Test and Evaluation of a SoS using a Prescriptive and Adaptive Testing Framework

John T. Hess
Massachusetts Institute of Technology
Cambridge, MA, USA
johnhess@mit.edu

Ricardo Valerdi
Massachusetts Institute of Technology
Cambridge, MA, USA
rvalerdi@mit.edu

Abstract - *Testers need the ability to adapt test planning on the order of days and weeks. PATFrame will use its reasoning engine to prescribe the most effective strategies for the situation at hand. Strategies in this context include methods of experimental designs, test schedules and resource allocation. By facilitating rapid planning and re-planning, the PATFrame reasoning engine will enable users to use information learned during the test process to improve the effectiveness of their own testing rather than simply follow a preset schedule. This capability is particularly attractive in the domain of Systems of Systems testing because the complexity of test planning and scheduling make frequent re-planning by hand infeasible.*

Keywords: *Test and Evaluation, Scheduling Problem, Adaptive Testing.*

1 Introduction

A battle plan seldom survives first contact with the enemy. This is because a battle plan, like a test plan, is predicated on incomplete information and is prone to uncertainties and unexpected events. A well planned test schedule for a major system can be "overcome by events" (OBE) at any time, and we have no way of knowing when or how this will be. Sometimes, testers may not even realize they've been "OBE". For example, any time a tester has discovered a deficiency in their System Under Test (SUT) that will require a system redesign and subsequent "regression loop" in our testing process but have not yet re-planned, they may be wasting time and money executing their original test plan as scheduled.

Testers are, like the majority of their acquisition counterparts, constantly asked to do more with less. They also face a special challenge as a by-product of being a downstream process from design and delivery: they are often asked to absorb cost and schedule overruns of other phases of acquisition. Because of the subjective nature of test planning, test organizations and testers in general are asked to make decisions under a great deal of uncertainty. Testers must generate and defend budget and schedule estimates months and even years in advance of testing.

These difficulties are compounded when testing in a SOS environment. Decentralized system responsibility and the inescapable need to interact with legacy systems -- some decades old -- make it nearly impossible to find, assign responsibility for, and correct SOS defects. It is prohibitively expensive, if even possible, to perform an adequate amount of live SOS testing for the massive number of permutations of interactions possible between weapons systems. Furthermore, aside from RF interoperability, there are virtually no formal SOS requirements to test against in Developmental Testing, meaning we often rely on Stewart-esque "I know it when I see it" criteria to define SOS failures.

Testing Unmanned and Autonomous Systems (UASs) with novel requirements complicates matters even further. Test and Evaluation planning is burdened with a great deal of uncertainty even when testing incremental upgrades to commodity products. This uncertainty grows by orders of magnitude when testing an entirely new genus of machine. As individual systems approach full autonomy, and are inevitably given more sensitive decisions to make, we will be forced to test their intelligence using metrics and methods not yet envisioned.

The overwhelming amount of uncertainty logically leads to more complex decision making. Unfortunately, as humans, we are notoriously poorly equipped to deal with this complexity. We are poor at estimating probabilities, and personal and institutional factors lead us to be risk-averse or risk-seeking even when it's to our detriment. As humans, we are prone to "satisfice" [15], that is we make "adequate" but suboptimal decisions. In a world where tests can cost

upwards of a million dollars, "adequate" just won't do.

Through several interviews and workshops with Department of Defense testers, we've identified several key questions, including:

- How much testing is enough?
- How do I prioritize my tests?
- How do I test effectively in a compressed schedule?

We propose a solution to these challenges: A Prescriptive and Adaptive Test Framework (PATFrame). PATFrame's Decision Support System will provide prescriptive advice designed to help users address the gaps between the normative models of testing, "how they should test", with their current practices, or descriptive models.

This paper defines a component of PATFrame's Decision Support System that will allow testers to rapidly plan and re-plan entire test programs quickly and effectively in light of newly learned information. When integrated with other techniques, users will be able to use this decision support system to track risk in their systems, estimate the cost and duration of their test programs, and identify critical test events. This sheds light on the risk in the test system, not just the system under test.

This component has two main parts, a data repository and a set of algorithms. It will be implemented as a computer program. This paper will present PATFrame conceptually. The implementation and validation of this framework is left for future work.

2 Methodologies Leveraged

We have leveraged three existing methodologies in our design for PATFrame:

2.1 Design of Experiments (DOE)

Design of Experiments is a set of methods for efficiently gathering information. The simplest incarnations of DOE facilitate planning tests of a system by taking into account information already known about dependent variables and their effects (including interactions with one another) on the independent variables being measured. DOE includes methods for dealing with variability, unknown interactions and a host of problems that face experimenters.

2.2 Defect Modeling

In a 2008 whitepaper, IBM proposed a method for tracking software defects discovered during a test program. They proposed using this information to project the number of remaining defects, and the rate at which they will be discovered. This method is used to answer the question (and title of the whitepaper) "When am I done testing?" We adopt the underlying methodology with the acknowledgment that the exact trends of defect discovery in a weapons system (or a SoS) will differ from those IBM proposed.

2.3 Exploratory Testing (ET)

ET, a strategy used in software testing, leverages human judgment to improve a test program "on-the-fly" [2]. As opposed to scripted testing, ET encourages the tester to use information learned from their previous tests to select the next test case. The underlying logic of this system, which we co-opt, is that information learned during a test program can be used to select more effective test cases for the remaining tests. In other words, the more you know, the better you can plan your testing, so why plan your whole test program before you've learned anything?

2.4 Integration

The last of these methodologies, exploratory testing, was the inspiration for PATFrame's core re-planning function. The other two methodologies, DOE and defect modeling, both support sub-functions of PATFrame. Clearly, PATFrame's functionality can be expanded with the application of countless other normative methods and best practices. For the sake of simplicity, not all are discussed here.

3 Core Prescriptive and Adaptive Framework (PATFrame)

PATFrame is composed of a data repository and a set of algorithms that are used to plan and, more importantly, re-plan a given test program. The data repository is used to specify the testing required and the constraints placed on the program, and the algorithm executes PATFrame's underlying heuristics, which are described below.

The core functionality of PATFrame is to plan a test program by matching all the tests to be conducted with the times that the resources are available. This is minimally useful, since most test programs have neither the money nor the time to fully test all requirements. The valuable feature of PATFrame is its ability to apply the methodologies listed above and our four heuristics to plan an effective test program, even in the face of uncertainty.

Presented here is a subset of possible elements of PATFrame. Clearly, the amount of information expressed in the data repository and the number of algorithms and heuristics used to plan the test program can be expanded.

3.1 The PATFrame Data Repository

The information stored in the repository falls into two main categories: (1) Information used to describe the requirements for the SUT and the tests needed to verify them and (2) information that describes the constraints placed on the test program.

The requirements section details the Measures of Performance (MOP) associated with each requirement. A requirement like "Must out perform all current USAF fighters" might include several measures of performance such as Maximum Altitude, Maximum Speed, and Maximum Acceleration. Information like the relative importance and the desired statistical confidence for each MOP (i.e. I want to know that I met the passing criteria Maximum Altitude to a 95% confidence) is included in this repository. Desired confidence is usually set by the test center, but may be altered on a project or even MOP basis. This is largely a risk management and costing decision, since higher certainty is more time consuming and more expensive to achieve. Furthermore, the user specifies the current confidence that the System Under Test (SUT) will pass each MOP.

Surely, defining the confidence that a SUT will pass (or has passed) a given MOP is daunting. However, a number of methods for estimating this sort of probability exist [8]. No one will know these values for sure (if they did, we wouldn't need to test), but we can make educated guesses and use them to help plan our test program.

While it may cause some consternation in the DOD test community, it may well be that the SUT's prime contractor can provide the most information about the system (indeed, when a new SUT shows up for testing, the testers know much less about the system than its builders). Since the prime contractor is likely to know "where the bodies are buried", their input in making this estimate is invaluable. We do acknowledge the vast array of political and organizational barriers to this sort of information sharing, but their resolution is well beyond the scope of this paper.

The repository also contains information about the individual tests that can be performed and the constraints on those tests. Constraints include:

- when a test plan will be completed
- when a range is available
- how many test points of any given type can be executed in a single mission

Furthermore, the repository contains information about the cost structure of different tests (it is often the case that the first test point costs hundreds of thousands of dollars, while any further points conducted in that mission are "free"). If it has already been set, the test program's allotted budget and schedule are also recorded in this repository.

3.2 The PATFrame Algorithm

Given the unpredictable nature of testing, attempting to create a definitive plan for a multi-year test effort is foolish. Even if we could examine all possible permutations for a test schedule to find the "best" according to a given set of criteria, it would be likely to be OBE in just a few weeks or months. Our algorithm addresses the scheduling aspect of a test plan. Accordingly, we make our first two assumptions:

A1.) Schedule is our dominant constraint. Cost of individual tests is not considered in this model since we assume we are more likely to run out of time than money.

A2.) A solution that defines an optimal start (on the order of weeks) to our test schedule is sufficient.

This is fortunate from the standpoint of simplicity. Solving scheduling problems is a well characterized as NP-Hard, and this model is no

exception. NP-Hard problems are those for which the computational load scales in proportion to the factorial of a given input variable. The most common example of an NP-Hard problem is the "Traveling Salesman Problem" in which a salesman must determine which path through *N* cities is the most efficient. When brute-forcing a solution, we need to consider the possibility of starting with any of the *N* cities. For each of these possible starts, we must consider visiting the *N*-1 remaining cities next, then for each of these, the *N*-2 which remain and so on. For *N*=4, we need only consider 24 possible paths, for 10 cities, the number of paths exceeds 3.6 million, and for 30 cities, there are several quintillion paths (over 10^{32}).

Considering only a moderate amount of MOPs, possible tests and resources over a span of only days to weeks creates a staggeringly large problem space.

Those familiar with the "nurse rostering problem" (NRP) and "Resource-Constrained Project Scheduling Problems" (RCPSP) will note a resemblance between the constraints and solution space of those problems and ours [4], [18].

In order to find an optimal solution to a scheduling problem, a number of methods can be employed. Brute force is an option, albeit an infeasible one, for an NP-Hard problem unless we are working in a very simple problem space. There are, however, a number of mathematical programming, heuristic, meta-heuristic, and artificial intelligence methods that have been successfully applied to similar problems like NRP and RCPSP [1], [11]. While many of these methods of optimization may not work as efficiently or even at all in our scenario, we believe that at least some may be sufficient to find an optimal (or near optimal) solution in a feasible computation period. Selecting a suitable scheduling method is left for our future work.

In order for any of the aforementioned methods to look for a "best" schedule, we must decide how we will value each test. This leads to our third assumption:

A3.) The sole purpose of testing is to determine whether a SUT has passed or failed each MOP. The value of a test or test point is wholly determined by its ability to reduce uncertainty to this end.

As our work progresses, we will likely work beyond the narrow scope of this assumption. In light of A2, we present four heuristics for valuing individual tests. These heuristics are a product of experience in T&E and interviews with testers from different organizations across the DOD [12], [13].

H1.) Regardless of "estimated" certainty of passing any given MOP, all MOPs should be demonstrated if at all possible. This means that even if our estimate is a 99% chance of passing a MOP, we must conduct some minimal testing during our program (which may extend beyond our "local" window) to verify our estimate is not wildly invalid.

H2.) Reduction in uncertainty is valuable until the threshold value is reached, and not beyond (once we reach our desired level of confidence, we should stop).

H3.) Test A is always more valuable than test B if: (1) both test A and B are for unverified MOPs or both are for verified MOPs and (2) A is expected to reduce its MOP's uncertainty by a larger numerical value and (3) test A's MOP is equally as or more important than test B's MOP.

H4.) If presented with a choice between the two, a MOP that is unverified (not tested at all) should be verified before increasing confidence on a verified MOP of the same (or lesser) importance.

Our strategy can be succinctly described as "tests most likely to reveal important deficiencies should have a higher priority." A keen observer will note that the heuristics above allow us to create a cardinal ordering of the value of independent tests, but not a discrete value. A discrete value will be required to implement the models described above. We have not yet defined a method for calculating the discrete value of a test, and leave this for our future work.

4 Extensibility of PATFrame

Certainly, the core of PATFrame doesn't address all of the questions posed in the introduction. Improving on this core, however, can increase the utility of PATFrame. Once we reach a more mature design for the core, we see the potential for even more valuable improvements through extending the model.

A number of possible enhancements are proposed below.

4.1 IBM Defect Model

The aforementioned IBM whitepaper proposes tracking defects found in a software test program and using this data to measure testing progress. The failure discovery rate described in the IBM whitepaper decays over time and is typical of software. Understandably, failures in different domains will have different patterns. Therefore, merging all deficiency reports from all disciplines and all tests into one unit for analysis will be less than helpful. By tracking each deficiency report (including associated discipline, and the sortie during which it was found) as it is generated, test managers can track the progress of particular disciplines and determine which may be in need of more attention.

This information can be used not only to determine whether or not the user is "done testing" but as the program matures, it can be used to make an educated and defensible estimate of the "right" amount of testing for each domain. That is, the amount of testing which will find n failures where the cost of finding the $(n+1)^{th}$ failure is greater than the benefit of finding that failure.

4.2 Design of Experiments and Adaptive Statistical Methods

Design of Experiments (DOE) has recently made its way into the DOD test community. We propose including it in PATFrame for two reasons. First, the re-planning enabled by PATFrame will allow testers to better utilize adaptive techniques, like sequential analysis, which are virtually unheard of among DOD testers. Second, in order to re-plan quickly and without excessive operator input, PATFrame will need to be able to conduct some of the computational work to determine how many (and which) test points are expected to generate the most value.

PATFrame may implement any or all of the following capabilities. (1) The ability to determine how many test points should be scheduled for each MOP or even per sortie (setting *n*). This is part of basic DOE, but will speed the planning process. (2) A feature to select appropriate DOE strategies for highlighting the relevant failure modes of a particular system. Possible DOE strategies include:

- Adaptive Random Testing (ART) – ART is best suited to finding clusters of failures [5].
- Latin Hypercube Sampling (LHS) – LHS is a method for sampling each dependent variable state exactly once [10]. It only covers a very small portion of a complex problem space.
- Adaptive One-Factor-at-a-Time (OFAT) [7] – Adaptive OFAT is well suited for determining a "local maximum" with respect to a number of dependent variables, and is particularly useful when there are no interaction effects between variables.
- Sequential Analysis may be used in situations where the number of trials *n* that will be required is highly uncertain, such as when the independent variable has an unknown variability. This will permit testers to use fewer test points in some cases [6], [14].

4.3 Test Plan Simulation

Testing is an uncertain endeavor, and evaluating every possible contingency is impossible for a human. The use of computer simulations to model thousands of possible outcomes could be very useful. User input and even historical data from similar test programs can inform PATFrame and help evaluate the effects of discovering critical deficiencies which lead to regression testing, failure to complete certain tests on time, unavailability of resources, and other possible contingencies.

This has three readily apparent uses. First, it can be used to estimate the likelihood that a particular resource will be used on a particular date. Testers can use this information to determine whether or not it is worth it to book expensive time on DOD test ranges months in advance. Second, testers can use these simulations to determine the likelihood that a test program can accomplish its objectives within particular budget and schedule constraints. Third, simulations can serve as a sensitivity analysis, informing the user of the most critical events and resources.

By simulating different outcomes, PATFrame can recommend opportunities for more robust test planning.

5 Future Work and Implications

Our framework is by no means complete. While there are myriad extensions that could be developed to PATFrame, our primary focus now is on the core functionality. Finding and developing a suitable scheduling function for a scenario with so many complicated constraints will be a primary focus.

Our other focus will be evaluating, and if necessary, improving our heuristics for valuing individual tests. Humans can naturally rank the importance of competing MOPs and requirements, but have trouble consistently assigning a discrete value to these items. "Translating" user intent from cardinal or other input methods to a discrete value will be important to PATFrame's functionality.

6 Conclusion

Our proposed method addresses a number of important SoS testing challenges by enabling users to adapt their test plans in light of newly learned information. The ability to re-plan test programs has already shown benefits through exploratory testing, and we believe that our algorithm-driven method for adaptation may provide similar results.

7 Acknowledgement

This material is based upon work supported by the Department of Defense, United States Army, White Sands Missile Range, NM Under Contract No. W9124Q-09-P-0230. Any opinions, findings and conclusions or recommendations, expressed in this material are those of the authors(s) and do not necessarily reflect the views of the Department of Defense, United States Army, While Sands Missile Range, NM.

References

[1] Aickelin, Uwe, and Kathryn A. Dowsland. "An Indirect Genetic Algorithm for a Nurse Scheduling Problem." *Computers and Operations Research* 31.5 (2004): 761-78. Print.

[2] Bach, James. *Exploratory Testing Explained*. 16 Apr. 2003. Web. 7 Oct. 2009. <http://www.reference.com/go/http://www.satisfice.com/articles/et-article.pdf>.

[3] Bell, David E, Howard Raiffa, and Amos Tversky. *Decision Making: Descriptive, Normative, and Prescriptive Interactions*. Cambridge: Cambridge University Press, 1988.

[4] Cheang, B., H. Li, A. Lim, and B. Rodrigues. "Nurse Rostering Problems - A Bibliographic Survey." *European Journal of Operational Research* 151 (2003): 447-60. Print.

[5] Chen, Tsong Y., Fie-Ching Kuo, Robert G. Merkel, and T. H. Tse. "Adaptive Random Testing: The ART of Test Case Diversity." *The Journal of Systems and Software* 83 (2009): 60-66. Print.

[6] Chernoff, Herman. "Sequential Design of Experiments." *The Annals of Mathematical Statistics* 30.3 (1959): 755-70. Print.

[7] Frey, Daniel D., and Hungjen Wang. "Adaptive One-Factor-at-a-Time Experimentation and Expected Value of Improvement." *Technometrics* 48.3 (2006): 418-31. Print.

[8] Hubbard, Douglas W. How to Measure Anything: Finding the Value of "intangibles" in Business. Hoboken, N.J.: John Wiley & Sons, 2007. Print.

[9] Macias, Fil. 2008. The Test and Evaluation of Unmanned and Autonomous Systems. *ITEA Journal* 29: 388-395.

[10] McKay, M. D., R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code." *Technometrics* 21.2 (1979): 239-45. Print.

[11] Merkle, Daniel, Martin Middendorf, and Hartmut Schmeck. "Ant Colony Optimization for Resource-Constrained Project Scheduling." *IEEE Transactions on Evolutionary Computation* 6.4 (2002): 333-46. Print.

[12] *PATFrame El Paso ITEA Conference Trip Report*. Rep. 8 March 2010. Web. <http://mit.edu/patframe>.

[13] *PATFrame Ft. Hood Trip Report*. Rep. 1 Sept. 2009. Web. <http://mit.edu/patframe>.

[14] Robbins, Herbert. "Some Aspects of the Sequential Design of Experiments." *Bull. American Math Society* 58.5 (1952): 527-35. Print.

[15] Simon, Herbert A. Models of Man: Social and Rational; Mathematical Essays on Rational Human Behavior in Society Setting. New York: Wiley, 1957. Print.

[16] Singer, P. W. *Wired for War: the Robotics Revolution and Conflict in the Twenty-first Century*. New York: Penguin, 2009. Print.

[17] Streilein, James J. "Test and Evaluation of Highly Complex Systems." *ITEA Journal* 30.1 (2009). Print.

[18] Yang, Bibo, Joseph Geunes, and William J. O'Brien. *Resource-Constrained Project Scheduling: Past Work and New Directions*. NSF. Web. 6 Mar. 2010.

[19] Valerdi, R., Ross, A.M., and Rhodes, D.H. 2007. A Framework for Evolving System of Systems Engineering. *Crosstalk*. October 2007, pp. 28-30.

[20] von Winterfeldt, D. A Re-examination of the Normative-Descriptive Distinction in Decision Analysis. Annals of Operations Research 19, 499-502, 1989.