safety critical systems such as power grids), security is a particular challenge [9]. Securing a cyber-physical system must consider its defining characteristics such as its interactions with the real world, management / control methodology, uncertainty, performance requirements and physical dispersion [10]. A variety of security approaches have been proposed including attack detection and response [11]. The use of testbeds has also been proposed [12].

In addition to proper functionality of cyber-physical systems and their security, their sustainability for continued operations must also be considered [13]. A variety of methodologies [14] and validation approaches [15] have also been proposed including the use of artificial intelligence for testing cyber-physical system command decision making [16].

B. *Testing of Software and Artificial / Computational Intelligence Systems*

Significant prior work exists related to the testing of software and, in particular, artificial and computational intelligence systems. Cholewinski, et al. [17], for example, show how autonomous testing could be used via testing the Default Reasoning System (DeReS). For this, they used test cases which were based on the TheoryBase benchmarking system. In this work, they contended that TheoryBase was used to validate DeReS. Specifically, it was used to demonstrate that it was implemented properly and would function. Billings, et al. [18] show how a similar aim can be achieved using self-play testing. Under the self-play testing approach, two versions of the autonomous control software are run against each other. The performance of the two systems is compared, facilitating the prospective identification of bugs and performance issues. An alternate approach is 'live-play' testing, where the performance of the system is compared to a human player's performance. In this example, the testing was used to demonstrate that new versions of Poki (poker playing software) improved upon (or at least didn't significantly underperform previous versions) before they were publically released.

Other work demonstrated how mutation can be used in and enhance testing. Wotawa, et al. [19] performed work related to localizing faults in source code. Their algorithm is based on manually creating base test cases and applying mutation. This approach is also applicable to many autonomous testing routine input parameters.

Even base cases can be autonomously created, AdiSrikanth, et al. [20] contend. In this work the used an artificial bee colony algorithm seek paths through the program and generate test cases based on what they've learned. This exhaustive approach, however, is also quite costly for large programs. It is also predicated on an assumption that the program's logic is correct and thus it is primarily looking for small bugs to correct.

Since testing has been shown to be a search or exploration problem, a multitude of algorithms can potentially be used for this purpose. Examples of prospective algorithms include the Cuckoo [21], Water Drop [22] and Firefly [23] approaches.

C. *The Problem of Long Term Operations*

Long-term unattended operations are a near worst-case scenario. Their consideration is helps demonstrate the need for accelerated testing (as replicating a ten year or longer mission in real time is probably not feasible). An interplanetary exploration mission scenario is presented which could be of this length or longer.

Multiple types of long-term exploration missions are relevant. These could include missions to the outer plants or even the solar systems of other stars. The spacecraft used in these missions, once they are significantly away from the Earth, can effectively be considered to be a transmit-only system, as discussed in [24]. While mission controllers may send updates or even new tasking, the transmission delay means that for virtually all purposes the spacecraft and its command system must make decisions without the real-time (or even near real-time) aid of ground controllers.

Additionally, due to the delay and bandwidth limitations, the ability for controller intervention for maintenance purposes is limited and undesirable. Thus, the spacecraft needs to be able to perform maintenance activities (such as cleaning up memory and storage usage of a long-running program) on its own.

In [24], a Blackboard Architecture-based system was presented. Its autonomous maintenance needs were largely driven by the need of a particular speed of maintenance being performed. The system was, in some cases, required to perform real-time and near real-time decision making. Maintenance activities could (among other things) reduce the amount of time required for this decision making by removing irrelevant and unneeded information from the blackboard and thus from consideration in the decision making process.

The longer the blackboard command system runs (without the pruning of unneeded information being performed) the more data that will be stored on the Blackboard. Left unresolved, the system's storage and memory would fill and the speed of decision making would decline. Accelerated testing would be required, under this scenario, to ensure that the increase in data over time (after seeing a wide variety of new phenomena and through the impact of normal system operations) is kept in check and does not result in excessive storage needs (and, potentially, to help determine non-volatile storage requirements) or an excessive delay in decision making.

III. TESTING IN VIRTUAL TIME

One approach to testing a cyber-physical system that is expected to operate over an extended period of time is to test in 'virtual' time. Under this approach, testing proceeds at a rate that is a multiple of the typical progress. System inputs are sent and outputs expected at a much faster than normal rate.

A. *Description*

One approach to testing (in particular software components of) cyber-physical systems is the use of accelerated (or 'virtual') time. With accelerated time, the system runs significantly faster than (typically at a multiple of) its normal pace. An entire mission or scenario that takes hours in real time may be tested in minutes. Typically, some level of detail is lost by this process and testing focuses on identified key areas (excluding, for time purposes, other inputs and real-world happenings considered extraneous to the testing).

B. Benefits

The key benefit to this approach is the ability to test the response to a wide variety of conditions in a time-effective and controlled environment. This facilitates performing testing under numerous (perhaps unlikely) combinations of inputs and happenstance. Any defects identified through this accelerated testing process can be corrected to ensure that the final system is robust to the tested (and unsuccessfully completed) scenario.

Accelerated testing is also useful for gaining a holistic understanding of the performance and outcomes of a mission and for estimation purposes. It is also useful for evaluating a high-level operating approach (without having this evaluation be confounded by issues with the implementation of lower-level hardware or software components).

C. Problems

The largest problem with this approach has been previously identified as also being a benefit. The abstraction that is used (and typically, absent significantly greater-than-actual-system processing capabilities) can prevent problems that would exist in the real-world from being identified. This type of issue can come from two sources. The first is the simulation environment that is used to create the faster-than-real-time operations. It may not fully encapsulate all relevant areas of real world operations or may oversimplify or erroneously represent relevant (and perhaps not known a priori to be relevant) factors. The second is the abstraction that is (in many cases) required to operate at significantly faster-than-real-time speeds. This abstraction may hide integration or lower-level implementation issues or demonstrate suitable performance under simplified scenarios that should not (due to scenario design and configuration decisions) be extrapolated to all real world operations.

A more general problem that is inherent to this approach is the inability to test various hardware components at significantly greater-than-normal speeds. In some cases, the hardware is simply not capable of meeting the higher speed requirements. In other cases, performance at these higher speeds is sufficiently different such as to make the data unhelpful to predicting longer-duration (at normal speeds) performance.

IV. Quantity-Based Testing

Quantity-based testing parallels, for certain types of hardware, the accelerated speed testing for software. Under this approach, lifetime use levels are defined and parts are tested to ensure that they can survive this many uses (plus provide an additional safety margin).

A. Description

Quantity-based testing attempts to, within a short period of time, test a lifetime's worth of manipulations (or other uses) of a hardware component. This process begins with estimating the total number of uses that the particular hardware component will be subjected to. Some level of safety margin is added to this (representing both the possible estimation error plus some acknowledgement of the potential time-based deterioration of the hardware, in addition to the use-based wear). Then, a (typically automated) process is devised to repeatedly perform the type of use of the hardware component being tested.

B. Benefits

The principal benefit of this approach, like with the accelerated testing (most well-suited to software), is the ability to test a lifetime's worth of use within a constrained period of time. Because of some of the drawbacks (discussed in the next subsection), this type of testing represents a near best case scenario (presuming that the margin between actual projected uses and tested uses is not significant). If the hardware component fails under this testing, thus, it would seem to be (in most cases, presuming the absence of testing error) a fairly safe assumption that it would not perform suitably in actual use.

Other benefits include the prospective speed of setup and performance of the testing and the approach's simplicity and ease of understanding. For issues identified in this way, there may be significant cost benefits compared to other prospective testing techniques

C. Problems

There are four types of issues with this approach. Each is now briefly discussed.

The first type of issue is the potential that the testing does not actually reflect the rigor-level of actual use. If users are rougher with the hardware than expected, or if they apply force in different areas or manners or at different levels than expected, the testing may not reflect the object's performance in real use.

The second category of prospective issues relates to failed assumptions. For example, assumptions regarding the number of uses, frequency of use, impact of non-use related wear and a number of other factors could render the testing irrelevant and unreliable if they are materially wrong.

The third issue with this sort of testing is that it is hardware-centric and does not effectively test software components (or hardware for which the wear levels may be different based on software decision making during use).

Finally, the robustness of the testing system may be another area of potential problem. The regular forces regularly applied by the system and/or the combined wear of the system being tested and the system performing testing, over time, may also cause the system to fail to fully capture certain types of issues that might occur during real world use.

V. Combined Software and Cyber-physical System Testing

Having now discussed two techniques, the first of which was better suited to software and the second of which was targeted more at hardware, the question of how to rapidly test a full cyber-physical system is now considered.

A. Description

Even with a combined technique, one cannot expect to fully rapidly test a cyber-physical system. The increased speed inherently imposes limitations on how much of the real world operating conditions the hardware-software system under test is exposed to.

Unlike with the separated approaches, however, the combined approach aims to rapidly provide matching simulated input to both hardware and software components simultaneously.

To do this, testing scenarios must be developed, based on functional and non-functional requirements. Ideally, complex intersections of a number of different potentially relevant factors can also be considered to attempt to determine if the testing plan itself may have flaws or failed assumptions.

This approach requires synchronization between hardware and software simulation units as well as scenarios that are designed to both electronically and physically stimulate and cause actuation of the cyber-physical system.

B. Benefits

The principal benefit of this combined approach is the ability to test software-hardware integration and catch issues that the two separate testing systems were not able to catch on their own. Because the two previous approaches may operate significantly faster than this approach, it may be advisable to use the separated testing first, before graduating the system to the combined testing. Otherwise, the combined testing may identify issues that would have been able to be identified via the individual testing approaches (faster) and which may hide integration issues (requiring the more time-expensive combined testing to be run multiple times to reach an acceptably low level of detected issues.

Like the previous approaches, this approach does allow system designers to focus testing on particular areas of interest. It also facilitates the disambiguation of real world happenstance from logical design and implementation issues.

C. Problems

This combined approach suffers from many of the problems that the two separate approaches did, as well as some problems particular to the combination. Like the individual approaches, this approach is highly reliant on a set of assumptions regarding what to test and when to test it. If these assumptions are invalid, the testing process may fail or it may fail to catch errors that may present themselves during real world use.

Scenarios may fail to test key system areas, fail to test in a way the causes an issue to be identified and not adequately reflect the impact of long-term operations on both hardware and software components.

Additionally, the simulation system itself may have issues. For example, it may fail to note an issue that presents itself or classify it correctly.

The combination of the two types of inputs and actuation may slow down testing significantly, as efficiencies gained by rapidly testing and re-testing physical parts may not be enjoyed. On the other hand, having to interact with and wait for the physical hardware will prevent the software testing from reaching the same speeds that it potentially could during a software-only testing regime,

VI. Case Study: Long-Duration Space Exploration Mission

A case study is now presented to demonstrate the efficacy of the previously discussed solutions in some areas and also illustrate areas of testing that are still underserved by the solution. This case study is based on a long-duration autonomous spacecraft mission, like might be performed for planetary science or similar purposes.

This scenario is characterized by several requirements. First, the cyber-physical system will be running continuously for multiple years (absent a limited number of intentional reboots and potentially some power-saving periods). Second, during this period, the system will be continuously collecting data and adding some of this data to its decision making, data processing and other software systems. Third, there will be no direct human access to this system to perform maintenance, change configuration settings or fix problems that arise.

A. Testing Plan

For a system such as this, several types of testing would be relevant. The first considerations would be similar to those for most cyber-physical systems: does the software work, does the hardware work and does the combined system work. With unit and sub-system (hardware / software) level testing and integration testing (hardware and software operating together) the basic functionality under standard operating conditions can be assured using conventional techniques.

The second area of consideration is the long-duration operations of the system. There would be two areas that would need to be assessed for this. The first would be to check for issues that may simply be tied to elapsed time (e.g., variable overruns and hardware use limitations). The accelerated testing for the software and hardware systems, described previously, is well-suited to answer these questions, as long as testing plans are well thought out and properly implemented.

The other question that must be answered in this area is dealing with all of the data that the system will be exposed to. This may include sensed data, spacecraft telemetry, error tracking data and such. Thus, it would be necessary to develop another testing regime that would be effective at simulating dealing with a similarly sized data set over a simulated operating period.

Because of the autonomous control, there would also be a need to validate the decision-making capabilities of the software system under both standard conditions as well as foreseeable non-standard conditions and considering how the system would react to an unforeseen situation (without immediate human aid). This may require the use of another software system that generates unexpected scenarios to test the onboard command software under a multitude of projected and randomly generated conditions. The use of autonomous testing is discussed further in Section VII.

B. Evaluation

The previously discussed testing solutions are not, by any means, a panacea for testing this type of a system. They require deliberate customization and the accuracy of the assumptions and the implementation of the testing plan is critical to the efficacy of the testing in detecting possible areas of concern. The three approaches, however, do provide a basic framework for structuring tests. Practically, far more would need to be done to extend these basic concepts to suit this type of a mission than is gained from the concepts in their base form.

VII. Autonomous Testing and Analysis

Autonomous testing provides an alternate or augmentative approach to manually configured and manually performed or automated testing. Autonomous testing can prospectively take several forms. First, autonomous testing could be designed to follow and expand on a human-designed testing program. For example, the autonomous system could expand the range of values manually identified to be tested. The approach taken could be designed to vary based on initial results (expanding, for example, to gather more examples if a problem was found or avoiding additional testing related to a well-documented problem).

Second, autonomous testing could be designed to explore the problem space and identify other areas (not logically selected by a human test plan designer) that could potentially experience problems. This exploration could be random or it could use a priori information about the design of the software and the testing plan to explore more deliberately.

Finally, autonomous testing could be observation based. A software system could be created to observe user behavior (or problems, or security attacks, etc.) and develop test cases to replicate, stress and concurrency test and expand upon standard user behaviors.

VIII. Creation of Onboard Maintainance System

Using the information gained through testing, it may be possible to create an onboard autonomous maintenance system for some applications. This system would utilize the detected faults from the autonomous error detection system and knowledge about their resolution. It would either implement pre-programmed responses or proactively work to resolve the problem based upon a knowledge base. The accelerated testing and (potentially) the learned responses from observing and training from user activities in response to faults would inform the operations of this system.

IX. Conclusions and Future Work

This paper has presented an overview of the use of accelerated testing for cyber-physical systems. It has discussed prior work related to cyber-physical systems and their testing and presented a framework for the accelerated testing of such systems. The utility of this testing has been partially evaluated and a case study has been presented further illustrating the difficulties and benefits of the accelerated testing of cyber-physical systems.

References

References

[1] R. R. Rajkumar, I. Lee, L. Sha and J. Stankovic. Cyber-physical systems: The next computing revolution. Presented at Proceedings of the 47th Design Automation Conference. 2010, .

[2] C. Berger and B. Rumpe. Autonomous driving-5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system. Presented at Proc. of the 10th Workshop Automotive Software Engineering, September 2012, Braunschweig, Germany. 2012, .

[3] K. Sampigethaya and R. Poovendran. Aviation cyber–physical systems: Foundations for future aircraft and air transport. *Proc IEEE 101(8),* pp. 1834-1855. 2013.

[4] J. Shi, J. Wan, H. Yan and H. Suo. A survey of cyber-physical systems. Presented at Wireless Communications and Signal Processing (WCSP), 2011 International Conference On. 2011, .

[5] E. A. Lee. Cyber-physical systems-are computing foundations adequate. Presented at Position Paper for NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap. 2006, .

[6] M. Woehrle, K. Lampka and L. Thiele. Conformance testing for cyber-physical systems. *ACM Transactions on Embedded Computing Systems (TECS) 11(4),* pp. 84. 2012.

[7] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski and K. Ueda. Conformance testing as falsification for cyber-physical systems. *arXiv Preprint arXiv:1401.5200* 2014.

[8] M. Spichkova, H. Schmidt and I. Peake. From abstract modelling to remote cyber-physical integration/interoperability testing. *arXiv Preprint arXiv:1403.1005* 2014.

[9] S. Sridhar, A. Hahn and M. Govindarasu. Cyber–physical system security for the electric power grid. *Proc IEEE 100(1),* pp. 210-224. 2012.

[10] C. Neuman. Challenges in security for cyber-physical systems. Presented at DHS: S&T Workshop on Future Directions in Cyber-Physical Systems Security. 2009, .

[11] F. Pasqualetti, F. Dorfler and F. Bullo. Attack detection and identification in cyber-physical systems. *Automatic Control, IEEE Transactions On 58(11),* pp. 2715-2729. 2013.

[12] A. Hahn, A. Ashok, S. Sridhar and M. Govindarasu. Cyber-physical security testbeds: Architecture, application, and evaluation for smart grid. *Smart Grid, IEEE Transactions On 4(2),* pp. 847-855. 2013.

[13] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee and S. K. S. Gupta. Ensuring safety, security, and sustainability of mission-critical cyber–physical systems. *Proc IEEE 100(1),* pp. 283-299. 2012.

[14] H. Woo, J. Yi, J. C. Browne, A. K. Mok, E. Atkins and F. Xie. Design and development methodology for resilient cyber-physical systems. Presented at Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference On. 2008, .

[15] E. M. Clarke and P. Zuliani. "Statistical model checking for cyber-physical systems," in *Automated Technology for Verification and Analysis*Anonymous 2011, .

[16] J. Straub and J. Huber, "A Characterization of the Utility of Using Artificial Intelligence to Test Two Artificial Intelligence Systems," *Computers,* vol. 2, pp. 67-87, 2013, 2013.

[17] P. Cholewiński, V. W. Marek, M. Truszczyński and A. Mikitiuk. Computing with default logic. *Artif. Intell. 112(1),* pp. 105-146. 1999.

[18] D. Billings, A. Davidson, J. Schaeffer and D. Szafron. The challenge of poker. *Artif. Intell. 134(1),* pp. 201-240. 2002.

[19] F. Wotawa, S. Nica and M. Nica. Debugging and test case generation using constraints and mutations. Presented at Intelligent Solutions in Embedded Systems (WISES), 2011 Proceedings of the Ninth Workshop On. 2011, .

[20] AdiSrikanth, N. J. Kulkarni, K. V. Naveen, P. Singh and P. R. Srivastava. "Test case optimization using artificial bee colony algorithm," in *Advances in Computing and Communications*Anonymous 2011, .

[21] X. Yang and S. Deb. Cuckoo search via lévy flights. Presented at Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress On. 2009, .

[22] H. Shah-Hosseini. Problem solving by intelligent water drops. Presented at Evolutionary Computation, 2007. CEC 2007. IEEE Congress On. 2007, .

[23] C. B. Pop, V. Rozina Chifu, I. Salomie, R. B. Baico, M. Dinsoreanu and G. Copil. A hybrid firefly-inspired approach for optimal semantic web service composition. *Scalable Computing: Practice and Experience 12(3),* 2011.

[24] J. Straub, "Automating Maintenance for a One-Way Transmitting Blackboard System and Other Purposes," *Expert Systems,* in press.

BIOGRAPHY

Jeremy Straub conducts research in 3D printing, spacecraft development, autonomy and policy at the University of North Dakota. He has published over 35 journal articles and 120 full conference papers (in addition to numerous other conference, panel and other presentations) on topics ranging from the development and assessment of technology to technology policy and law. As a result of this work, Jeremy has won multiple best paper/poster awards, been included in Marquis Who’s Who in the World and Who's Who in America and been inducted into several professional honorific societies including Sigma Xi. Jeremy's work at UND has been featured in numerous media publications, coast-to-coast, including coverage in the Albuquerque Journal, Houston Chronicle, Washington Times, Oklahoman and San Francisco Chronicle. Prior to returning to academia, Jeremy had a successful career in industry where he held progressively responsible positions in software and technology development, technology management and management. He was responsible for the development of North America's first traffic-adaptive navigation solution and his work was featured in numerous media articles in publications including Entrepreneur, Forbes and the Silicon Valley Business Journal.