13845

OPSEC REVIEW CERTIFICATION

(AR 530-1, Operations Security)

I am aware that there is foreign intelligence interest in open source publications. I have sufficient technical expertise in the subject matter of this paper to make a determination that the net benefit of this public release outweighs any potential damage.

Reviewer: Mark Slominski GS-14 TEAM LEADER
Name Grade Title Uses

Mark Slominski 1/15/03
Signature Date

Description of Information Reviewed:

Title: HEALTH MONITORING AND DIAGNOSTICS OF GROUND COMBAT VEHICLES.

Author/Originator(s): ELENA BANKOWSKI, CHRISTOPHER MILES

Publication/Presentation/Release Date: 03/02/2003

Purpose of Release: HEALTH AND DIAGNOSTICS CONFERENCE, SPIE.

An abstract, summary, or copy of the information reviewed is available for review.

Reviewer's Determination (circle one):

(1.) Unclassified Unlimited.

2. Unclassified Limited, Dissemination Restrictions IAW _____

3. Classified. Cannot be released, and requires classification and control at the level of _____.

Security Office (AMSTA-CS-S):

(Concur)/Nonconcur _____ 24 JAN 03
Signature Date

Public Affairs Office (AMSTA-CS-CT):

(Concur)/Nonconcur _____ 24 JAN 03
Signature Date

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	03/04/03	Conference Proceedings, 2003 SPIE

4. TITLE AND SUBTITLE

Health monitoring and diagnostics of ground combat vehicles

5. FUNDING NUMBERS

6. AUTHOR(S)

Elena Bankowski, Christopher Miles, Michael S. Saboe
US ARMY TACOM, Warren, Michigan 48397-5000
Peggy Gilbert, Q-Labs, 6301 Ivy Lane, Greenbelt, MD 20770

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

US ARMY TACOM
Warren, MI 48397-5000

Q-Labs, 6301 Ivy Lane,
Greenbelt, MD 20770

8. PERFORMING ORGANIZATION REPORT NUMBER

13845

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The paper was presented at the SPIE Symposium: NDE for Health Monitoring and Diagnostics, 2-6 March 2003, San Diego, California, USA

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Unlimited, approved for public release.

12b. DISTRIBUTION CODE

A

13. ABSTRACT *(Maximum 200 Words)*

The proposed technology is the Dependable Automated Reconfigurable Software (DARTS). The DARTS health and situation control continually tests the processing elements with Probe/Agent technology. Algorithms within the Health & Situation Control assess the health of the processors based on a criticality scoring system that considers mission requirements. Probes launched by the DARTS Controller query processing elements. The probed data is sent to a gauge that has a variable sensitivity or gain. Statistical Usage models and criticality scoring control the sensitivity of the gauge. In response to the gauge, the replicating process launches agents that can insert anomalous events for diagnostic purposes. In this context, a probe is a subset of an agent having only the abiiity to query without affecting framework, I/O protocol or Quality of Service. Each weapon system fitted with a DARTS Controller will control self-repair and reconfiguration of on-board processors utilizing a statistical based intelligent scoring system. It considers criticality of the function in the current battlefield situation. DARTS is a software system that enhances the performance of a weapon system by providing on-the-fly reconfiguration to accommodate the loss or malfunction of processing elements or to optimize onboard performance capability.

14. SUBJECT TERMS

15. NUMBER OF PAGES

8

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

Health monitoring and diagnostics of ground combat vehicles

Elena Bankowski, Christopher Miles, Michael S. Saboe
US ARMY TACOM, Warren, Michigan 48397-5000
Peggy Gilbert, Q-Labs, 6301 Ivy Lane, Greenbelt, MD 20770

ABSTRACT

The proposed technology is the Dependable Automated Reconfigurable Software (DARTS). The DARTS health and situation control continually tests the processing elements with Probe/Agent technology. Algorithms within the Health & Situation Control assess the health of the processors based on a criticality scoring system that considers mission requirements. Probes launched by the DARTS Controller query processing elements. The probed data is sent to a gauge that has a variable sensitivity or gain. Statistical Usage models and criticality scoring control the sensitivity of the gauge. In response to the gauge, the replicating process launches agents that can insert anomalous events for diagnostic purposes. In this context, a probe is a subset of an agent having only the ability to query without affecting framework, I/O protocol or Quality of Service. Each weapon system fitted with a DARTS Controller will control self-repair and reconfiguration of on-board processors utilizing a statistical based intelligent scoring system. It considers criticality of the function in the current battlefield situation. DARTS is a software system that enhances the performance of a weapon system by providing on-the-fly reconfiguration to accommodate the loss or malfunction of processing elements or to optimize onboard performance capability.

Keywords: Health monitoring, diagnostics, prognostics.

1. INTRODUCTION

Today's Main Battle Tanks (MBT) contains a multitude of processors yet systems such as the Abrams provide redundancy only between the hull electronics unit and the turret electronics unit. The Abrams employs duplicate processors hosting redundant software in different vehicle compartments. Over a half a million lines of software code span multiple processors. Future weapon platforms will host a significant increase in software. The processing burden of the front line vehicles will require a further increase in processing capability. Next generation weapon systems processing requirements will grow with the incorporation of intelligent decision aids, sensor fusion, and advanced communications. A future system will have 2×10^6 more configuration combinations than today's MBT. Cost, reliability, space, and mission requirements will preclude achieving redundancy with dedicated, embedded processors that duplicate functionality. The next generation systems vision is that a collection of general-purpose processors connected to a common bus will be scattered throughout the vehicle and assigned dynamically to the various vehicle control and mission-specific tasks as required. This approach, shown in Figure 1, reduces cost and provides greater effective redundancy, since any healthy processor can be assigned to any task instead of having a limited number of backups of ever decreasing capability. Line Replaceable Units (LRU's) are: engine, displays, electronic control units, etc.

Next generation systems process requires extensive monitoring and analysis capabilities to track whether the weapon system is operating properly. A robust reconfiguration capability is required to gracefully and rapidly reorganize the assignment of tasks to processors to respond to hardware and software failures and to changed mission requirements (e.g., switching from surveillance mode to combat/engagement mode). The software in the Abrams tank is created in a highly sophisticated development and testing environment. In 1998, the U.S. Army Next Generation Software Engineering Technology Area (Next Generation) began implementing Statistical Usage Testing (SUT) in the Post Production Software Support (PPSS) project for the U.S. Army's main battle tank. The pilot project on the Abrams M1A2 had the following objectives:

- the determination of how SUT can improve M1A2 field reliability;
- the development of usage model(s) for the Driver's Integrated Display (DID);

- development and documentation of a tailored modeling process to allow for scale-up of SUT, including guidance on modeling practices;
- investigation of tie-in of SUT with existing PPSS testing and future research.

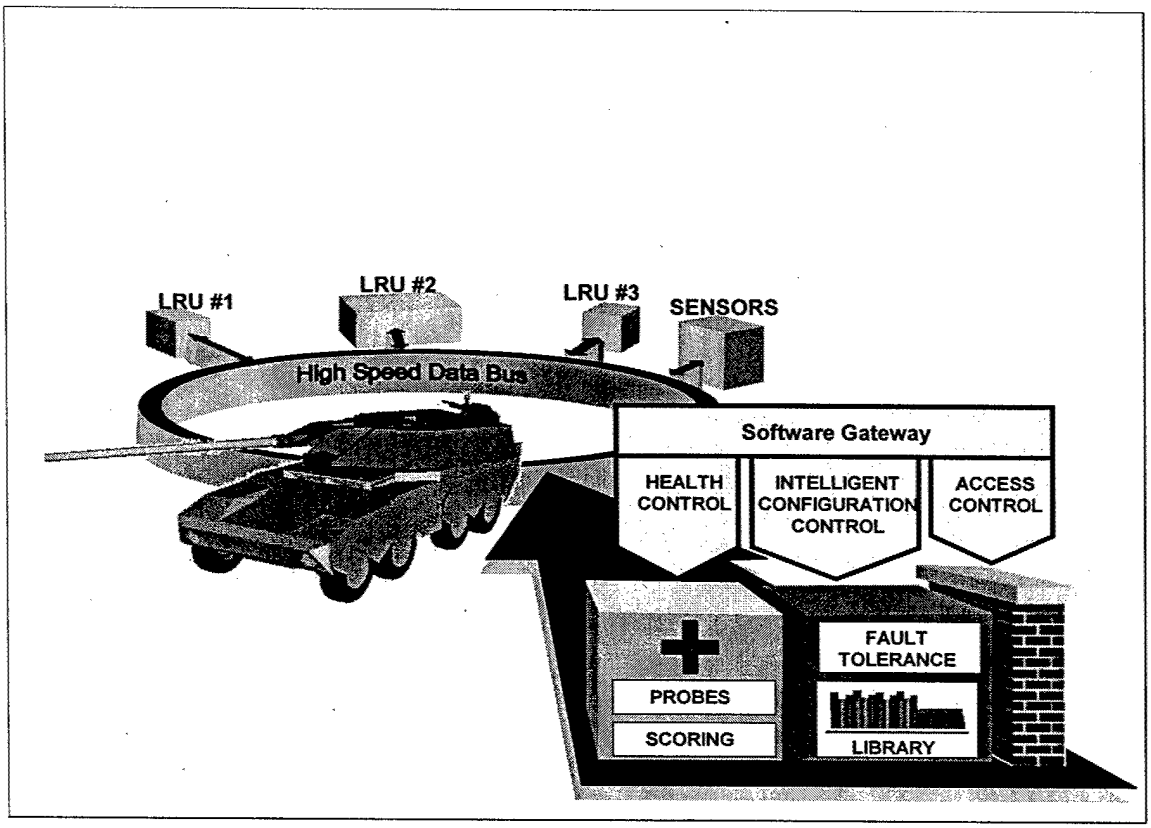

Figure 1. The Next Generation Dynamic Software Assembly.

Usage model development for the DID was facilitated by the CASE tool toolset Certify® which supported Markov chain usage model development, test management, test case generation and statistical testing. A prototype tool for composing top-level models and lower-level sub-models into one flattened toolset Certify® model was also used. This Model Compose utility allowed for the development of sub-models similar to subroutine development in programming. The major lessons learned from the SUT project were:

- SUT can positively impact PPSS testing, since the focus is on operational usage
- usage modeling is feasible in the M1A2 PPSS environment
- usage modeling uncovers a number of issues that relate to behavior and testing
- A logical and complementary relationship exists between the current testing approach used by Next Generation and SUT.

The method and results of a pilot project were discussed in a paper [1]. The SUT modeling techniques was applied to the Driver's Integrated Display (DID), a component of the soldier-machine interface of the tank. Since then, additional LRU's with increasing complexity have been modeled using SUT. In applying modeling techniques, a high degree of complexity was observed, consisting of the numbers of screens to be modeled and the amount of information that could impact a tester's next action. These challenges were overcome using some innovative approaches.

Next Generation investigated the feasibility of using SUT in the PDSS test environment. The primary motivation was the realization that there were not enough test assets, people or time to test the main battle tank software for each release. TACOM personnel completed training in SUT. Q-Labs had expertise and provided mentoring. Next Generation test staff and Q-labs developed practical techniques to link current black box, regression test procedures with SUT tools and models. Software Engineering Technology (SET, acquired by Q-Labs) delivered SUT training. The purpose of the experiment was to determine better ways to test increasingly complex systems and system of systems in the future. The paper [1] reported on work performed in 1998-1999. Significant progress has been made since that time. Efforts are now well underway to combine SUT with a number of other approaches and scale up the approach even further.

2. THE SUT METHOD

The SUT modeling process was proposed by the Q-Labs. The steps of the SUT modeling process were tailored for the Next Generation situation and for easier hand-off in the future. These steps are shown below:

1. Define testing goals – The goals to be defined include the boundaries of what will be tested, reliability goals, etc.
   Next Generation specific approach – The goal was to build models that test possible inputs consistent with operational use. Actual goals during testing were to find remaining errors to conduct sufficient testing, and to gain additional confidence in product quality. These goals appeared in the Test Plan.
2. Define a 'use' – The use was defined as the start and end of a test case. It related to some 'real-world' interpretation of usage, as the testing results are to be used to make inferences on future usage.
   Next Generation specific approach – Uses correspond to the tank going from off to on, back to off (Master Power Switch On to full shut down). However, due to the nature of the models, each represents independent sets of functionality, a use corresponds to the tank going from off to on, back to off where only a limited set of the possible inputs are available. Additional tests were created to address the situations where there was interaction between the functions. These tests ensured that the functionality was indeed independent. This also appeared in the Test Plan.
3. Build List of Stimuli – Build the list of all things that enter through the boundary defined in the previous step. The organization of the stimuli list was critical. Grouping the inputs by screen, protocol, or other factor enhancesd readability and usability.
   Next Generation specific approach – Stimuli were generally defined to be button pushes. In the cases where an input is a data value, two stimuli were recorded, one to handle values that are in the valid range, and the other to handle values that are outside the valid range. 'Exceptional' stimuli included time lags, for the situation where certain actions did not occur until a defined time period has passed. They were also used in the models as needed, and appeared in the stimuli list.
4. Define usage variables – All factors that modify the possibility/probability of inputs are listed here. Possible values for each factor are also enumerated. Typically this list was larger than the final list of usage variables, since some usage variables were abstracted away.
   Next Generation specific approach – Usage variables were Engine/Tank Status (7 values), CID Status (2 values), Light Status (8 values), Heater Status (5 values).
5. Define models to develop based upon usage variables and testing goals – The number of 'unique' models was defined here. Significant factors include the potential size of models, potential yield in testing variants in certain areas, etc.
   Next Generation specific approach – Models were created corresponding to each of the major areas of functionality. They consisted of Lights, Auxiliary, Navigation and Maintenance/Back-Up. This was the basis for the Model Dictionary. The major models were composed from the sub-models.
6. Decompose model(s) into sub-models – Defined a rule/approach by which sub-models were created. This was based upon screens, functionality, model distribution or other factors. While doing this, the rules by which lower level models were composed with higher level models were defined. The key was that all the models/sub-models were built consistently so that testing could go on with a single, smooth approach.
   Next Generation specific approach – Each menu was a different model. There were approximately 40 menus in the DID. They appeared in the Model Dictionary, and each selected model was a separate document.

7. Build 'single variable' state transition diagrams – For each usage variable state transition diagrams were built. These diagrams defined the manner in which that usage variable could have changed the state. It only included the stimuli that could have caused state changes for the usage variable. The benefit of such a diagram was that it allows a model builder to see 'globally' while working 'locally.' Creating a state model for a part of the software was done by synthesizing the behavior of single variable models, while adding unique inputs and actions for that particular section of the model. Next Generation specific approach – State diagrams have been created for the CID, Engine/Tank Status, Heater and Lights usage variables, and for the DID. The state transition diagrams were separate documents.
8. Build sub-models and models – The Models were built one at a time, using the Microsoft Word template. All modifications in the models, except for the assignment of probabilities, were done in Word. Models were created by working from the bottom up. Models had four types of states: Entry/Exit, Model States, Composition States, and Collector States. There was one Entry and one Exit state for each model. Model states were typical states of a usage model. Each model state represented a different case of use and a different input of a possibility/probability. Composition states were those which were replaced by a sub-model. Collector states addressed the situations where multiple identical arcs were identical exit arcs of the model. The Collector state had a single exit arc, making model composition less complex. Models addressed all usage variables that could have been impacted within the model or its sub-models.
   Next Generation specific approach – Models were created for each of the main combat menu options – Navigation, Lights, Auxiliary, and Maintenance/Backup. Entry states were noted by the letter S. Exit states were noted by the letter T. Collector states were noted by a T and a number. Composition and Model states were numbered. Most Composition states had the word 'Screen' in the name of the state. Sub-models for each of the lower level menus were created first. The models were created in Word. They were also imported into tool SET Certify and existed there as .mod files.
9. Review sub-models and models – Each model was reviewed to ensure that the model was constructed correctly, appropriate state transitions were made, correct entry and exit states were defined, and lower level models were properly defined.
10. Do initial model composition – We had to perform conversion and transformation first to convert models from Word files into tool SET Certify. Check/analysis was performed next to ensure that models were structurally valid. Finally, we checked that models were composed. This was done to ensure that Composition states had the right numbers of entry and exit arcs same as sub-model actually had.
11. Determine expected outputs – When tests were executed, the tester had to ensure that two things occurred (in ascending order). First, that the correct state transition in the usage model was made. Second, that the correct response was issued within the performance requirements defined for the system. The correct response and performance for each state transition was documented.
    Next Generation specific approach - usage model transitions were mapped to test procedure actions. Each action in each test procedure was mapped to one or more state transitions in one or more usage models. In this manner, the detailed inputs as well as expected outputs were readily available for usage model generated tests. Mappings to existing test procedures were maintained as separate documents.
12. Insert probabilities into models/sub-models – Insertion of probabilities that were consistent with the definition of use and test goals. Next Generation specific approach - two or three top arcs were selected at each state, and the probabilities were assigned to these arcs. The rest of the weight was uniformly distributed across the remaining arcs. Probability information was gathered; this information was inserted directly into the tool SET Certify models.
13. Compose models and sub-models – Once probabilities have been defined, a final composition could occur for each set of models. The model composition utility was implemented to achieve this objective.
    Next Generation specific approach – Models were composed from the bottom up. All intermediate compositions were available, along with the fully composed models. Figures 2 and 3 show the examples of a Top-Level Model and a Sub-Model.
14. Conduct test – Tests were executed and the expected results were compared to actual results. Next Generation specific approach – The detailed inputs and expected outputs were determined from the mapping of usage model transitions to test procedure actions. Actual results were compared to these units of a test procedure.

Example of a Top-Level Model

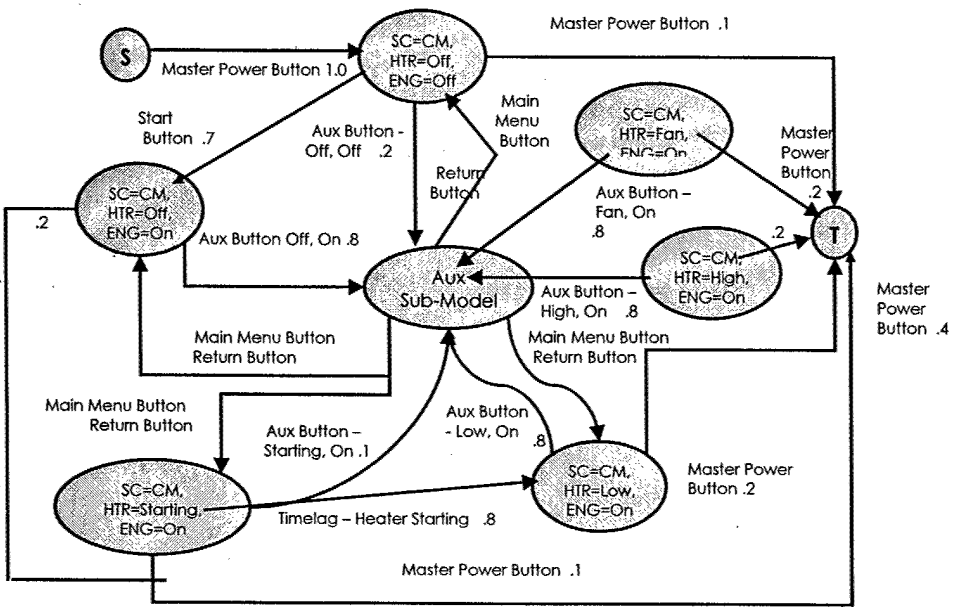

Figure 2: The SUT method - an example of a Top-Level model.

Example of a Sub-Model

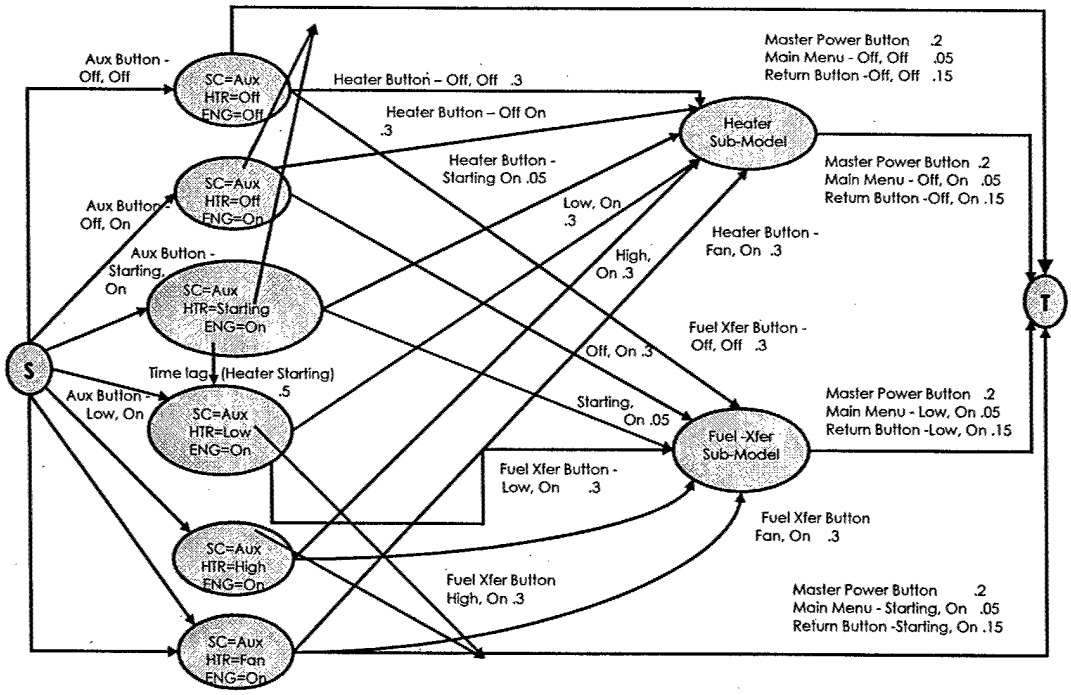

Figure 3: The SUT method - an example of a Sub-Model.

15. Compile test results – Passed tests, failed tests, and unresolved/un-executed test were recorded in tool SET *Certify*.
16. Analyze test results – Quality and stoppage criteria information were computed by tool SET *Certify*, once the test results have been entered.
    Make decisions based upon test results – Next Generation is in the process of adapting the SUT methods for health monitoring and diagnostics of selected line replaceable units of ground combat vehicles.

3. THE DARTS METHOD

The DARTS health and situation control continually tests the processing elements with Probe/Agent technology. Algorithms within the Health & Situation Control assess the health of the processors based on a criticality scoring system that considers mission requirements. Probes launched by the DARTS Controller query processing elements. The probed data is sent to a gauge that has a variable sensitivity or gain. Statistical Usage models and criticality scoring control the sensitivity of the gauge. In response to the gauge, the replicating process launches agents that can insert anomalous events for diagnostic purposes. In this context, a probe is a subset of an agent having only the ability to query without affecting framework, I/O protocol or Quality of Service. Each weapon system fitted with a DARTS Controller will control self-repair and reconfiguration of on-board processors utilizing a statistical based intelligent scoring system. It considers criticality of the function in the current battlefield situation.

Selected software components of soldier machine interface in a crew station will be modeled using DARTS architecture modeling techniques. The hardware environment will be modeled so that DARTS analysis tools can select compatible hosts from candidate processors. Missions will also be modeled so that DARTS tools can make intelligent choices considering the task criticality. DARTS models will automatically insert software probes into the crew station to monitor the system behavior. Gauges will determine if the system is operating within acceptable performance bands by monitoring data provided by the probes. DARTS will detect faults and select the optimal crew station configuration to maintain essential functionality in response to current battlefield conditions.

The experiment will inject artificial faults into the system according to known and anticipated patterns common to the weapon platform. The induced faults will reach a magnitude that will ultimately force DARTS to replicate a new processing element, bringing it on line as a replacement to the failed element.

The architectural models of the system software are part of the build standards that provide the software framework for processing elements expected to reside on a future military vehicles. The build standards will document structure of the reconfigured software and hosting processors. Failures will be inserted into the Systems Integration Lab (SIL) to exercise and validate each of the build standard requirements.

DARTS will construct correct configurations of software to load onto a vehicle for combinations of weapons systems, sensors, and missions. It will collect usage and runtime error data that can be used to improve the software development and testing processes. DARTS models will automatically insert software probes into the crew station to monitor the system behavior. Gauges will determine if the system is operating within acceptable performance bands by monitoring data provided by the probes. DARTS will detect faults and select the optimal crew station configuration to maintain essential functionality in response to current battlefield conditions. DARTS-collected usage information and runtime error patterns will be fed back into Next Generation SUT models to improve the modeling fidelity and software testing process. Success of this aspect of DARTS will be measured by the reduction in time for the SUT models to identify, isolate and repair errors. DARTS architecture descriptions will be used to improve SUT usage modeling techniques and processes.

The DARTS Probe Controller will employ the following tool: the Probe as an agent of the DARTS Controller, reporting the health of the weapon system elements. Off-vehicle probes will be also launched to assess the health of companion vehicles within the Operations Unit. Figure 4 illustrates the DARTS controller system health check process. Figure 5 shows the prognostics screen concept for different LRU's of the tank, based on the DARTS method.

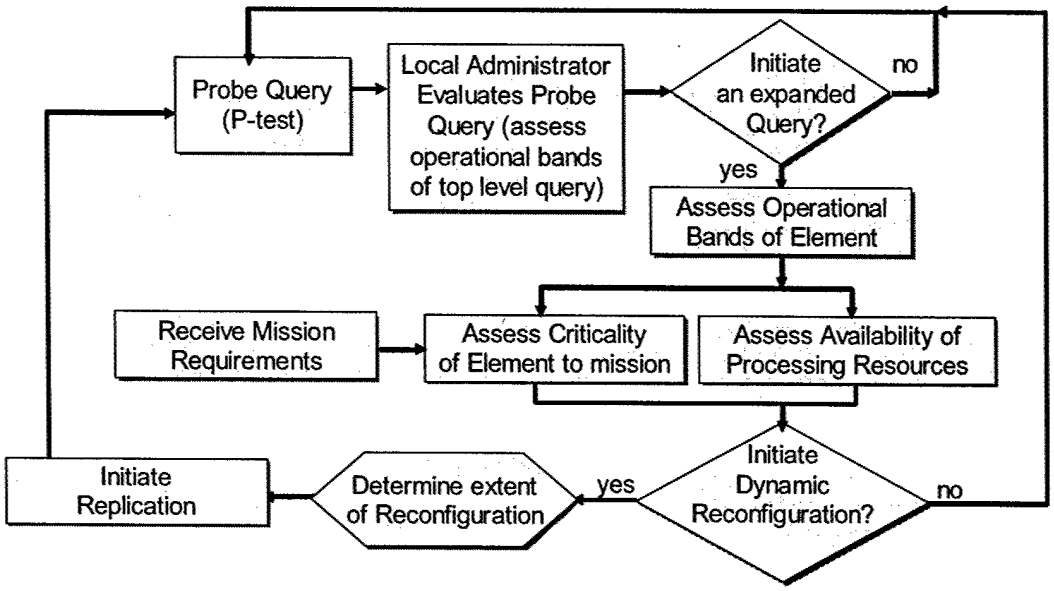

Figure 4: DARTS controller system health check process.

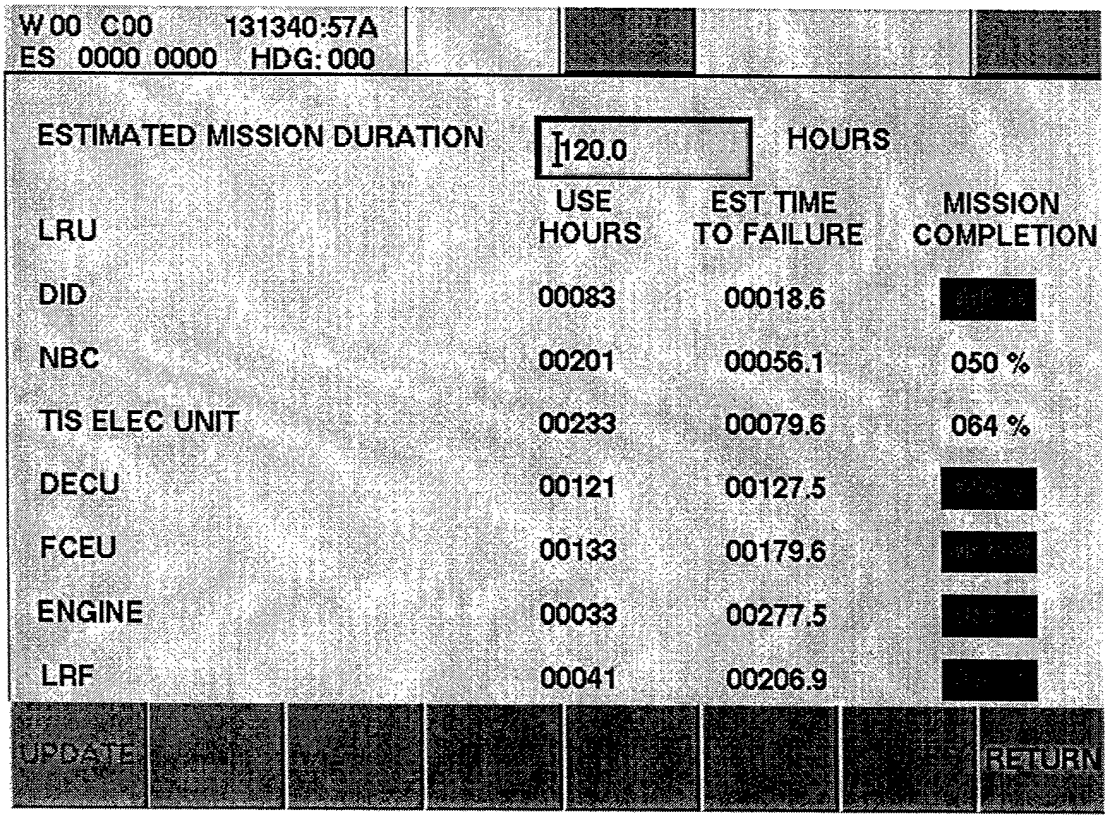

Figure 5: Commanders display showing the prognostics screen concept.

4. CONCLUSION

The DARTS health and situation control will test the processing elements continually with Probe/Agent technology. DARTS architecture descriptions will be used to improve usage modeling techniques and processes. DARTS-collected usage information and runtime error patterns will be fed back into Next Generation SUT models to improve the modeling fidelity and software testing process. Success of this aspect of DARTS will be measured by reduction of time of the SUT models development. DARTS is a software system that will enhance the performance of a weapon system by providing on-the-fly reconfiguration to accommodate the loss or malfunction of processing elements or to optimize onboard performance capability.

REFERENCES

1. M. S. Saboe, P. Gilbert, A. Kouchakdjian, "Applying Statistical Usage Testing (SUT) on a High-Complexity Application", *Proceedings of the Workshop on Statistical Methods in Software Engineering for Defense Systems,* National Academy of Sciences, Washington DC, July 2001.