Software Health Management with Bayesian Networks —Extended Abstract—

Ole Mengshoel
CMU Silicon Valley
Moffett Field, CA 94035
ole.mengshoel@sv.cmu.edu

Johann Schumann
SGT, Inc. / NASA Ames
Moffett Field, CA 94035
Johann.M.Schumann@nasa.gov

***Index Terms*—software health management, fault detection and diagnosis, Bayesian Networks, arithmetic circuits**

I. Introduction

Most modern aircraft as well as other complex machinery is equipped with diagnostics systems for its major subsystems. During operation, sensors provide important information about the subsystem (e.g., the engine) and that information is used to detect and diagnose faults. Typically, FDDR (fault detection, diagnosis, and recovery) or IVHM (Integrated Vehicle Health Management) systems are used for this purpose.

Most of these systems focus on the monitoring of a mechanical, hydraulic, or electromechanical subsystem of the vehicle or machinery. Only recently, health management systems that monitor *software* have been developed (for an overview see, e.g., [1]). In this paper, we will discuss our approach of using Bayesian networks for Software Health Management (SWHM) [2], [3], [4].

The field of system health management for hardware is quite mature; many industrial systems use diagnostics/IVHM systems (e.g., automotive or aerospace industry). However, the health management of *software* has to adhere to substantially different requirements. The most striking difference is that faults in a software system usually occur instantaneously, whereas faults in hardware systems tend to develop over time (e.g., an oil leak). [1] Furthermore, many software problems are caused by problematic software-hardware interactions, which means that both the software and the hardware must be monitored.

At the same time, software has features that might make system health monitoring easier and more promising in some ways. First, software redundancy does not increase the weight of a system, while hardware redundancy clearly does. Second, software can be debugged and fixed remotely, without need for human presence at the location where the system (say, a robotic vehicle on Mars) is deployed.

Based on the brief discussion above, it is clear that software has several unique features that makes a dedicated research and development effort worthwhile. At the same time, it is also important to utilize and extend existing results from the area of system health management. In our SWHM approach, briefly presented here and discussed in more detail elsewhere [2], [3], [4], we are using Bayesian networks [5], [6] to define the health model for the software to be monitored. In the rest of this extended abstract, we will first discuss SWHM requirements, which make advanced reasoning capabilities for the detection and diagnosis important. Then we will present, on a high level, how our Bayesian models are constructed.

II. SWHM Requirements

Traditional FDDR and IVHM systems are tied to the individual components or subsystems they monitor. Based upon sensor readings, such a system tries to detect, for a component or subsystem, anomalous behavior and if such behavior is found, produces a diagnostics message. While in many cases such an approach is reliable, adverse effects that have been caused by the interaction between different subsystems or components cannot be captured properly. A typical example is a recent incident on a Qantas A380. When one of the engines exploded during flight, taking out the hydraulic system and damaging the wing, the pilots had to sort though literally hundreds of diagnostic messages in order to find out what happened. In addition, several diagnostic messages contradicted each other[2]. If the diagnostics had been system-wide, the number of warnings (and thus the pilot's workload) could have been reduced tremendously and no contradictory diagnostic messages would have been produced. Furthermore, emergent behavior can only be detected if information from all subsystems can be taken into consideration.

The problem of interaction between components or subsystem, as discussed above, is an subclass of a broader class of problems: A specific set of observations could have been caused by a number of different, potentially contradictory

[1] This is a rule of thumb, with exceptions. A memory leak, for example, is a type of software fault that develops over time and as such is a an exception to our rule.

[2] http://www.aerosocietychannel.com/aerospace-insight/2010/12/exclusive-qantas-qf32-flight-from-the-cockpit/

faults. The SWHM should be able to distinguish those and provide a metric on how confident the SWHM is that a certain fault has actually occurred.

Many approaches to diagnostics and IVHM use discrete models and do not properly account for sensor failure; diagnostic messages are often produced using table-driven or fault-tree based mechanisms. The input of such systems are most often discretized sensor values (e.g., pressure_low, pressure_hi) and the reasoning uses one or more "firing" diagnostic rules. However, those approaches usually do not take into account that sensors, which produce the input to the IVHM, might return noisy data or can be broken altogether. Advanced SWHM, however, should be able to reason about sensor reliability and quality of sensor data.

Finally, for real-time and embedded systems there are requirements for SWHM, like other types of system health management, to have predictable and short execution times and not use much memory [7]. A more general requirement is ease of modelling, either by supporting machine learning or automated Bayesian network construction from a domain-specific language.

III. Bayesian Networks for SWHM

Bayesian networks are an approach to represent multivariate probability distributions in a compact manner such that they are amendable to learning and inference [5], [6]. In our recent work, we have successfully compiled Bayesian networks to arithmetic circuits. These arithmetic circuits are then used to perform, by an on-line evaluator, system health management functions including detection and diagnosis.

In our SWHM approach, we use Bayesian networks to model software [2]. Our modelling of software is inspired by previous work on system health management for electrical power systems, in which each electrical power system component is represented by a small number of nodes (typically 2-6), and then separate Bayesian networks structures represent the connections between components. In a similar way, each software component is in our approach represented by a small number of nodes, one of which represents the "health status" of the software. Currently, we have initial results for software for small satellites, specifically for a simplified aircraft guidance, navigation, and control (GN&C) system implemented using the OSEK[3] embedded operating system [2]. Using scenarios with injected faults, we have shown that that our SWHM approach using Bayesian networks is able to detect and diagnose software faults.

In our demonstration, we have implemented the SWHM concept using Bayesian networks, which hare model software as well as interfacing hardware sensors. Of particular interest to us is that this approach can fuse information from different layers of the software stack, from firmware to operating systems and application software. After compilation to arithmetic circuits, Bayesian networks are well-suited for on-line execution in embedded software systems found in vehicles (aircraft, spacecraft, and cars) or mobile devices (cell phones, tablets, etc.)

IV. Conclusion

Software plays an important and increasing role in aircraft and other complex machinery. Unfortunately, software can fail in spite of extensive verification and validation efforts. In this paper, we discuss a Software Health Management (SWHM) approach to tackle problems associated with software bugs and failures. We have briefly presented a SWHM system that can help to perform fault detection and diagnosis in embedded systems, using Bayesian networks as the underlying modeling paradigm. In these networks, we concisely capture and fuse information from hardware sensors, software status signals, software quality signals, and information from the operating system. Given these data, Bayesian reasoning can compute the most likely causes of failures, if present, and also give a statistically sound measure for the quality (probability) of the answer.

System health models in the form of Bayesian networks can be compiled into efficient arithmetic circuits, which yield a high-performance SWHM and are suitable for execution within embedded (on-board) software systems, and are amenable to V&V [8].

Furthermore, software tools for Bayesian modeling and compilation into arithmetic circuits—such as SamIam[4] and Ace[5]—are readily available.

In this abstract, we only covered a small range of a SWHM system's capabilities. Current work investigates, how information on the quality of a computation (e.g., numerical quality or quality of the state estimation) can be smoothly incorporated into the SWHM. Research on hierarchical SWHMs will address the issue of detecting complicated software-hardware interactions for large- and extreme-scale BNs [9], and will focus on the fusion of multiple information streams for the purpose of increasing diagnostic accuracy. can deal with unexpected and unmodeled failures (e.g., due to unforeseen environmental circumstances) and emerging behavior. Bayesian networks have, due to their modeling capabilities, its efficient execution, and high reasoning power, to find their way into on-board software health management.

Acknowledgment

This work is supported by a NASA NRA grant NNX08AY50A "ISWHM: Tools and Techniques for Software and System Health Management".

References

[1] G. Karsai, Ed., *1st International Workshop on Software Health Management (SMH 2009)*. ISIS, Vanderbilt University, 2009. [Online]. Available: http://www.isis.vanderbilt.edu/workshops/smc-it-2009-shm

[2] J. Schumann, O. Mengshoel, and T. MBaya, "Integrated software and sensor health management for small spacecraft," in *Proc SMC-IT*, 2011.

[3] We are currently using the Trampoline implementation of OSEK—see http://trampoline.rts-software.org for details.

[4] http://reasoning.cs.ucla.edu/samiam/

[5] http://reasoning.cs.ucla.edu/ace/

[3] A. Srivastava and J. Schumann, "The case for software health managment," in *Proc SMC-IT*, 2011.
[4] J. Schumann, T. MBaya, and O. Mengshoel, "Bayesian software health management for aircraft guidance, navigation, and control," in *Proc. PHM 2011 (submitted)*, 2011.
[5] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
[6] A. Darwiche, *Modeling and Reasoning with Bayesian Networks*. Cambridge, UK: Cambridge University Press, 2009.
[7] O. J. Mengshoel, M. Chavira, K. Cascio, S. Poll, A. Darwiche, and S. Uckun, "Probabilistic model-based diagnosis: An electrical power system case study," vol. 40, no. 5, pp. 874–885, 2010.
[8] J. Schumann, A. Srivastava, and O. Mengshoel, "Who guards the guardians? — toward v&v of health management software (short paper)," in *Runtime Verification 2010*. Springer, 2010.
[9] K. Przytula, G. Isdale, and T.-S. Lu, "Collaborative development of large Bayesian networks," in *Proc. of the 2006 IEEE Autotestcon*, 2006, pp. 515–522.