A verification framework with application to a propulsion system

Bin Zhang\textsuperscript{a,\,*}, Marcos Orchard\textsuperscript{b,c}, Bhaskar Saha\textsuperscript{d}, Abhinav Saxena\textsuperscript{e}, Young Jin Lee\textsuperscript{f}, George Vachtsevanos\textsuperscript{f}

\textsuperscript{a}Department of Electrical Engineering, University of South Carolina, Columbia, SC 29208, USA
\textsuperscript{b}Department of Electrical Engineering, Universidad de Chile, Santiago 8370451, Chile
\textsuperscript{c}Advanced Mining Technology Center of the Universidad de Chile, Santiago 8370451, Chile
\textsuperscript{d}Palo Alto Research Center, 3333 Coyote Hill Rd, Palo Alto, CA 94304, USA
\textsuperscript{e}Stinger Ghaffarian Technologies Inc., NASA Ames Research Center, CA 94035, USA
\textsuperscript{f}School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

\textbf{Article info}

\textbf{Keywords:}
Verification
Offline verification
Runtime verification
Monte Carlo simulations
Propulsion systems
Automated contingency management (ACM)

\textbf{Abstract}

This paper introduces a novel verification framework for Prognostics and Health Management (PHM) systems. Critical aircraft, spacecraft and industrial systems are required to perform robustly, reliably and safely. They must integrate hardware and software tools intended to detect and identify incipient failures and predict the remaining useful life (RUL) of failing components. Furthermore, it is desirable that non-catastrophic faults be accommodated, that is fault tolerant or contingency management algorithms be developed that will safeguard the operational integrity of such assets for the duration of the emergency. It is imperative, therefore, that models and algorithms designed to achieve these objectives be verified before they are validated and implemented on-board an aircraft. This paper develops a verification approach that builds upon concepts from system analysis, specification definition, system modeling, and Monte Carlo simulations. The approach is implemented in a hierarchical structure at various levels from component to system safety. Salient features of the proposed methodology are illustrated through its application to a spacecraft propulsion system.

\copyright 2014 Elsevier Ltd. All rights reserved.

1. Introduction

The modern systems have continuously growing demands for improving the safety and survivability when unexpected faults/failures occur, which may lead to critical damage, expensive downtime, costly repairs, and even loss of assets and lives. For instance, aircraft/spacecraft and other military and industrial assets are increasingly required to operate with improved reliability and autonomy; they must be designed, operated, and maintained in ways that maximize their performance and availability with the presence of faults while reducing costs. To meet these demands, many condition-based maintenance (CBM) and Prognostic and Health Management (PHM) strategies have been developed (Chen et al., 2012; Liu & Han, 2014; Orchard, Hevia-Koch, Zhang, & Tang, 2013; Tran, AlThobiani, & Ball, 2014; Zhang et al., 2011). The successful implementation of CBM and PHM requires the subsystems, such as modeling, fault detection and identification, failure prognosis, etc., are well-designed to meet the performance specification from designers and customers. For this reason, verification and validation (V&V) has become an essential and integral component of the design cycle and brought potential benefits including the reduction of the design timeline, reduction of the life-cycle costs, and the improvement of the performance and reliability (Roychoudhury, Saxena, Celaya, & Goebel, 2013; Zhang, Tang, DeCastro, & Kai, 2010).

Verification and validation (V&V) of complex systems has drawn considerable attention from academia and industries in various application domains and has met with limited success over the recent past (Bartak & Rovensky, 2014; Bertolini, Oliveira, Justino, & Sabourin, 2013; Cpalka & Zalasinski, 2014; Kumar, Hammandu, & Gupta, 2013; Luque-Baena, Elizondo, Lopez-Rubio, Palomo, & Watson, 2013; Ouchania & Mohamedb, 2014; Souri & Jafari, 2014; Villalobos-Castaldi & Suaste-Gómez, 2013). Simulation platforms, field testing and formal methods have been exploited as potential means to address V&V issues (Li, Li, Xu, Rizy, & Kueck, 2010; Ouchania & Mohamedb, 2014; Seidel, Donath, & Haufe, 2012; Souri & Jafari, 2014). The complexity of large-scale dynamic systems has presented major challenges to the V&V designer necessitating new engineering-based methods to assist in the conduct of V&V studies. Researchers and practitioners do not quite agree on basic definitions on techniques to arrive

\footnotesize{\textsuperscript{\* Corresponding author.}
E-mail addresses: zhangbin@ece.sc.edu (B. Zhang), morchard@ing.uchile.cl (M. Orchard), bhaskar.saha@parc.com (B. Saha), abhinav.saxena@nasa.gov (A. Saxena), gjv@ece.gatech.edu (G. Vachtsevanos).}

http://dx.doi.org/10.1016/j.eswa.2014.03.017
0957-4174/© 2014 Elsevier Ltd. All rights reserved.
at solutions. The consensus currently within the defense community and other research establishments regarding the definitions for V&V is as follows (Tang, Orchard, Goebel, & Vachtsevanos, 2011 and Vachtsevanos et al., 2006):

Verification: The process of determining that a model/system implementation and its associated data accurately represent the developer’s conceptual descriptions and specifications.

Validation: The process of determining the degree to which a model/system and its associated data provide an accurate representation of the real world from the perspective of the intended use of the model/system.

Verification answers the question: “Have I built the system right?” (i.e., does the system as built meet the performance specifications as stated?). Validation answers the question: “Have I built the right system?” (i.e., is the system model close enough to the physical system and are the performance specifications and system constraints correct? (Vachtsevanos et al., 2006; Zhang et al., 2011). Different from verification, which usually depends on conceptual design and development, validation often relies on statistically sufficient data from system. System verification and validation activities are in support of system development activities to achieve system accreditation. Successful system accreditation occurs when an affirmative answer is obtained to the question: “Do I trust that the system will meet the system performance specifications within stated system constraints?”

This paper focuses on the topic of verification and introduces a generic framework with specific emphasis on model-based verification of a PHM system module. Different from the existing approaches (Ouchania & Mohamedb, 2014; Roychoudhury et al., 2013; Seidel et al., 2012; Souri & Jafari, 2014; Zhang et al., 2010), the proposed verification approach has novelities in the following aspects: (1) It is based upon the integration of system analysis, specification definition, modeling, and Monte Carlo simulations; (2) It extends traditional off-line verification into a combination with runtime verification; (3) It is featured with a hierarchical structure with verification at different levels. Due to these new features, the verification framework proposed in this paper requires more modeling and computing efforts. In addition, since this is a model-based verification, system modeling has direct impacts on the verification.

System model plays an important role in simulation-based off-line verification studies (Lee, Ryu, Lee, Shin, & Kim, 2010; Souri & Jafari, 2014; Vachtsevanos et al., 2006; Wei, Cai, Wang, Wang, & Gou, 2009). To better understand the system architecture, our approach decomposes a large-scale system into constituent constructs but also to capture their interactions. Structural and functional system models are developed towards that end. Then a generic verification architecture, consisting of off-line verification and runtime verification, is introduced to address major complexity issues. The verification of the complex system in the proposed approach is conducted at component-, system-, mission-, and safety-levels to guarantee the performance specifications are met for the system and the deployed PHM algorithms. The enabling technologies include mathematical tools for safety and optimization, temporal logic of actions, modeling and model check, and performance metrics. The developed verification approach is case studied on the automatic contingency management of a monopropellant propulsion system to demonstrate its efficiency for complex system verification.

2. The verification framework

A significant hurdle towards applying an effective verification methodology is the need for a good understanding of the system’s architecture. A possible approach typically consists of decomposing the large-scale system into a “system of systems” and defining constructs from the interactions between the components, i.e., partition the problem into well-understood sub-problems; furthermore, capture the interactions between components and the system behaviors using appropriate computational tools. Traditionally, verification is carried out after the system design is implemented, which is referred to runtime verification or field-testing verification. Such an approach incurs high costs when error correction iterations are necessitated. A new approach, which consists of off-line verification and run-time verification as shown in Fig. 1, therefore, is dictated and pursued in parallel with the system design. In this approach, off-line verification is based on a system model in a suitable simulation environment, which aims to correct design errors at early stage of design. Runtime verification is then implemented on the prototype system to track and assess system performance on stability/availability, constraints satisfaction, optimization, and reconfiguration etc. In this configuration, the verification of offline verification is low-cost and it will reduce the number of iterations of costly runtime verification, and finally reduce the cost of system design.

Fig. 1 depicts the off-line versus runtime philosophy. By increasing the inexpensive error correction (N) in offline verification, we can reduce the costly error correction (M) in runtime verification to reduce design cost and timeline.

With this new approach, a generic verification framework needs to be introduced in order to address major complexity issues. The essential elements of the architecture include system analysis, specification definition, system modeling, and Monte Carlo simulations. System analysis investigates the system functions, decomposes high-level missions or activities into operations of different system components, and provides a list of failure modes. Specifications dictated by the customer/end-user or the designer are used to assess the system performance. Exhaustive tests are carried out via Monte Carlo simulations to ascertain that the system/model performs properly under all operating conditions. In this paper, the verification is considered in the context of automated contingency management, both normal and fault conditions need to be addressed considered. For the fault operating conditions, a fault injector is employed to simulate fault scenarios. The proposed verification framework is shown in Fig. 2.

By viewing a complex large-scale system through a hierarchical configuration, a layered off-line verification scheme is constructed. It is based on an implementation of a Failure Modes and Effects Criticality Analysis (FMECA) (Lu, Jia, Gao, & Han, 2013;
and a decomposition of the system into its essential components. The proposed approach provides useful insight into the nature of fault modes and the system's operating regimes under fault conditions. The layered verification is implemented as illustrated in Fig. 3, following a system engineering philosophy and a division in terms of a system model and a high-level configuration/management model.

2.1. System analysis

A thorough review of system analysis studies, carried out as part of the design process, must be considered as a preliminary step to verification. Particular emphasis is placed on FMECA and other reliability-related results that will help to define requirements, fault modes and symptoms, etc. The modeling and verification activities must account for all system analysis findings.

As systems become more complex, verification increases significantly due to required testing resources – hardware in the loop (HIL) testing labor, hardware costs, etc. That is why simulation-based verification tools play an important role, avoiding the need of seeded fault tests in complex systems. Simulation-based verification considers the design and testing of a process model, which will be used to emulate the effect of operational conditions of interest in the system. Outputs of the model may be evaluated in terms of pre-determined performance indices, quantifying and rating the overall behavior of the process/system under analysis. Cost functions, indicators for performance of critical components in the system, and quality specifications are good examples of indices that may be used for this purpose. Monte Carlo methods are used in order to characterize the statistical uncertainty of failure modes, in terms of the probability distribution of failure time or intensity of the observed fault condition.

The model/system is usually examined as it is being executed, by comparing the simulation results in terms of the proposed performance indices. This almost always requires instrumentation, i.e., insertion of additional code, or sensors (if it is an actual system) to collect and monitor behavioral data during execution. Optimality in the design is verified by the implementation of cost functions as soft-sensors in the plant, in such a way that alternative approaches may be compared on the basis of those cost functions. In this sense, it is desired for both cost functions and performance indices to be related to expedite the evaluation of each proposed approach in the design.

2.2. System specification

Specifications are the benchmark against which to evaluate the performance of the model, i.e., whether the model is correct and whether it meets the customer requirements. Therefore, it is very important for system designers and customers to negotiate a common understanding of the system so that correct requirements and specifications can be established. Requirements are often related to conditions needed by customers to solve problems or achieve objectives. Specifications, on the other hand, often require complete and precise forms. Additionally, formulating a specification for a system also helps the design process. The act of describing a system or its behavior precisely can reveal overlooked “corner cases” or subtle interactions between subsystems. These problems are easier to correct in the design phase than during system implementation.

An important guideline in defining specifications is that they must be expressed in a form that can be checked against the output or response of the model. The original customer requirements are often in a form not easy to be evaluated and must be translated suitably. Some other crucial concerns that have to be taken into consideration are the system objectives, the operational environment, and a good understanding of the system functions, such as the input/output constraints, the physical limitations of some mechanical components, etc.

As an example of typical specifications, consider the case of a controller at the component level. Here, specifications may relate to stability, rise time, overshoot, settling time, and steady state error, among others. However, verification for fault diagnostic and failure prognostic algorithms may entail such specifications as false positive rate, false negative rate, confidence intervals, accuracy, etc. When the specifications are not satisfied, the controller must be adjusted or re-designed and the diagnostic/prognostic algorithms modified. Specifications for a high-level configuration/management model are usually very different. For example, the specifications for an automated contingency management (ACM) system of the monopropellant propulsion system, discussed below, are mission completion and cost. The first specification verifies whether the system can accomplish the mission with the given resources under the current operational conditions while the...
second verifies whether the ACM finds an optimal or sub-optimal solution. These two specifications, in turn, are described by the extra time taken and the fuel consumption to accomplish the mission.

2.3. System model and model verification

The system model plays an important role in simulation-based off-line verification studies. To build a model, we begin with a structural model of the system where components and their inter-connections are identified. A functional model is built next to describe the component functions. The functional model provides also information about the possible states, inputs, outputs and constraints imposed on the component functions. Simulation tools, such as Matlab Simulink and temporal logic based model-checkers are called upon to simulate the given system behaviors. In the example presented in this paper, a Simulink and a temporal logic of actions (TLA+) model (Li, Tang, & Li, 2009; Regnier, Lima, & Andrade, 2009) are developed in parallel with the latter used as a model checker to verify the former. Based on the assumption that both models will faithfully reflect the same system behaviors, any mismatch between the models may indicate an error in the modeling.

Model checking is a technique for formally verifying finite-state concurrent systems (Alrajeh, Kramer, Russo, & Uchitel, 2013; Gao, Xu, Zhan, & Zhang, 2013). It is chosen as the preferred method of system verification due to the intuitive appeal of its results and the efficiency of its approach. Specifications about the system are expressed as temporal logic formulas, and efficient symbolic algorithms are used to traverse the state space of the model and check if the system specification holds or not. Extremely large state spaces can often be traversed in significantly less time than simulation methods.

Among the plethora of software tools available for model checking purposes TLA+ (Li et al., 2009; Regnier et al., 2009) is chosen for our application since it is convenient for specifying and reasoning about concurrent systems. Systems and their properties are represented in the same logic, so that the assertion of the specifications that a system meets and the assertion of what a system implements are both expressed by logical implication. The verification for a system model can be illustrated in Fig. 4 and summarized in the following steps:

- Express the system and its expected behavior as logical constructs in TLA+ using the system’s functional model.
- Run the model checker to test the compliance of the system to stated specifications.
- Express the code for the control module under test in the form of TLA+ constructs.
- Verify the module (fault detection model, fault accommodation or contingency management model, etc.) against its own behavioral specifications independent of the application platform.
- Verify the control module along with the system with injected faults in order to check for overall system compliance in fault or anomaly conditions.

In the case of complex large scale systems, major components of the overall system are modeled and verified individually. These temporal logic modules are then collated to form the description of the entire system.

2.4. Monte Carlo simulations

When the model is verified, Monte Carlo (MC) simulations must be carried out to test the model under different operational conditions. The fault simulator generates different faults at randomly selected time instances during the system’s operation. The objective of the Monte Carlo simulations is to generate as much operational data as possible. From the results of MC simulations, a statistical evaluation of the verification scheme can be achieved. For some systems, test data for MC simulations may need to be prepared in advance according to different operational modes.

3. Verification strategy: a case study

In accordance with the objectives of verification, we will highlight those generic attributes of model verification that may be applicable to a variety of military and industrial assets. A mix of formal methods and good engineering practice constitute the innovative aspects of the approach.

The case in point is a hypothetical monopropellant propulsion system discussed in the NASA Fault Tree Handbook (Vesely et al., 2012). Space systems are required to operate autonomously. They must be robust to certain failure modes and must accommodate contingencies so that major mission objectives can be achieved. Fault tolerant design of these and other critical aerospace and industrial systems is attracting recently the attention of the research community.

3.1. A verification example of model

The monopropellant propulsion system under study uses hydrogen peroxide (H₂O₂) that passes over a catalyst and decomposes into oxygen, water, and heat, creating an expanding gas that produces the required thrust (Vesely et al., 2012).

The system, as shown in Fig. 5, consists of a reservoir tank of inert gas that feeds through an isolation valve IV1 to a pressure regulator RG. The pressure regulator senses the pressure downstream and opens or closes a valve to maintain the pressure at a given set point. Separating the inert gas from the propellant is a bladder that collapses as the propellant is depleted. The propellant is forced through a feed line to the thruster isolation valve IV2 and then to the thruster chamber isolation valve IV3. For the thruster to fire, the system must first be armed, by opening the IV1 and IV2. After the system is armed, a command opens the IV3 and allows H₂O₂ to enter the thruster chamber. As the propellant passes over the catalyst, it decomposes producing oxygen, water vapor and heat. The mixture of hot expanding gases is allowed to escape through the thruster nozzle, which in turn creates the thrust. The relief valves RV1-4 are available to dump inert gas/propellant overboard should an overpressure condition occur in any corresponding part of the system.

The corresponding functional model is depicted in Fig. 6. The circles denote the system components, the rounded rectangles...
represent the component functions, the boxes are external inputs, and the labeled arrows denote the system variables of interest. The Simulink model and the TLA+ logic model for the propulsion and ACM systems are then built in parallel. The logic models verify the models in the Matlab Simulink environment. This process is illustrated in Fig. 7. Note that the verification can be implemented at the component level, system level, as well as configuration/management level (ACM in Fig. 7), such as the mission level and system safety level.

The Simulink model of the monopropellant propulsion system is shown schematically in Fig. 8. The main components that must be modeled include the Heater, Tank, Pressure Regulator, and Valves. Certain assumptions have been made to simplify the modeling process. For example, when the heater is turned ON, the dynamic response of the temperature of gas in the tank can be simplified as a first-order system. If a more accurate model is required, the transfer function can be modified accordingly.

As an example of the verification methodology consider a simplified behavior of the monopropellant and the ACM module together. We restrict attention only to a qualitative analysis of the interactions between the ACM module and the monopropellant. It should be noted that the entire system may be modeled down to the level of partial differential equations involving the system variables, as dictated by first principle equations. However, it is our intention to only verify the logic of the ACM module as it applies to restricting the monopropellant from reaching unwanted states, since the quantitative details are being taken care of in the Simulink environment in the next section.

3.2. A verification example of ACM

The ACM system is expected to adapt autonomously to fault conditions with the goal of still achieving mission objectives by allowing some degradation in system performance within permissible limits. When a fault occurs, the ACM system will find the optimal solution to mitigate the impact of this fault on the entire system. For a complex system, a large number of components/subsystems could be subjected to fault conditions and different faults lead to different results. Some of them are critical and require immediate response while others are not critical and the system can maintain an acceptable performance with the fault over a certain period of time. The ACM module for verification is developed in a Simulink Stateflow platform and is illustrated in Fig. 9.

3.3. Hierarchical verification structure

As systems become more complex, the verification should be conducted at different levels leads to a hierarchical verification structure. In this hierarchical structure, the functions and algorithms are verified at component level where components are considered as an isolated unit, system level where all components are interconnected and functioned as an entire system, mission level where the mission objectives are integrated in verification, and safety level where the system safety under system failure are verified.

3.3.1. Component level verification

Verification at the component level is intended to ascertain that the component model is correct. The pressure regulator RG is a critical component in the monopropellant system and will be used as an example. The function of the RG is to maintain the output pressure at a given set value when the input pressure is equal to or higher than the set value. The model of the RG and its desired input–output characteristics are illustrated in Fig. 10.

The function of the RG is realized by regulating the flow of the inert gas from the input to the output and, therefore, a controller
must be designed to achieve this objective. In the monopropellant system, a proportional controller is used. Then, the response of the RG and its associated controller can be expressed as:

\[ P_o = \begin{cases} P_{sp} & \text{if } P_i \geq P_{sp} \\ P_i & \text{otherwise} \end{cases} \]

\[ n = K(P_{sp} - P_o) \]

where \( P_i \) is the input pressure, \( P_o \) the output pressure, \( P_{sp} \) the set value, \( n \) mass flow of inert gas through regulator, and \( K \) the controller gain.

Verification of the RG function must ascertain that the RG model can regulate the pressure according to the specifications stated in the above equations. The response of the RG model with a feedback controller is given in Fig. 11(a). It is clear that the function of the RG is not fulfilled when the input pressure is lower than the set value and the model needs to be modified.

After modification of the model, the response of the RG is given in Fig. 11(b). This time, the functions of the RG are correctly realized. Now, the specifications are checked. We will check for the steady-state error. Note that there is an obvious steady state error. Suppose the specification for the steady-state error is 0.1 psi, then...
Fig. 8. The Simulink model.

Fig. 9. ACM model.
the specification is not met. In this case, we need to adjust the parameter of the controller or re-design the controller.

Suppose next the proportional gain of the controller is adjusted to a large value, the result is shown in Fig. 11(c). It can be seen that the specification is now satisfied and the verification of this component is complete. If the specification is zero steady state error, the controller may change to a PI controller to meet this specification.

### 3.3.2. System level verification

Suppose that all the components and subsystems are verified. At the system level, components are interconnected and their couplings must be included in the overall model. The overall system must be verified as well. The model of the ACM system for the monopropellant propulsion is shown in Fig. 12.

The relationship between the heater and gas pressure will be used for illustration purposes. When the heater is turned ON to increase the gas temperature, the pressure of the gas tank will increase too. The increment of the pressure is related not only to the increment of the temperature, but also to the level of the gas in the tank. Fig. 13 shows the increase of the temperature and the pressure at different time instants and the same increment of temperature results in different pressure increments.

### 3.3.3. Mission level verification

At this level, high level commands, such as “moving forward x distance” or “increase speed to y miles/hour” are sent to the

---

**Fig. 10.** Schematic of the regulator model with input–output characteristics.

**Fig. 11.** Verification of RG.

**Fig. 12.** The overall model of the ACM system.
system model to check whether the mission can be fulfilled under normal and faulty conditions.

For the ACM system, verification at this level is intended to ascertain that when a fault occurs, the mission can be accomplished with the given resources. The following simulation shows the performance of the system in a scenario where a heater failure occurs after the regulator fault. The expected outcome of this situation is an elevated temperature (and pressure) inside the inert-gas tank. The results are shown in Fig. 14, where plots 1–6 illustrate the thrust mode, mission progress, tank pressure, RG output pressure, gas temperature, and gas level, respectively. As can be seen in plot 4, the regulator failure occurs at $t = 26$, causing the drop of the regulated pressure, and is detected at $t = 27$ by the ACM. To compensate for the pressure drop, the RG set-point is increased from 9 to 11.5, as shown in plot 3, to bring the Regulated Gas Pressure back to normal. At $t = 42$, the gas pressure falls below the new set point and the regulated pressure starts dropping again. The ACM reacts by turning the heater on at $t = 59$ to bring the pressure back. However, a failure occurs to heater and it fails to turn off after that. As a result, the ACM reacts at $t = 72$ by opening the relief valve 1 (RV1) for two seconds. The corresponding effect can be seen in plot 6 where the gas level decreases quickly when RV1 is opened. Since the temperature is still rising, the inert-gas pressure rises above safe levels for a second time and RV1 is opened again ($t = 82$) to release some gas. This indicates that the mission can be accomplished with the given resource when the ACM strategy is implemented.

3.3.4. Safety level verification

From previous example, it is clear that when the tank gas pressure is lower than the RG set-point, the heater must be turned ON to increase the pressure in the tank. Now, suppose a cascade fault occurs in the heater and it cannot be turned ON. As a result, the RG output pressure will become lower as more fuel is consumed. When the pressure is lower than a threshold value, the propulsion system will lose its function to provide thrust. Under such a scenario, the system must reset to a fail-safe mode before the thrust is totally lost.
Consider another example: suppose then the fault in the heater is that it cannot be turned OFF after it is turned ON. In this case, the pressure in the tank may continue to rise and reach a dangerous level. To lower the pressure, the system requires opening the relief valve RV-1. However, if this valve is stuck and the pressure in the tank cannot be lowered, the system also must revert to the fail-safe mode to avoid the loss of the system.

3.4. Monte Carlo simulations

Although ACM is designed to perform optimally for the described fault scenarios, results provided previously cannot guarantee that the optimality criteria have been met. To test whether the solution for ACM is optimal/sub-optimal, Monte Carlo simulations for the ACM system under the same fault scenario but different stages of the mission are performed. A comparison between an optimal ACM strategy with an ACM system without optimization is provided.

The specifications used to define the cost function in the optimization routine are the extra time and fuel consumption to accomplish the mission. These two specifications are formulated and different weights are assigned to both of them to form a total cost function as follows:

\[
\text{Total Cost} = 0.0009 \cdot t_{\text{heater, ON}} + 0.05 \cdot (t_{\text{final}} - t_{\text{scheduled}})^2 + \text{fuel consumption} + \text{extra time}
\]

Fig. 15(a) and (b) shows the results related to the extra time and the fuel consumption, respectively. Grey line is the cost related to ACM with optimization routine while black line is the cost related to ACM lacking such optimization. The results on extra time show that, when the fault occurs in the early 60 s, the ACM strategy with optimization needs more extra time to accomplish the task than that without optimization. If the fault occurs after 60 s of task, optimization does not cause too much difference. As for the fuel consumption, if the fault occurs in the early 60 s, the ACM strategy with optimization consumes much less fuel than that without optimization. Same as the situation in extra time, optimization does not cause too much difference on fuel consumption if the fault occurs after 60 s of task. When the total cost in the above equation is considered, the ACM strategy with optimization has a much less cost if the fault occurs in the early 60 s. The results indicate that the ACM system with optimization routine can more efficiently mitigate the failure and fulfill the mission if the fault occurs in the early stage of the task. If the fault occurs in the late stage of the task, there is not enough time and resource to do meaningful optimization.

4. Conclusions and future researches

This paper introduces a novel verification methodology that is applicable to the CBM and PHM schemes of a variety of safety critical systems. The contributions of the proposed verification approach includes a combination of offline verification with a runtime verification, a hierarchical structure to conduct verification at component-level, system-level, mission-level, and safety-level, and a framework that integrates system modeling in terms of structure and functions, with system analysis, performance specification definition, and Monte Carlo simulation. A case study of an automated contingency management scheme for a monopropellant propulsion system is developed to demonstrate the effectiveness of the proposed verification approach.

It is worth noting that the proposed approach is a systematic and generic solution. The suggested path of system analysis, specification definition, system modeling, and Monte Carlo simulations in a hierarchical structure can be suitably customized to address various systems in different fields, such as aircraft, nuclear power plants, power grids and delivery, unmanned ground/air/sea vehicles, among many others. The verification strategies entail tools and procedures that can be transferred to a variety of test cases. Since the proposed approach is model-based, it is, of course, necessary to proceed with a new system through the analysis, modeling and verification design steps followed in the example case presented here. It should be noted that the proposed verification approach can assist the designer to achieve optimum system design in addition to paving the path towards system validation and certification.

This paper only focuses on the work of verification and a number of works can be extended based on this work to make the V&V complete. Therefore, our first future research will be system validation, which is closely related to verification. To conduct validation, the system should be implemented under various working conditions with different fault scenarios to generate statistically sufficient data, which can be used toward the designed system for validation. The second future research is to include uncertainties in the system in V&V. To deal with the uncertainties in real systems, model adaptation, algorithm structure and parameter adjustments are absolutely necessary in the system design and should be verified and validated as well. Last but not the least, the algorithms in CBM and PHM may suffer from non-causal limitations since they require ground truth information that not available yet. For example, when we verify and validate prognostic algorithms in CBM and PHM, the actual remaining useful life of the failing component/subsystem will not be available before the component/subsystem become fail. Therefore, during the operation of the system, the
ground truth data have to be obtained from some prediction algorithms or from historical data of other systems with the same configuration. Some assumption may need to be defined and the process needs to be well developed and investigated.

References


Li, J., Tang, Z., & Li, X. (2009). Description and analysis of fairness on temporal logic of actions. In International conference on networking and digital society (pp. 41 and 44).


