International Journal of Pure and Applied Mathematics
Volume 120 No. 5 2018, 1225-1239
ISSN: 1314-3395 (on-line version)
url: http://www.acadpubl.eu/hub/
Special Issue

http://www.acadpubl.eu/hub/

USAGE OF SOFTWARE TEST BEDS FOR TESTING THE NEW TECHNOLOGIES

K. Baalaji* and Dr.V.Khanaa**

**Research Scholar Dept of CSE, Bharath University.*

***Dean - Information Technology, Dept of IT, Bharath University.*

Abstract

The advancement of another innovation relies on a decent hypothetical premise for building up the innovation, and in addition upon its exploratory acceptance. Keeping in mind the end goal to accommodate this experimentation, we have researched the production of a product proving ground and the plausibility of utilizing the same proving ground for exploring different avenues regarding an expansive arrangement of advances. The proving ground is an arrangement of projects, information, and supporting documentation that permits specialists to test their new innovation on a standard programming stage. An imperative segment of this proving ground is the Unified Model of Dependability (UMD), which was utilized to evoke trustworthiness necessities for the proving ground programming. With a gathering of seeded blames and known issues of the objective framework, we have the capacity to figure out whether another innovation is capable at revealing abandons or giving different guides proposed by its engineers. In this paper, we display the Tactical Separation Assisted Flight Environment (TSAFE) proving ground environment for which we displayed and assessed constancy prerequisites and characterized shortcomings to be seeded for experimentation. We depict two finished analyses that we directed on the proving ground. The initially trial concentrates on an innovation that distinguishes design infringement and assesses its capacity to identify the infringement. The second trial studies model checking as a feature of outline for confirmation.

Key words: *Software, UMD, TSAFE ,HDCP*

Introduction

The advancement of another innovation relies on a decent hypothetical premise for creating the innovation and in addition upon its test acceptance.

Programming building has no such settled convention for test acceptance. With a specific end goal to redress this restriction, we researched the production of a product test bed that is helpful for such experimentation with programming innovations (i.e., improvement and quality certification instruments, systems and techniques). The test bed is an arrangement of projects, information, and supporting documentation that will permit scientists to test their new advancements on a standard programming stage. With an accumulation of seeded surrenders and known issues of the objective framework, we have the capacity to figure out whether another innovation is skilled at revealing deserts or giving different guides proposed by its engineers. Making state sanctioned testbeds would permit more formal acceptances of large portions of the advances now proposed for development of programming improvement rehearses. This paper, which is an augmentation of a workshop paper , talks about a specific testbed. Keeping in mind the end goal to test its helpfulness, we needed to apply it to a specific programming advancement issue, however from the point of view of distinctive advances and researching the attainability of "reusing" the same testbed for diverse trials. We center in this paper on advances for accomplishing programming trustworthiness, yet the idea is effortlessly extendable to some other ascribe one wishes to address. We explored programming trustworthiness as a feature of the NASA's High Dependability Processing Program (HDCP), a helpful exploration assention in the middle of NASA and different colleges and examination focuses. The HDCP task explored accomplishing high steadfastness by presenting new advancements created by the partaking examination accomplices.

2 TSAFE INTRODUCTION

A key strategy for the HDCP initiative was to accelerate adoption of new software engineering technologies by evaluating them on testbeds representative of NASA software. One such testbed is TSAFE, a component of a proposed Automated Air Traffic Control system. TSAFE was defined by Erzberger at NASA Ames Research Center, implemented as a prototype by Dennis at

MIT, and then instrumented and packaged for experimentation by University of Maryland and Fraunhofer Center

Maryland as described in Section 4. The US Air Traffic Control (ATC) system, consisting of a large network of people and equipment that monitor and direct aircraft, is a critical infrastructure that manages more than 30,000 commercial flights to move 2,000,000 passengers safely each day . The main goal of the system is to keep a safe distance between the aircrafts while achieving efficient air traffic movement in order to minimize delays. The proposed Automated ATC (AATC) software system encompasses all the characteristics that make establishing dependability a challenge, such as distributed computation, concurrency, safety critical functionality, communication protocols, sensitive data and user interaction. Hence, the AATC system raises a large set of implementation issues that propagate to many of its system components.

Figure 1. THE TSAFE DISPLAY SHOWING THREE PLANES FLYING OVER MARYLAND

TSAFE is a product segment without bounds AATC intended to help air activity controllers in recognizing and determining transient clashes between flying machines. At present, air activity controllers keep up surveying so as to fly machine detachment radar information for potential clashes what's more, issuing clearances to pilots to change their directions as needs be. Under the

current framework, just piece of the airspace ability is misused. Abusing the full airspace limit requires the new Automated Airspace Concept to be actualized, which implies that computerized components assume an essential part in keeping up air ship division. The part of TSAFE is to go about as a dependable free security net from inescapable blemishes in this proposed framework. Its point is to identify clashes somewhere around 2 and 7 min later on and, in its full usage, to issue shirking moves as needs be.

3. Modeling TSAFE Dependability

The International Federation for Information Processing characterizes steadfastness as the reliability of a registering framework that permits dependence to be legitimately put on the administrations it conveys. "Dependence" is relevantly subjective and relies on upon the specific partners' needs. Diverse partners will concentrate on distinctive frameworks properties, e.g., accessibility, capacity to maintain a strategic distance from disastrous disappointments, and aversion of conscious interruptions, too as on distinctive levels of adherence to such qualities. Likewise, the same quality can mean distinctive things to diverse individuals, and it is basic to discover different definitions for the same trait .UMD has been executed as an electronic apparatus sorted out around two tables: the Table "Degree" (Fig. 3), which permits partners

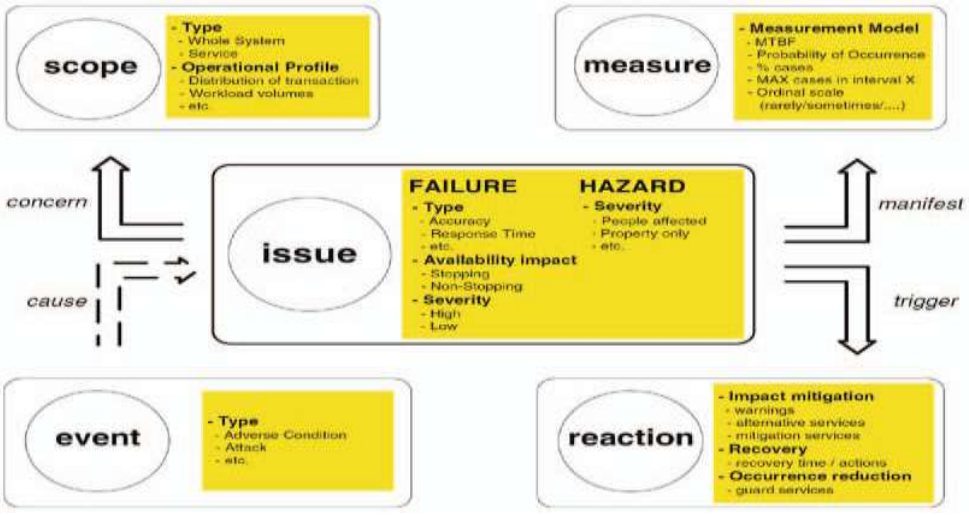


Figure 2. The dependability modeling framework

to portray every one of the framework's administrations for which constancy could be of concern; and the Table "Issue" (Fig. 4), which permits clients to indicate their trustworthiness needs by characterizing, for the entire framework or a particular administration (chose from the "Degree" table), the potential issues (disappointments or perils), their mediocre appearances, the conceivable activating outside occasions, and the craved responses.

3.1 Applying UMD to Identify the Dependability Requirements for TSAFE

We utilized UMD to characterize the constancy necessities for TSAFE. A little gathering of software engineering scientists and understudies went about as partners (particularly as air activity controllers), in the wake of being given a short prologue to TSAFE and its reasons, while one individual went about as an investigator Scope definition All partners, cooperating and bolstered by the investigator, chose from the useful prerequisites accessible for TSAFE the administrations for which they trusted trustworthiness could be significant. The distinguished administrations are portrayed in the extension table (Fig. 3). Necessities elicitation and displaying Each partner, bolstered by the investigator and guided by the structure gave by the apparatus, filled the same number of tables as important to characterize her/his reliability needs (Fig. 4). The portrayals of the UMD ideas of degree, issue, occasion, measure, and response gave helpful direction to partners. Every partner utilized the portrayals effectively accessible, or, at whatever point fundamental, expanded it with his/her own particular definitions.

Figure 3. "Degree" of the UMD model for TSAFE, which is utilized to recognize pertinent administrations for the product under thought

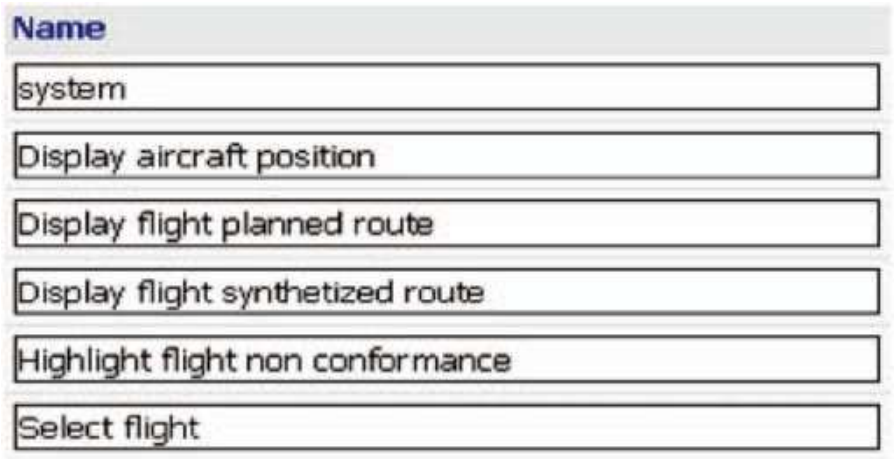
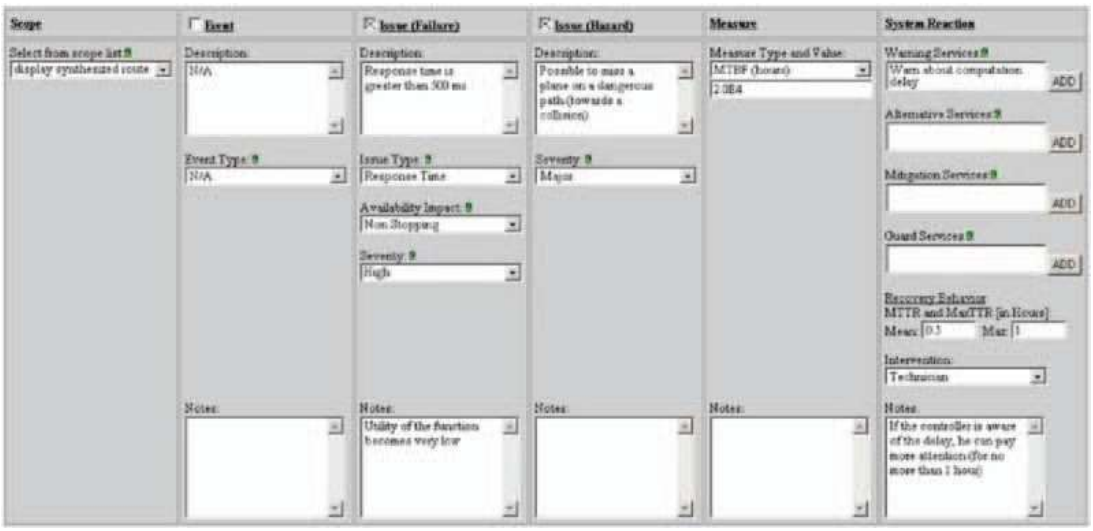


Figure 4. UMD Tool—TSAFE issues not related to an external event



he/she could miss detecting a plane on a perilous way. This could prompt a crisis circumstance and potentially cause high weight on the cockpit team, required to perform sudden break moves by the transient clash shirking frameworks. The partner distinguishes this disappointment as an exceedingly basic one, driving the investigator to propose MTBF of 2*104 (between the qualities proposed for high and mission basic accessibility in Table 1). All together to be more positive about the framework, the partner presents a notice administration that will prompt on the off chance that computational time gets to be more prominent than 500 ms, cautioning him

when consideration is required. At long last, the partner characterizes the recuperation to be performed inside of one hour. In the event that this disappointment condition keeps going more than 1 h, he would be not able to perform his obligations because of the need to keep up a higher than common level of consideration.

4 Building the Test bed Environment

This segment talks about why and how the proving ground was constructed.

4.1 Goals for the Test bed Environment

So as to actualize the trustworthiness necessities for TSAFE that were evoked with support from UMD, there are a few noteworthy classes of activities, for example, identifying and killing the issues that can prompt disappointments, or, if disappointments can't be anticipated, then guaranteeing fitting recuperation inside sought time. This is incompletely because of a taking after's portion elements:

• Empirical studies are costly, bringing about generally few studies being completed. Conveying out observational work is intricate and tedious; this is particularly genuine for programming building. Not at all like assembling, we don't manufacture the same item, again and again, to meet a specific arrangement of details. Programming is created furthermore, every item is unique in relation to the last. Along these lines, programming ancient rarities don't give a expansive arrangement of information focuses allowing adequate factual force for affirming or dismissing theories. Also, human elements tend to expand the expense of experimentation (the most applicable subjects for most studies originate from the populace of expert programming designers, whose time is quite often over-booked also, profoundly costly) making it more hard to accomplish measurable hugeness. We have seen that creating reusable programming testbeds have relieved the above troubles, in any event when those testbeds are illustrative of genuine programming improvement extends and have fitting documentation.

4.2 Turning TSAFE into a Testbed

The TSAFE model, whereupon we based the testbed, was at first created at MIT and is a 20,000 lines of Java program that performs two essential capacities:

*conformance observing and direction blend.

*The experience from these innovation tests and additionally input and lessons educated have been gathered and are furnished together with alternate curios as a component of the testbed keeping in mind the end goal to boost the convenience and also to minimize the expense and exertion of experimentation.

*Presently, the accompanying antiques are accessible: a necessities particular, construction modeling

documentation, source code, an establishment aide, and some recorded flight information that serve as test data. An apparatus to make fake test information is additionally accessible.

*Deficiency seeding We verified that, for experimentation, an arrangement of flaws should have been be incorporated for infusion in the source code. We recognized three sorts of issues that speak to

designer lapses and in this manner constitute conceivable shortcoming classes: Technology-driven, *Dependability- driven, and History-driven.

* Dependability-driven flaws are blames that will bring about run-time disappointments, if activated. These shortcomings can be gotten from steadfastness models or prerequisites of the framework.

The shortcomings could possibly be perceptible by a specific innovation, and may on the other hand may not speak to recorded deficiencies for a specific framework.

4.3 TSAFE Instrumentation

The testbed engineers instrumented TSAFE as a testbed for outlining and executing the accompanying steadfastness related test exercises:

• Define what steadfastness implies for TSAFE, by applying the UMD model.

• According to this meaning of steadfastness, distinguish potential disappointments and

relating shortcomings in the code that could bring about these disappointments.

• Identify experiments that would trigger these deficiencies and reason disappointments.

• Seed the code with the recognized flaws.

For every trial, the innovation designers (or somebody talented in utilizing the innovation) planned their speculations as far as blames that can be identified by their innovation (also,

conceivably the effect on steadfastness) and assessed the expenses related with applying the innovation. They then:

• Applied the innovation on the code containing seeded blames and recorded the distinguished flaws and also the related discovery cost.
• Exercised experiments on the code containing the seeded blames and recorded the happening disappointments (and their recurrence of event).
• Analyzed these deficiencies and disappointments and accepted or disproved the innovation speculations.

5 Software Architecture Evaluation Method

A significant number of the innovations examined in HDCP manage the structural planning of a product framework, which is one of the significant reasons we chose to begin trying different things with this class of advances. Practicality is one of the reliability traits (Laprie 1992)
what's more, structural planning assessment accept that if the execution adjusts to the arranged structural engineering, then the product is less demanding to keep up.
Setup of testbed The testbed designers initially arranged an arrangement of principles depicting properties the construction modeling of TSAFE ought to have. In request to seed this shortcoming, nine changes (erasures, augmentations, and changes of nine lines of code) to four unique classes were connected. The static perspective of the change that speaks to issue 11.1.1 is outlined in Fig. 5 and was built utilizing the SAVE apparatus .
This required a two-stage process:

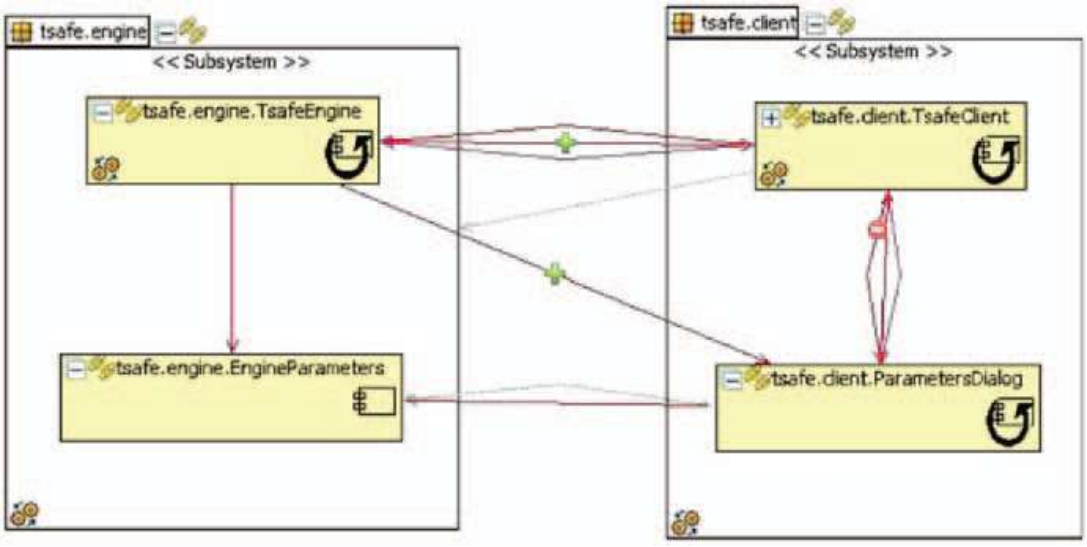


Figure 5. Zooming in on the engine and client components highlighting added (+) and deleted (-) connections.

(1) The issues reported by the subject were checked against the list of seeded rule violations. This required some subjectivity since the subject was not expected to independently formulate the same wording as was used to describe the seeded violations; the testbed developers had to assess whether the underlying issue described was the same as the item on the seeded list.

.

(2) The rule violations were mapped to specific faults, by which we mean specific instances within the code that would affect system flexibility or maintainability that were directly caused by the rule violations. Along with the list of discrepancies found, the subject was required to report for each

discrepancy the following information:

1. Which rule was being violated?

2. What part of the architecture was being examined when the discrepancy was found?

This information was used to understand the process followed by the subject as well as to help improve process conformance (relative to the process provided to the subject as part of the experiment), since it was reasoned that making up plausible answers to those questions would

require more effort from the subject than simply applying the SAE documented process and reporting the results according to the provided form and instructions.

Results: The results of the study were useful for the evolution of the SAE method:

• SAE found six of the seven expected (relative to the claims of the technology) seeded rule violations.

• The six seeded rule violations found by SAE were considered to map to three out of the four seeded faults. Fault 11.1.1 was among the detected seeded faults.

• The one seeded rule violation that was missed helped identify a bug in the tool that has since been fixed. Thus, the bug in the tool caused one (fault 7.1.1) of the four seeded faults to be missed.

6. Software Testing (Ongoing Study)

We are expanding the testbed to allow for experiments using standardized execution based technologies, i.e., software testing. During software testing, the software's behavior is studied by executing test cases, which represent controlled input. The software's execution is checked against an expected behavior and the software's specifications and test coverage criteria are used to drive the test-case generation process.

In the future, we will apply the same process to other, less understood execution-based technologies. Second, since software testing has been widely studied, the results of experiments on testing will give us a baseline technology, which we can use to compare other technologies.

Consequently, the test pool (which currently contains 121 test cases) has the following characteristics:

1. Each statement in the TSAFE application is covered by at least 30 test cases.
2. Each branch is covered by at least 30 test cases.
3. If the test case was developed using some requirements specification, we provide traceability to the particular requirement that was being tested.

7. Summary, Discussion, and Future Work

Testbeds have proven to be an effective vehicle for testing new technologies. We previously demonstrated that one testbed could be used to study several technologies that detect similar dependability issues . In this paper, we studied whether one testbed could be used in experiments

studying different technologies that detect different kinds of dependability issues. Our conclusion is that even though these three experiments are very different in terms of the technology that was the subject for experimentation, it was feasible to apply them to the same testbed.In addition, we do not suggest a broad development of testbeds by each organization that wishes to conduct technology experiments. Our vision is instead that a few independent organizations are funded to develop and maintain testbeds as a service to technology developers allowing technologies to be studied outside of the inventor's laboratory before applying them to real situations.

This will keep the cost of experimentation down and will allow for coordination and experience sharing between all stakeholders in an efficient manner. Our future work is to enhance the testbed by synthesizing and seeding more faults that would make the testbed more interesting for other families of technologies. We will also run several additional experiments, analyze, and interpret the results in order to build a selection of examples and baselines that can be reused by technology developers who want to design and run their own experiments on the testbed. If necessary, we will then evolve the experiment, the testbed and the technology under investigation. Based on these results and the explicit links from faults to failures, we will be able to reason about the impact of technologies on dependability.

References

1.Erzberger H (2004) Transforming the NAS: the next generation air traffic control system. In: 24th International
Congress of the Aeronautical Sciences, Yokohama, Japan

2.Godfrey MW, Lee EHS (2000) Secrets from the Monster: extracting Mozilla's software architecture.
In: Proc 2nd symp. constructing software engineering tools (CoSET00), Limerick, Ireland, June.
ACM Press, New York

3.Huynh D, Zelkowitz MV, Basili VR, Rus I (2003) Modeling dependability for a diverse set of stakeholders
(Fast abstracts), Distributed Systems and Networks, San Francisco, CA, June, B6–B7
International Federation for Information Processing (IFIP WG-10.4) http://www.dependability.org

Laprie J-C (1992) Dependability: basic concepts and terminology, dependable computing and fault
tolerance. Springer-Verlag, Vienna, Austria
4.Betin-Can A, Bultan T, Lindvall M, Lux B, Topp S (2005) Application of design for verification with
concurrency controllers to air traffic control software. In: Proceedings of 20th IEEE/ACM inter438
national conference on automated software engineering, pp 14–23
5.Boehm B, Huang L, Jain A, Madachy R (2003) The nature of information system dependability—a
stakeholder/value approach. USC Technical Report
6.Brat G, Havelund K, Park S, Visser W (2000) Java PathFinder—a second generation of a Java model
checker. In: Proceedings of the workshop on advances in verification, July 2000, Chicago, USA
7.Brooks FP (1995) The mythical man-month. Addison Wesley, Reading, MA
Bultan T, Yavuz-Kahveci T (2001) Action language verifier. In: Proc. 16th IEEE international conference
on automated software engineering, San Diego, USA, 382–386
8.Dennis G (2003) TSAFE: building a trusted computing base for air traffic control software. Masters
Thesis, Computer Science Dept., Massachusetts Inst. Technology
9.Donzelli P, Basili V (2006) A practical framework for eliciting and modeling system dependability
requirements: experience from the NASA high dependability computing project. J Syst Softw
79(1):107–119
10.Erzberger H (2001) The automated airspace concept. In: 4th USA/Europe air traffic management
R&D seminar, Santa Fe, New-Mexico, USA
11.Asgari S, Basili V, Costa P, Donzelli P, Hochstein L, Lindvall M, Rus I, Shull I, Tvedt R, Zelkowitz
M (2004) Empirical-based estimation of the effect on software dependability of a technique for

architecture conformance verification, ICSE/DSN 2004 twin workshop on architecting dependable
systems (WADS 2004), Edinburgh, Scotland
12.Basili V, Donzelli P, Asgari S (2004) A unified model of dependability: capturing dependability in
context. IEEE Softw 21(6):19–25
13.Bhansali S, Nii HP (1992) Software design by reusing architectures. Proceedings of the Seventh
Knowledge-Based Software Engineering Conference, McLean, Virginia, USA, pp 100–109
14.Lehman MM (1996) Laws of software evolution revisited. In: European Workshop Software Process
Technology, Nancy, France, October 1996. Springer, Berlin Heidelberg New York, pp 108–124
15.Lehman MM, Belady LA (1985) Program evolution: processes of software change. Harcourt Brace
Jovanovich, London.