Distilling the Verification Process for Prognostics Algorithms

Indranil Roychoudhury[1], Abhinav Saxena[1], Jose R. Celaya[1], and Kai Goebel[2]

[1] *Stinger Ghaffarian Technlogies, Inc., NASA Ames Research Center, Moffett Field, CA 94035, USA*

{*indranil.roychoudhury, abhinav.saxena, jose.r.celaya*}*@nasa.gov*

[2] *NASA Ames Research Center, Moffett Field, CA 94035, USA*

kai.goebel@nasa.gov

ABSTRACT

The goal of prognostics and health management (PHM) systems is to ensure system safety, and reduce downtime and maintenance costs. It is important that a PHM system is verified and validated before it can be successfully deployed. Prognostics algorithms are integral parts of PHM systems. This paper investigates a systematic process of verification of such prognostics algorithms. To this end, first, this paper distinguishes between technology maturation and product development. Then, the paper describes the verification process for a prognostics algorithm as it moves up to higher maturity levels. This process is shown to be an iterative process where verification activities are interleaved with validation activities at each maturation level. In this work, we adopt the concept of technology readiness levels (TRLs) to represent the different maturity levels of a prognostics algorithm. It is shown that at each TRL, the verification of a prognostics algorithm depends on verifying the different components of the algorithm according to the requirements laid out by the PHM system that adopts this prognostics algorithm. Finally, using simplified examples, the systematic process for verifying a prognostics algorithm is demonstrated as the prognostics algorithm moves up TRLs.

1. INTRODUCTION

Prognostics and health management (PHM) systems are important to ensure safe and correct operation of real-world engineered systems, reduce their downtime, and reduce maintenance costs. Integral components of PHM systems include diagnostics and prognostics algorithms, the associated diagnostics and prognostics models, sensors, and other hardware, and interfaces between these different components. Diagnostics algorithms involve fault detection, isolation, and identification capabilities; and contribute towards system safety by enabling fault mitigation steps. prognostics algorithms involve prediction of how the system will evolve in the future, thereby contributing towards system safety. Prognostics algorithms also enable reduction of downtime and maintenance costs by providing decision makers with predictions of future system behavior so that decision makers can use this information to either take preventative, fault mitigating, or maintenance actions, or modify mission operations to prolong system use, and maximize mission utility.

Before a PHM system can be deployed in real-world scenarios, it is critical that the PHM system undergoes verification and validation. At the most general level, *verification* of a product is the process where stakeholders answer the query "are we building it right?", while *validation* of a product is the process where stakeholders answer the query "are we building the right thing?" Intuitively, verification is the quality control process of evaluating whether or not a product, service, or system complies with testable constraints imposed by requirements at the start of the development process. In contrast, validation is the quality assurance process of evaluating whether or not a product, service, or a system accomplishes its intended function when fielded in the target application domain.

A PHM system may include several hardware and software components, including software implementations of diagnostics and prognostics algorithms. While many publications discuss the verification of hardware (Gupta, 1993; McMillan, 2000) and software verification (Bérard et al., 2010; Wallace & Fujii, 1989) only, this paper focuses on the verification of all the different components that constitute prognostics algorithms. To this end, first, this paper distinguishes between technology maturation and product development contexts to

characterize various PHM verification and validation scenarios often discussed in the literature, and then, proposes a process that identifies specific steps that can facilitate verification of prognostics algorithms. Specifically, the contributions of this paper are as follows:

1. This paper describes the verification process for a prognostics algorithm as it moves up to higher maturity levels. In this work, the concept of technology readiness levels (TRLs) is adopted to represent the different maturity levels of a prognostics algorithm.
2. Next, it is shown that at each TRL, the verification of a prognostics algorithm depends on verifying the different components of the algorithm according to the requirements laid out by the PHM system that adopts this prognostics algorithm.
3. Finally, using simplified examples, the systematic process for verifying a prognostics algorithm is demonstrated as the prognostics algorithm moves up TRLs.

2. Verification and Validation of What - A Product or a Technology?

In order to put our proposed view of the maturation process into context, first we distinguish between developing a system or a *product*[1] versus maturing a *technology*[2]. The development of a system/product is driven by the high level need to accomplish a certain goal in a specific application, whereas technology is understood to be more general and applicable to more than one system when matured.

Examples of systems or products include PHM systems, such as a health and usage monitoring system (HUMS) (Romero, Summers, & Cronkhite, 1996), battery health management system (BHMS) for an electric unmanned aerial vehicle (e-UAV) (Saha et al., 2011), health management system for a water recycling system (WRS) (Roychoudhury, Hafiychuk, & Goebel, 2013), and so on. As shown in Figure 1, a PHM system generally consists of several components, such as sensors (including data acquisition (DAQ), signal conditioner, etc.), technologies such as diagnostics and prognostics algorithms, diagnostics and prognostics models, and other hardware (e.g. communication channels, decision making, interfaces, data storage, and displays, among others). Some of these components, such as sensors, DAQ, etc., are often already matured technologies used in commercial off-the-shelf (COTS) products while others such as prognostics algorithms may be viewed as technologies that need to be matured before they can be used in the PHM systems.

An example of a prognostics algorithm or technology is the `ComputeRUL` algorithm, whose flowchart is shown in Figure 2. `ComputeRUL` consists of three main functions: (*i*) current state estimation, (*ii*) future state prediction, and (*iii*) remaining useful life (RUL) computation. The current state estimation function takes as inputs the sensor readings and the system input data and estimates the current state of the system using a particle filtering scheme (Arulampalam, Maskell, Gordon, & Clapp, 2002) that uses a prognostics model of the system. The future state prediction function takes, as inputs, estimated future operational and environmental profiles and uses a Monte Carlo technique (Kalos & Whitlock, 2008) to predict future system state using the prognostics model. Finally, the RUL computation function compares the predicted values of system state to a predefined threshold and computes RUL as the time remaining before the predicted system state values cross this threshold (Daigle & Goebel, 2011).

Verification and validation are key steps in maturing both products and technologies; however the specifics for each

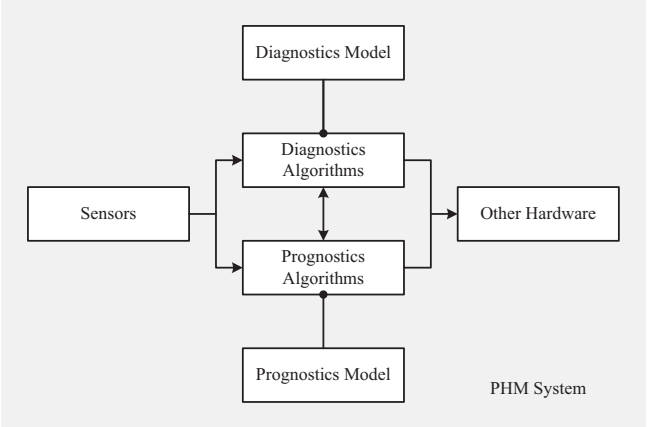

Figure 1. Typical components of a PHM System.

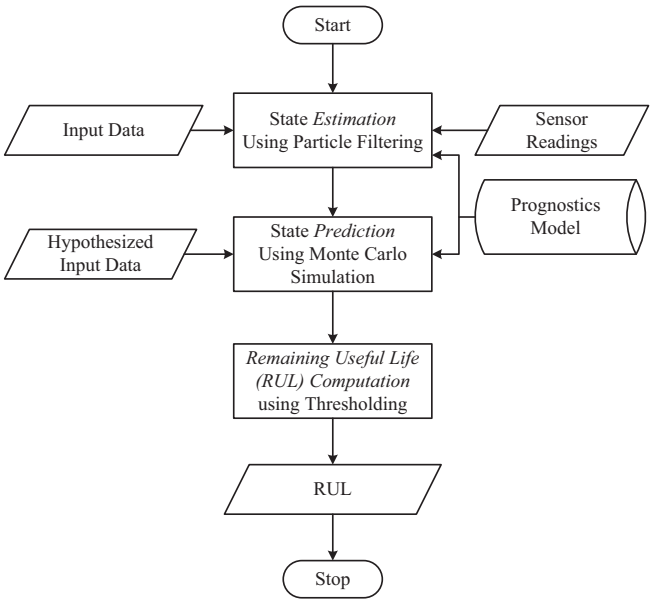

Figure 2. Flowchart of `ComputeRUL`, an example prognostics algorithm.

[1] In this paper, we use the terms 'system' and 'product' interchangeably.
[2] We use the terms 'algorithm' and 'technology' interchangeably.

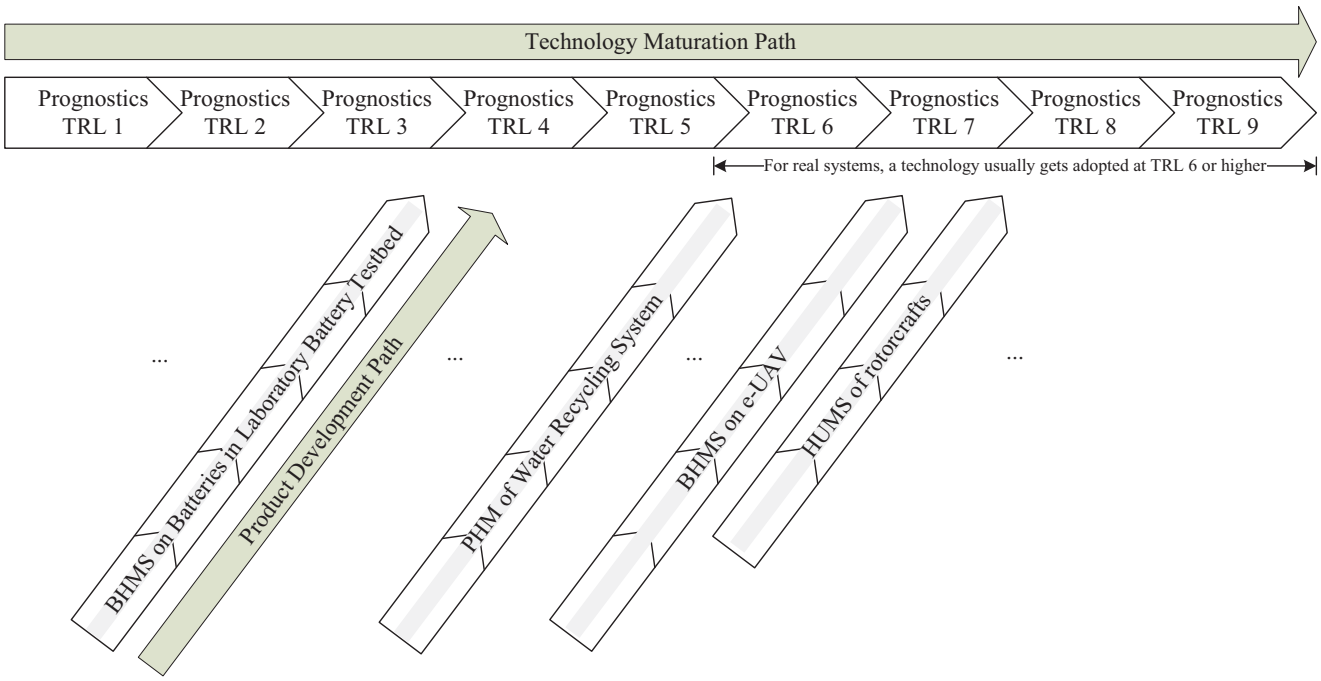

Figure 3. The technology maturation and product development paths.

case may differ. It is important to understand these differences in order to clearly identify what involves verification and validation of a prognostics technology. In the PHM literature we identified that there are several efforts that report verification and validation activities in their respective applications or products, however it was very confusing to get a consistent understanding of what activities are geared towards verification, and what activities enable validation, separately (Tang et al., 2007; Feather & Markosian, 2008). Many efforts combine verification and validation as one task (Aguilar, Luu, Santi, & Sowers, 2005; Byington, Roemer, Kalgren, & Vachtsevanos, 2005), while others use similar methods but sometime refer to them by verification and sometime by validation. Further, most of these reported developments represent different levels of technical maturity, or in other cases, different levels of system integration. Here we attempt to describe a systematic process that allows us to put most of these efforts into a common context and clearly identify the nature of distinct verification and validation activities.

But first, we distinguish between two related but different contexts that influence the nature of verification and validation activities but often get confused with each other, i.e. *product development* versus *technology maturation* (Hicks, Larsson, Culley, & Larsson, 2009). While the steps for both activities look similar there are some differences that are important to understand verification and validation for PHM system development versus for prognostics technology maturation.

Product development typically starts from a top level need for a product (such as a PHM system) for which several ideas may be evaluated at the concept stage. Based on a selection process some ideas move forward with development. At that point a system gets broken into its subsystems and components and requirements *flow down* (Saxena et al., 2012) for individual component development and system integration. At the lowest level some of these components may already exist as COTS components based on mature technologies. But if there are gaps identified, new components may be developed using new technologies. It must be noted that in this scenario the new technology is developed and matured with an end product in mind and, therefore, most of the testing is driven by the requirements flowed down from the top level product. Once developed, these components are first tested individually (quality control process) and then integrated into a subsystem, which undergoes quality control again at the integrated level. At each level tests are designed to help fulfill higher level requirements. This process continues iteratively until the entire system has been integrated and tested as a whole. The product can then be further certified for specific use by domain-relevant certifying agencies.

The *technology maturation*, on the other hand, typically starts at the very low level where a technological concept is considered potentially useful. Prototypes and simulations are developed and tested on simpler cases. Feedback is used to refine the implementation and retest. It is desirable to apply and demonstrate the technology to a variety of applications for establishing its generality. This is often accomplished by

proof-of-concept developments for various use cases. Note that each of these proof-of-concept use cases can be considered as a *product* with its own product development cycle, and a successful development of each of these products helps in placing increased trust in the new technology as a whole (see Figure 3), consequently also increasing the maturity level. Conversely, the technology can also be matured without any specific product in mind, or rather with several potential products in mind. As a technology matures through demonstrations and testing it may be adopted for a specific use case for which a directed and dedicated product development cycle is usually followed. Specific verification and validation activities, may be pursued in order to integrate this technology into that product. Note that although the technology at this point can be claimed as matured to be used in that particular product, the generality of technology may allow it to be usable for other products, often with required customization.

2.1. Technology Maturation

We realize that from our research-perspective, maturation of a prognostics algorithm as a technology falls under the general technology maturation category. There are several efforts, currently undergoing, to integrate prognostics algorithms into specific PHM system products such as BHMS for an e-UAV, health management of a WRS, and so on. Therefore, from here on we will describe what prognostics technology maturation would look like and what the verification and validation specific steps are for this maturation. In this work, we adopt NASA's Technology Readiness Level (TRL) concept (Mankins, 1995) to describe various maturity levels for a prognostics technology with no particular preference. Other similar concepts can be used just as well. With this understanding, a technology moves up the TRL as it matures, whereas a product moves up a system integration ladder as it gets developed.

NASA TRLs are defined from TRL 1 through 9 (Mankins, 1995). TRL 1 describes a technology at its very concept or first level of maturation, where only basic principles are observed and reported. TRL 2 describes the stage when a technology concept and/or application is formulated. At TRL 3, analytical and experimental critical function and/or characteristic proof of concept of the technology has been performed. Component and/or breadboard validation in laboratory environment is performed in TRL 4. TRL 5 represents the stage when the component and/or breadboard validation is performed in a relevant environment, while TRL 6 indicates the maturation stage when the system/subsystem model or prototype demonstration is performed in a relevant environment (either ground or space). When the technology reaches TRL 7, the system prototype demonstrations are performed in a space environment. TRL 8 represents the stage when the actual system gets completed and flight qualified through test and demonstration (either ground or space), and finally, TRL 9 represents the stage when the actual system is 'flight proven' through successful mission operations. Note that while an OEM component has reached TRL 8 or 9, it may be integrated into a larger system (product) which is at a lower integration level, i.e. not a full system on its own.

From the technology's point of view, Figure 4 illustrates that at each TRL of a technology, such as prognostics, both verification *and* validation activities must be performed. It is expected that at low TRLs (TRL 1-2), more effort would be on validation of the concepts than verification because the goal at these TRLs is to ensure that the prognostics technology is indeed useful in accomplishing system level performance, safety, and cost goals. In these stages, the technology is still being developed and is adopted in less mature prototypes and products. At middle TRLs (TRL 3-7), more effort is expected on verification activities than validation, since at these TRLs, the emphasis is on adopting and implementing a particular prognostics technology (already verified and validated at lower TRLs) in different PHM systems (at different maturation levels). At high TRLs (TRL 8-9), relatively more effort is again on validation than verification, since by now it is established that the implementations of the prognostics technology (in middle TRLs) are verified and validated to be 'working', and the emphasis at higher TRLs is to ensure that the intended functions of the target PHM system that adopts this prognostics algorithm is fulfilled successfully. As we can see in the above description, it is clear that verification and validation of a prognostics technology at any TRL assumes completion of verification and validation at previous TRLs.

Figure 4 also points out that the scope of the products (e.g. PHM systems) that adopt this technology gets more focused as the prognostics technology development proceeds from low TRLs to high TRLs, e.g., from less-mature PHM systems implemented on breadboards to mature BHMS for the particular Lithium-Ion batteries used in the e-UAV. Moreover, as the prognostics technology matures to higher TRLs, they get integrated into PHM systems that are part of progressively larger systems.

2.2. Product Development

From a products point of view, verification and validation steps are performed for the product (PHM system in our case) by verifying and validating each of its components, the interfaces between these components, and their interactions. The individual components of a product, however, follow their own maturation cycle and integrate into the main product life-cycle when they have matured to a certain degree within their own maturation scale. For example, the prognostics algorithm, like other components of the PHM system, follows its own maturation (TRL development) stages and gets integrated into a product when a minimum TRL is achieved. Typ-

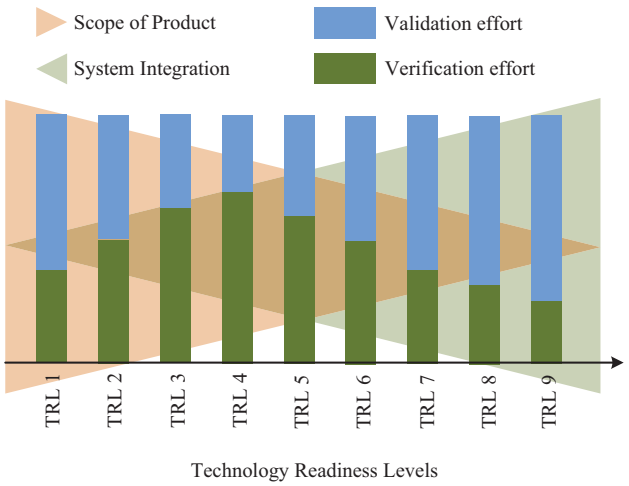

Figure 4. Verification and validation activities across different TRLs.

ically technologies demonstrated to be at TRL 6 or higher are considered a candidate to be integrated into a product ready for use. Once integrated into a product, the prognostics algorithm gets verified and validated together with other integrated components, as a subsystem for that product. Note that other components of the PHM system, e.g. original equipment manufacturer (OEM) components such as sensors may be at very high maturation level, but the product, i.e., the PHM system, as a whole may not be considered fully mature.

In the remainder of this paper, we focus our attention specifically on the verification of the prognostics algorithms, which is the main topic in this paper. The validation of prognostics algorithms is beyond the scope of this work, and will be investigated as part of future work. However, for the sake of highlighting how both verification and validation activities are performed at different TRLs (shown in Figure 4), we will describe some specific validation activities while discussing the case study (in Section 4). The illustrated examples of validation activities also help in drawing a contrast with specific verification activities at various TRLs, especially since, in literature, these validation activities are often included as verification that leads to confusion.

3. Verification of Prognostics Algorithms

Before we describe the verification process for prognostics algorithms, we first have to define what constitutes a prognostics algorithm. Figure 5 shows the different components of a prognostics algorithm when adopted by a product, such as a PHM system. The components of a prognostics algorithm, according to our understanding, are:

- The core prognostics algorithm (CPA) is a high-level abstraction of the prognostics approach which can be represented in terms of a system block diagram, a flowchart, or pseudocode. It is not implemented code. Figure 2 presents an example of the CPA for the `ComputeRUL` prognostics algorithm.
- The implementation specific aspects (ISA) relate to a particular implementation of the core prognostics algorithm (denoted by CPA) in a particular coding language and a particular computational processing architecture and hardware.
- The domain specific entities (DSEs) of a prognostics algorithm when the prognostics algorithm gets adopted in a particular product. The DSE will typically include domain-specific models. We note that every diagnostics and prognostics algorithm is based on a corresponding underlying model. For instance, in classical model-based prognostics algorithms, the models may be state-space models or some other mathematical construct or abstraction that represent or describe physical behavior of the system under consideration. These models can be built upon the use of physics first principles or empirically by observing the physical behavior. For data-driven prognostics algorithms, DSEs consists of domain specific feature extraction methods along with structures for different mathematical abstractions. These abstractions are typically built by observing and extracting the information available in the data often without explicit use of physical phenomena knowledge. As a result, in the data-driven prognostics context, features and abstractions as part of the DSEs are typically equivalent to the concept of features and models in statistical learning. For example, in case of data driven diagnostics and prognostics algorithms, mathematical constructs such as Artificial Neural Network (Yegnanarayana, 2004), Gaussian Process Regression (GPR) (Seo, Wallat, Graepel, & Obermayer, 2000), etc. are trained using data by learning parameters and fixing a structure (topology, covariance structure, etc.) to develop models that can be regressed to make predictions.
- The data sources (DS) consist of sensor measurements of physical variables. These data are typically assumed to be part of a modern instrumentation system in which a transducer is used to measure a physical quantity and its output is processed through a signal conditioner and DAQ in order to obtain a digital representation of such measurement that can be logged for future usage, or use immediately by the algorithm. There are cases in which a physical quantity is not directly measured by a physical sensor but it is estimated from other physical sensors.

By making distinctions between the different components, we identify separate pieces of a prognostics technology that can be verified in parts. This decomposition, to our understanding, makes it easier to verify if something changes, since, in this way, whoever makes a change to a particular component

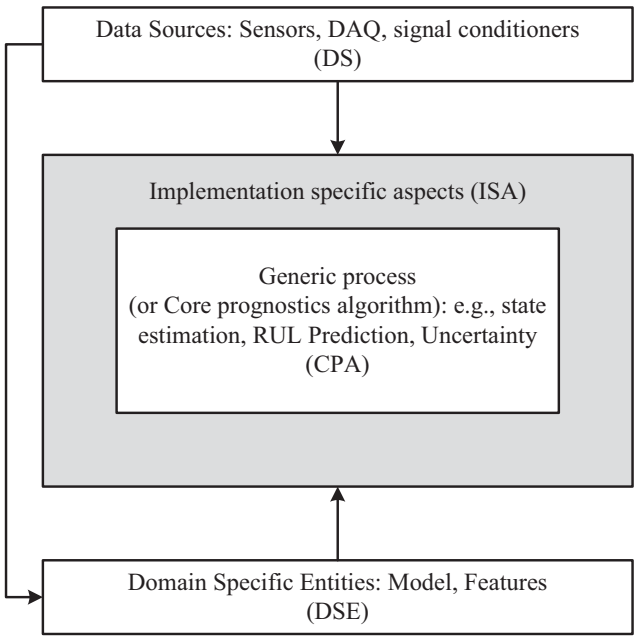

Figure 5. Components of a prognostics algorithm adopted by a PHM System.

must also verify that (updated) component at that level. Typically, low TRL technologies get developed by different contractors and do not get adopted in a product until verified and validated at that level. So from a high TRL, system integrators' point of view, they are not tasked with the verification of low-level components. But, if a component changes at a high TRL, the component will have to undergo verification at that high TRL again.

As an example, consider the `ComputeRUL` algorithm whose CPA is described using the flowchart in Figure 2. It is possible that ISA, DSE, and DS may all change at different maturation levels of `ComputeRUL`, but the CPA may remain the same. The ISA, DSE, and DS change because through the course of maturation of the CPA, several systems or products employing such technology are going to be developed at different points in time and likely by different parties with different target application domains.

For example, at TRL 4, the ISA could be in Matlab on a laboratory computer, while at a higher TRL (say, TRL 8), perhaps the ISA would be an assembly language implementation running on embedded processors onboard the e-UAV. Similarly, at TRL 3, DS could be a simulated data, while at TRL 8, the DS could be the actual system sensors onboard the e-UAV as its data sources. Similarly the DSE for TRLs 3 and 8 could be the model of a generic COTS battery cell, and a high fidelity model of the specific Lithium-Ion battery used on board the e-UAV, respectively.

At each TRL, the verification of the prognostics technology implies the verification of ISA with respect to requirements defined using the corresponding DSE and DS. As the prognostics technology moves from one TRL to the next, if any of the DSE, DS, or ISA of the prognostics algorithm at the higher TRL differs from those at the lower TRL, all three components need to be verified again at the higher TRL. Typically, at higher TRLs, the ISA and corresponding DSE may be the same from one TRL to the other, but the DS usually changes.

Recall that verification is the quality control process of evaluating whether or not a product, service, or system complies with testable constraints imposed by *requirements* at the start of the development process. Therefore, requirements play an integral part in verification efforts, and, verification, in a way, can only be as good as the requirements (Saxena et al., 2012; Rajamani et al., 2013; Saxena, Roychoudhury, Lin, & Goebel, 2013). Several publications list various attributes that characterize the goodness of individual requirements, as well as the set of requirements (Firesmith, 2003; Sommerville & Sawyer, 1997). For the sake of brevity, we will discuss here only the most important of these attributes good requirements must fulfill to enable verification:

1. Each requirement must be *verifiable*, i.e., a finite, cost-effective process has been defined to check that the requirement has been attained.
2. Each requirement must be *attainable* (or, *feasible*), i.e, solutions exist within performance, cost, and schedule constraints and the requirement can be satisfied within the constraints of the project.
3. Each requirement must be *unambiguous* (or, *understandable*), i.e., it expresses objective facts, not subjective opinions, and it is subject to one and only one interpretation.
4. Each requirement must be *design independent*, i.e., each requirement does not specify a particular solution or a portion of a particular solution. Stating implementation instead of requirements can lead to major issues, such as forcing a design where it is not intended, or leading the authors of these requirements to believe that all requirements are covered.
5. Each requirement must be *traceable* to an originating high-level requirement. Traceability refers to relationships between parent and child requirements, and between requirements and other design goals. Every requirement should be traceable to the needs, goals, objectives, and constraints of the target application.
6. The set of requirements must be *complete*, i.e., everything the system is required to do throughout the systems life cycle is included. Completeness is a desired property but cannot be proven at the time of requirements development, or perhaps ever.

The property of *traceability* is very important. This is because, a PHM system by design is almost always a part of a larger target system, and typically, PHM system requirements are derived from high level performance, cost, and schedule requirements of these target systems (Saxena et al., 2012). Such high level requirements are typically generated by the customer (the stakeholder who concerns with getting the system built), and often times, the vendor (the stakeholder who concerns with building the system to the customer's satisfaction) must flow down the requirements from the high level customer-requirements to low-level testable requirements.

As part of previous work, in (Saxena et al., 2012), we developed a process to flow down high level functional requirements to low level prognostics performance metrics parameters and illustrated this process using an e-UAV scenario. The low level prognostics metrics take into account several performance factors such as precision, timeliness, accuracy, and prediction confidence, e.g. the α-λ and β metrics developed in (Saxena et al., 2012).

4. Case Study: Verification of ComputeRUL

This section presents a procedure for verification of the prognostics algorithm `ComputeRUL`. As mentioned in Section 3 above, a prognostics algorithm consists of four distinct components, namely CPA, ISA, DS, and DSE based on the product that has adopted the prognostics technology at a particular TRL. For this particular example, as the prognostics algorithm moves to a higher TRL, the CPA is assumed to remain unchanged, although this is not always the case. However, the other three components, i.e., ISA, DS, and DSE, may change as the prognostics algorithm moves to higher TRLs, requiring that the prognostics algorithm is verified again.

Table 1 presents the four components of a prognostics algorithm at different TRLs along with a list of verification and validation testing activities at each TRL. At TRL 1, the prognostics algorithm `ComputeRUL` is in a concept form, and exists as a flowchart (shown in Figure 2). Recall that verification tests involve checking the implementation correctness while validation tests involve checking for functional correctness. Since there is no 'real' implementation, there are no DSE and DS for this algorithm at this TRL, and the testing activities involve evaluation of the applicability of the `ComputeRUL` towards predictive life estimation towards health management. At such a low TRL, therefore, the nature of the testing of this algorithm is more of validation than verification.

In TRL 2, `ComputeRUL` is implemented on paper using the detailed mathematical abstractions for particle filter (Arulampalam et al., 2002) and Monte Carlo methods (Kalos & Whitlock, 2008). The DSE at this stage involves representative nonlinear state-space equations of batteries, and the DS involved 'made-up' synthesized data from general battery dataset. The goal of testing activities at this stage is still more validation-oriented, and involves determining if the `ComputeRUL` algorithm can be applied to battery discharge prediction using current and voltage data. It is also important to study the battery data and ensure that features are available that correlate monotonically to measure fault growth in batteries.

At TRL 3, `ComputeRUL` is implemented using C++ on a generic computer. The DSE include equations of battery of arbitrary chemistry, and the DS used involve damage progression battery data obtained from simulations. The test activities at this stage include both verification and validation activities. The verification activities involve ensuring that uncertainty quantification error, modeling and discretization errors are within allowed limits. Validation activities involve ensuring that α-λ performance, prediction horizon, convergence, confidence interval, statistical hypothesis testing, reliability metric etc. are within allowed limits.

At TRL 4, `ComputeRUL` is implemented in C++ on the computer in the battery testbed in the laboratory. The DSE includes equations of Lithium-Ion batteries similar to those on-board the e-UAV. The DS at this TRL involves data from Lithium-Ion batteries in the environmental chamber (in laboratory setting) with constant load profiles. Both verification and validation activities make up the test activities at this TRL. The verification activities involve ensuring that measurement errors are within allowed limits; the algorithm works correctly in the presence of manufacturing variability; the channel biases are kept at a minimum; and that the algorithm works for constant load profiles. Validation activities involve ensuring that α-λ performance, prediction horizon, convergence, confidence interval, statistical hypothesis testing, reliability metric etc. are within allowed limits.

At TRL 5, the ISA and DSE of `ComputeRUL` remains the same as in TRL 4. However the DS now involves data from Lithium-Ion batteries in the environmental chamber (in laboratory setting) with varying load profiles, and hence the prognostics technology will have to be verified and validated again. The verification and validation testing activities at TRL 5 are similar to that of TRL 4.

The ISA for `ComputeRUL` at TRL 6 involves MATLAB implementation of the CPA running on computers similar to those on-board the e-UAV. The DSE include equations of exact type of Lithium-Ion batteries used on-board e-UAV. The DS at this TRL includes played-back data from actual Lithium-Ion batteries onboard the e-UAV from multiple ground tests. Verification tests at this TRL include ensuring that the no coding errors are made; discretization and sampling rate errors are avoided; and no communication errors occur. Validation tests include ensuring that prognostic horizon, computation time, α-λ performance, robustness to system noise, prediction update rate, etc. are within requirements.

At TRL 7, `ComputeRUL` is implemented in MATLAB running on the actual computers on-board the e-UAV. The DSE are equations of exact type of Lithium-Ion batteries used onboard the e-UAV and the DS consists of real-time data from actual Lithium-Ion battery sensors onboard the e-UAV from multiple flight tests with simplistic (safe) flight profiles.

Since the CPA, ISA and DSE of `ComputeRUL` does not change from TRL 7 - 9, once the ISA, DSE, and the interfaces are verified in TRL 7, they do not need to be re-verified in TRL 8 and 9. But, since DS changes from real-time data from actual Lithium-Ion batteries onboard the e-UAV from multiple flight tests with complex flight profiles in TRL 8 to real-time data from actual Lithium-Ion batteries onboard the e-UAV from actual science flight missions, validation activities are performed again at both TRL 8 and 9, and involve ensuring that prognostic horizon, computation time, α-λ performance, robustness to system noise, prediction update rate, etc. are still within requirements.

In our case study, we use an example that used BHMS products at each TRL to demonstrate how the prognostics algorithm matures to higher TRLs. But, as is shown in Figure 3, maturation can also be done through different products or systems (e.g., PHM of Water Recycling System, HUMS of rotorcrafts, etc.) with necessary customization and testing.

5. Conclusions and Future Work

In this paper, we presented a systematic process of verification of prognostics algorithms. We distinguished between technology maturation and product development processes, and described the systematic process of verification of a prognostics algorithm as it moves up to higher maturity levels. This process is iterative where verification activities are interleaved with validation activities at each maturation level. It was shown that at each maturation level, verification of a prognostics algorithm depends on verifying the different components of the algorithm according to the requirements laid out by the PHM system that adopts this prognostics algorithm. Finally, using simplified examples (mostly from the battery health management domain), the systematic process for verifying a prognostics algorithm was demonstrated.

In reality, verification and validation of prognostics technology is not trivial. These challenges arise from use of non-deterministic approaches to account for uncertainty in prognostics and the self-evolving nature of these algorithms exhibiting learning behaviors both of which result in an infinite testing space from an exhaustive verification point of view, which is practically impossible to cover. Apart from mathematical or theoretical limitations, prognostics methods suffer from acausality limitations towards their validation as they require ground truth information about actual time of failure for failures that have not happened yet. As part of future work, we will investigate how to address these challenges. We will also investigate the process for validation of PHM systems.

Acknowledgment

The funding for this work was provided by the NASA System-wide Safety and Assurance Technologies (SSAT) Project.

References

Aguilar, R., Luu, C., Santi, L. M., & Sowers, T. S. (2005). Real-time simulation for verification and validation of diagnostic and prognostic algorithms. In *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit* (pp. 1–8).

Arulampalam, M. S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, *50*(2), 174–188.

Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., & Schnoebelen, P. (2010). *Systems and software verification: Model-checking techniques and tools*. Springer Publishing Company, Incorporated.

Byington, C. S., Roemer, M., Kalgren, P., & Vachtsevanos, G. (2005). Verification and validation of diagnostic/prognostic algorithms. In *Machinery Failure Prevention Technology Conference.*

Daigle, M., & Goebel, K. (2011). A model-based prognostics approach applied to pneumatic valves. *International Journal of Prognostics and Health Management*, *2*(2), 008.

Feather, M. S., & Markosian, L. Z. (2008). Towards certification of a space system application of fault detection and isolation. In *Proceedings of the 2008 International Conference on Prognostics and health management* (pp. 6–9).

Firesmith, D. (2003). Specifying good requirements. *Journal of Object Technology*, *2*(4), 77–87.

Gupta, A. (1993). Formal hardware verification methods: A survey. In *Computer-Aided Verification* (pp. 5–92).

Hicks, B., Larsson, A., Culley, S., & Larsson, T. (2009). A methodology for evaluating technology readiness during product development. In *Proceedings of the International Conference on Engineering Design* (pp. 157–168).

Kalos, M. H., & Whitlock, P. A. (2008). *Monte carlo methods*. John Wiley & Sons.

Mankins, J. C. (1995). Technology readiness levels. *White Paper, April*, *6*.

McMillan, K. L. (2000). A methodology for hardware verification using compositional model checking. *Science of Computer Programming*, *37*(1), 279–309.

Rajamani, R., Saxena, A., Kramer, F., Augustine, M., Schroeder, J. B., Goebel, K., ... Lin, W. (2013).

Guidelines for writing ivhm requirements for aerospace systems. In *Proceedings of the SAE 2013 AeroTech Congress & Exhibition.*

Romero, R., Summers, H., & Cronkhite, J. (1996). *Feasibility study of a rotorcraft health and usage monitoring system (hums): Results of operator's evaluation.* (Tech. Rep.). DTIC Document.

Roychoudhury, I., Hafiychuk, V., & Goebel, K. (2013). Model-based diagnosis and prognosis of a water recycling system. In *IEEE Aerospace Conference* (pp. 1–9).

Saha, B., Koshimoto, E., Quach, C. C., Hogge, E. F., Strom, T. H., Hill, B. L., ... Goebel, K. (2011). Battery health management system for electric uavs. In *IEEE Aerospace Conference* (pp. 1–9).

Saxena, A., Roychoudhury, I., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2012). Requirement flowdown for prognostics health management. In *Proceedings of the AIAA Infotech @ Aerospace.*

Saxena, A., Roychoudhury, I., Lin, W., & Goebel, K. (2013). Towards requirements in systems engineering for aerospace ivhm design. In *Proceedings of the AIAA Infotech @ Aerospace.*

Seo, S., Wallat, M., Graepel, T., & Obermayer, K. (2000). Gaussian process regression: Active data selection and test point rejection. In *Mustererkennung 2000* (pp. 27–34). Springer.

Sommerville, I., & Sawyer, P. (1997). *Requirements engineering: A good practices guide*. John Wiley & Sons.

Tang, L., Saxena, A., Orchard, M. E., Kacprzynski, G. J., Vachtsevanos, G., & Patterson-Hine, A. (2007). Simulation-based design and validation of automated contingency management for propulsion systems. In *IEEE Aerospace Conference* (pp. 1–11).

Wallace, D. R., & Fujii, R. U. (1989). Software verification and validation: an overview. *Software, IEEE*, *6*(3), 10–17.

Yegnanarayana, B. (2004). *Artificial neural networks*. PHI Learning Pvt. Ltd.

BIOGRAPHIES

Indranil Roychoudhury received the B.E. (Hons.) degree in Electrical and Electronics Engineering from Birla Institute of Technology and Science, Pilani, Rajasthan, India in 2004, and the M.S. and Ph.D. degrees in Computer Science from Vanderbilt University, Nashville, Tennessee, USA, in 2006 and 2009, respectively. Since August 2009, he has been with SGT, Inc., at NASA Ames Research Center as a Computer Scientist. Dr. Roychoudhury is a member of the Prognostics and Health Management Society and the IEEE. His research interests include hybrid systems modeling, model-based diagnostics and prognostics, distributed diagnostics and prognostics, and Bayesian diagnostics of complex physical systems.

Abhinav Saxena is a Research Scientist with SGT Inc. at the Prognostics Center of Excellence NASA Ames Research Center, Moffett Field CA. His research focus lies in developing and evaluating prognostics algorithms for engineering systems using soft computing techniques. He is a PhD in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta. He earned his B.Tech in 2001 from Indian Institute of Technology (IIT) Delhi, and Masters Degree in 2003 from Georgia Tech. Dr. Saxena has been a GM manufacturing scholar and is also a member of IEEE, Prognostics and Health Management Society, AIAA, and ASME.

Jose R Celaya is a research scientist with SGT Inc. at the Prognostics Center of Excellence, NASA Ames Research Center. He received a Ph.D. degree in Decision Sciences and Engineering Systems in 2008, a M. E. degree in Operations Research and Statistics in 2008, a M. S. degree in Electrical Engineering in 2003, all from Rensselaer Polytechnic Institute, Troy New York; and a B. S. in Cybernetics Engineering in 2001 from CETYS University, Mxico.

Kai Goebel is Deputy Area Lead of the Discovery and Systems Health Technology Area at NASA Ames Research Center. He also coordinates the Prognostics Center of Excellence. Prior to joining NASA in 2006, he was a senior research scientist at General Electric Corporate Research and Development center since 1997. Dr. Goebel received his Ph.D at the University of California at Berkeley in 1996. He has carried out applied research in the areas of real time monitoring, diagnostics, and prognostics and he has fielded numerous applications for aircraft engines, transportation systems, medical systems, and manufacturing systems. He holds 17 patents and has co-authored more than 250 technical papers in the field of IVHM. Dr. Goebel was an adjunct professor of the CS Department at Rensselaer Polytechnic Institute (RPI), Troy, NY, between 1998 and 2005 where he taught classes in Soft Computing and Applied Intelligent Reasoning Systems. He has been the co-advisor of 6 Ph.D. students. Dr. Goebel is a member of several professional societies, including ASME, AAAI, AIAA, IEEE, VDI, SAE, and ISO. He was the General Chair of the Annual Conference of the PHM Society, 2009, has given numerous invited and keynote talks and held many chair positions at the PHM conference and the AAAI Annual meetings series. He is currently member of the board of directors of the PHM Society and associate editor of the International Journal of PHM.

Table 1. Verification of `ComputeRUL` prognostics technology: an example.

TRL	CPA	ISA	DSE	DS	Testing Activities
1	`ComputeRUL`	Flowchart	N/A	N/A	Evaluate applicability of predictive life estimation towards health management **(Validation)**
2	`ComputeRUL`	Mathematically instantiating the different components in CPA and simulating these analytically	Representative nonlinear equations	Synthesized data	1. Determine that battery discharge can be predicted using available current and voltage data. **(Validation)** 2. Verify that features are available that correlate monotonically to measure fault growth in batteries **(Validation)** 3. Quantify errors and confidence in computed features correlated to fault ground truth data **(Validation)**
3	`ComputeRUL`	CPA implemented in C++ on a generic laptop	Equations of battery of any arbitrary chemistry	Damage progression data obtained from simulations	1. Ensure that uncertainty quantification error, modeling error, discretization error are within allowed limits **(Verification)** 2. Ensure that α-λ Performance, prediction horizon, convergence, etc. metrics from the ISA are within allowed limits **(Validation)**
4	`ComputeRUL`	CPA implemented in C++ on the computer in the battery testbed in the laboratory	Equations of battery of Lithium-Ion chemistry similar to those on-board the e-UAV	Data from Lithium-Ion batteries in the environmental chamber (in laboratory setting) with constant load profiles	1. Ensure that measurement errors, manufacturing variability, channel biases, load profiles are all within allowed limits. **(Verification)** 2. Ensure that α-λ Performance, prediction horizon, convergence, confidence interval, statistical hypothesis testing, reliability metric etc. are within allowed limits. **(Validation)**
5	`ComputeRUL`	CPA implemented in C++ on the computer in the battery testbed in the laboratory	Equations of battery of Lithium-Ion chemistry similar to those on-board the e-UAV	Data from Lithium-Ion batteries in the environmental chamber (in laboratory setting) with varying load profiles	1. Ensure that measurement errors, manufacturing variability, channel biases, load profiles are all within allowed limits. **(Verification)** 2. Ensure that α-λ performance, prediction horizon, convergence, confidence interval, statistical hypothesis testing, reliability metric etc. are within allowed limits. **(Validation)**
6	`ComputeRUL`	CPA implemented in MATLAB running on the computers similar to that on-board the e-UAV	Equations of exact type of batteries of Lithium-Ion chemistry on-board the e-UAV	Played-back data from actual Lithium-Ion battery sensors on-board the e-UAV from multiple ground tests	1. Ensure no coding errors; discretization and sampling rate errors; and communication errors occur. **(Verification)** 2. Ensure that prognostic horizon, computation time, α-λ performance, robustness to system noise, prediction update rate, etc. are within requirements. **(Validation)**
7	`ComputeRUL`	CPA implemented in MATLAB running on the actual computers on-board the e-UAV	Equations of exact type of batteries of Lithium-Ion chemistry on-board the e-UAV	Real-time data from actual Lithium-Ion battery sensors on-board the e-UAV from multiple flight tests with simplistic (safe) flight profiles	1. Ensure that communication errors and delays, code verification, race conditions are all within allowed limits. **(Verification)** 2. Ensure that prognostic horizon, computation time, α-λ performance, robustness to system noise, prediction update rate, etc. are within requirements. **(Validation)**
8	`ComputeRUL`	CPA implemented in MATLAB running on the actual computers on-board the e-UAV	Equations of exact type of batteries of Lithium-Ion chemistry on-board the e-UAV	Real-time data from actual Lithium-Ion battery sensors on-board the e-UAV from multiple flight tests with complex flight profiles and different operating conditions	1. Ensure that prognostic horizon, computation time, α-λ performance, robustness to system noise, prediction update rate, etc. are within requirements. **(Validation)**
9	`ComputeRUL`	CPA implemented in MATLAB running on the actual computers on-board the e-UAV	Equations of exact type of batteries of Lithium-Ion chemistry on-board the e-UAV	Real-time data from actual Lithium-Ion battery sensors on-board the e-UAV during multiple actual science missions	1. Ensure that prognostic horizon, computation time, α-λ performance, robustness to system noise, prediction update rate, etc. are within requirements. **(Validation)**