

Characterizing Environmental Information for Monitoring Agents

Albert Esterline¹, Bhanu Gandluri², and Mannur Sundaresan²

¹ Department of Computer Science

² Department of Mechanical and Chemical Engineering

North Carolina A&T State University, Greensboro, NC.

esterlin@ncat.edu, bpgandlu@ncat.edu, mannur@ncat.edu

Abstract. A multiagent architecture for vehicle and structural health monitoring is proposed. A prototype using this architecture was developed using JADE. Critical aspects of the design were verified using the SPIN model checker. The tasks in our framework are related to data-fusion levels and Gibson's realist position on direct perception of objects and affordances. We show how a system consisting of a multiagent system along with the monitored platform exhibits behavior at several levels, from the physics of acoustic emissions to inter-agent conversations expressing desires and beliefs. Communication, perception of public events, and system design conspire to provide the common knowledge needed to coordinate diagnostic tasks.

1 Introduction

The rising costs of current maintenance paradigms for expensive, safety-critical, or mission-critical structures motivates the search for health monitoring systems that can determine the status of their platforms in near real time. An architecture for such a system must address highly dynamic relations among computational components. We propose a multiagent architecture for vehicle and structural health monitoring. The next section presents background on this architecture. The need and requirements for health monitoring are outlined, the general nature of multiagent systems is discussed, and applications of agent technology to vehicle health monitoring are reviewed. Since our prototype system uses JADE, it and the FIPA reference architecture are outlined as well as the contract net protocol, which provides flexibility while respecting agent autonomy. Section 3 presents our architecture in terms of the kinds of agents and their interactions. Since the behavior of non-trivial concurrent systems is too complex and uncontrollable to be thoroughly tested, formal methods have been developed to verify their designs. One such method, model checking, is used in the current study and is discussed in section 4. Section 5 discusses the implementation of our prototype health monitoring system. In section 6, we relate the tasks in our framework to data-fusion levels and Gibson's realist position on direct perception of objects and affordances. We show how a system consisting of a multiagent

system and the monitored platform exhibits behavior at several levels. Communication, perception of public events, and system design conspire to provide the common knowledge needed to coordinate diagnostic tasks. Section 7 concludes and suggests future work.

2 Background

2.1 Vehicle Health Monitoring Systems

Restoration maintenance involves replacing a part only when it has completely failed, but this practice leads to overall deterioration of system performance. Use- or schedule-based maintenance is preventative maintenance done after certain hours of service or a period of time [1]. Worst case scenarios are taken into account and there is wastage of useful components [2]. These practices incur huge costs involving maintenance of inventory, quality checking of removed parts, and labor [3]. Such costs are driving the research towards a system whose status can be accurately determined in real time. The system must identify any deterioration in the structure before it becomes catastrophic [4]. In particular, maintenance is based on the condition of the structure rather than the hours of service. The shift from reliability-centered to condition-based maintenance has led to structural health monitoring, where a structure is diagnosed continuously and remedial action is taken as needed. Safety and reliability are added to the structure because of an efficient and reliable health monitoring system [1].

For diagnosis, certain other tasks have to be done first: data collection, signal processing, feature extraction and classification [1]. Various techniques are used for each of these tasks. Data collection is done by various kinds of sensors and data acquisition equipment. Signal processing is done by various hardware- and software-based techniques. Feature extraction and classification use various analytical and non-analytical techniques. As these systems evolve, the coordination of the sensing, communication, feature extraction, and diagnosis will be critical [5]. An architecture that addresses this diversity of sensors and techniques as well as the huge amounts of data is needed.

2.2 Multiagent Systems for Vehicle Health Management

Wooldridge [6] defines an agent as a computer system capable of autonomous action that meets its design objective in the environment in which it is situated. An intelligent agent in particular is capable of flexible autonomous action. The characteristics of multiagent environments identified by Huhns and Stephens [7] are that they “provide an infrastructure specifying communication and interaction protocols,” they are typically open, and the agents are autonomous and distributed. Huhns and Stephens also see coordination as critical and, in the case of self-interested agents (where negotiation is involved), as a major challenge.

Some studies on structural health monitoring mention agents as an immediate future trends [4]. Agents have been developed for patient monitoring [8], airport

gate assignment [9], and machinery health monitoring [10]. Such agents are decision modules that use knowledge-bases, neural networks, and similar techniques. Agents are used for structural health monitoring in Lockheed Martin's MENSA architecture [11]. While, however, there is some cooperation among these agents, the interaction among them is not principled and control flow is rigid. To enjoy advantages of multiagent systems such as flexibility and fault-tolerance, a team of agents must use a protocol to structure their activity in a collaborative and flexible way [5].

2.3 Contract Net Protocol

The contract net protocol [12] is a protocol for high-level distributed problem solving whose efficiency has been demonstrated in flexible manufacturing [13] and other fields. An agent has a task that it decomposes into subtasks for which it becomes the manager. For each subtask, the manager broadcasts a task announcement. Each agent evaluates the task description in the announcement; if it wants the task, it sends a bid to the manager. The manager uses the descriptions in the bids to rank them and sends an award to the agent with the winning bid, now known as the contractor. When the contractor completes the subtask, it sends a final report to the manager. When all final reports are in, the manager combines the results to give the result for the overall task. Since tasks themselves may have subtasks, a given agent may be both a contractor (of one task) and a manager (of another). This gives rise to an ad hoc control hierarchy structured to perform the top-level task. Manager-contractor communication that occurs between sending the award and returning the report allows the contract net protocol to be used in two complementary ways: in initialization, the protocol is used to select agents to be responsible for subtasks in an ongoing task; in operation, the protocol allows agents performing ongoing tasks to handle specific problems on the fly. As another enhancement, if a set of potential contractors is narrowed in advance, focused addressing rather than broadcasting may be used.

2.4 FIPA Reference Architecture and JADE

FIPA (Foundation for Intelligent Physical Agents) [14] is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies. The FIPA reference architecture requires an agent management system, a directory facilitator, and a message transport system. A FIPA ACL (Agent Communication Language) message contains one or more parameters, such as the performative, content, ontology, and content language. JADE [15] is an open source tool to develop multiagent systems using an API provided through Java classes. It provides the runtime environment for the agents, is easily distributed across several machines, and adheres to FIPA standards. In particular, it provides classes to implement the contract net protocol. JMatLink [16] is open source software that provides an API for invoking Matlab commands from Java applications. All Matlab programs

used in our prototype (for example, neural networks) are accessed from JADE agents via wrappers that we have developed.

3 Architecture

Our multiagent framework is pictured in Figure 1. None of the implementations we envision involves mobile agents, and the fact that agents may be associated with locations on the monitored platform does not force any particular distributed design. There are system agents and data-management agents, which provide services but generally do not participate in the contract net protocol. Supervisor agents are bound to one or more sensors and announce diagnostic tasks when they detect suspicious values. Monitor agents coordinate the diag-

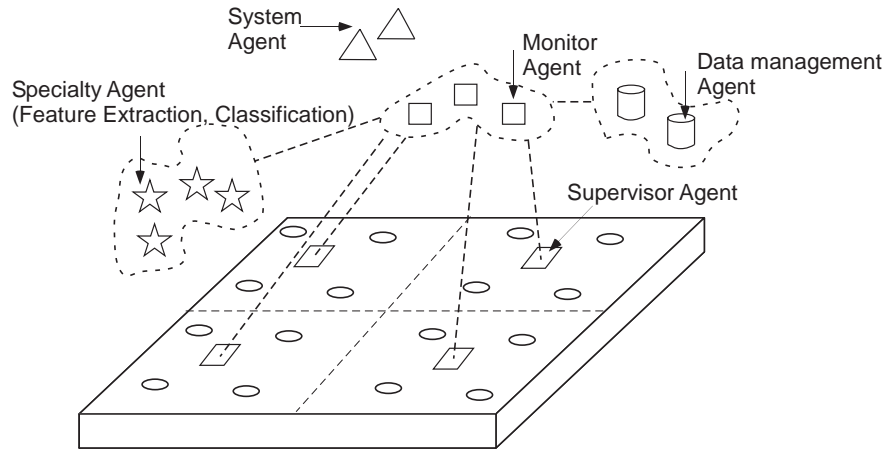


Fig. 1. Line sketch of Vehicle Health Monitoring System

nosis tasks. By participating in the contract net protocol, a monitor agent may become responsible for several supervisor agents. Other monitors may supervise the monitors that directly supervise the supervisor agents, and so on, resulting in a hierarchical configuration. Monitors focus on regions that warrant more attention and refocus as the situation evolves. A supervisor agent announces a diagnosis task to all monitors above it in the hierarchy. A specialty agent wraps a procedure for a specific task within an agent interface. In the current prototype, specialty agents include feature-extraction agents, classification agent, and location agents. In the applications we address, a specialty agent should include at least three things in its bids: an estimate of the reliability of its result, an estimate of the time it would take to produce the result, and an estimate of the bandwidth needed to produce the result. The monitor can use these estimates to make awards in a way sensitive to the current state of the system and the task at hand.

4 Model Checking

Concurrent systems present problems for testing since the behaviors of the separate threads can be interleaved in many ways, and interactions between threads can cause the effects of inputs to cascade in complex ways. A tester's ability to exercise the behavior of concurrent systems is thus limited. The alternative to testing a system is formal verification of its design. Properties required to hold of the system (its specification) are formulated in an appropriate formalism, and the design, a blueprint for generating the system's behavior, is formulated in a less abstract notation. Then one shows rigorously that the design does not violate the specification. One such method is model checking, where the specification properties are expressed in a temporal logic and the design is expressed as one or more finite state automata. One checks that the automata, taken as an interpretation of the properties, form a model for those properties, i.e., an interpretation in which all are true. The first subsection here describes the SPIN model checker [17], and the remaining subsections describe two abstract designs we verified with SPIN along with the specifications used.

4.1 The SPIN Model Checker

SPIN's design language is PROMELA, which has asynchronous processes (threads of control), buffered message channels, and structured data. There are two ways for processes to synchronize. One process may block on an expression until it becomes true because of the action of another process. And, when one process attempts to input from a channel with a zero-length buffer (a rendezvous channel), it must wait until another process outputs on that channel; likewise, output on a rendezvous channel must wait for input. PROMELA semantics involves non-determinism at both the statement and process-scheduling levels and there is no notion of time. So design models avoid implementation detail and focus on the assumptions made within each module about interaction with others. This abstract nature is necessary since the verification procedure must check each state to verify the specification properties, and abstraction defeats complexity by drastically reducing the number of states.

Correctness claims can be inserted into a PROMELA model. For example, a *never* claim checks system properties after each statement execution for behavior that should never occur. A more perspicuous way to express SPIN specifications is with Linear Time Temporal Logic (LTL), which has two kinds of formulas: state formulas and LTL formulas. The truth value of a state formula is determined at an extended system state, which is a control state with a unique combination of values for the system variables. An LTL formula, in contrast, is evaluated only for complete runs. A state formula contains no LTL operators while an LTL formula contains state formulas and LTL operators. Regarding the operators, $p \ U \ q$ is true if p is true until q becomes true (after which p may or may not be true), $\Diamond p$ is true if p is eventually true (true now or sometime in the future), $\Box p$ is true if p is always true (true now and at all times in the future), and Xp is true if p is true at the next time. To show that a property is

not violated, negate its formula. SPIN translates the LTL formula into a *never* claim that is included with the PROMELA system model, and the automata are combined. SPIN then tries to find runs satisfying the LTL formula, that is, it tries to find a falsifying behavior for the original property.

4.2 Verifying Concurrent Activity of Specialty Agents

We assume that the contract net protocol and FIPA infrastructure behave correctly and model synchronization directly between agents. Two PROMELA models were constructed. The first, called Operation, addressed the operation of monitor agents. A second, called Initialization, addressed hierarchy issues related to the establishment of a hierarchy. The intended behavior for Operation is depicted in Figure 2. We consider only one monitor agent and one supervisor agent. An event detected by the supervisor agent always results in an award to the monitor agent. We assume that the monitor agent finishes classifying and locating an event before the next one is detected. When the monitor agent finishes with one event, it waits for the next, and this carries on indefinitely. The monitor agent must manage four subtasks: feature extraction for classification, feature extraction for location, classification, and location. In this model, there is only one agent for each of the feature-extraction tasks, but there are two agents each for classification and location. Thus, the selection afforded by the contract net protocol comes into play only for deciding between the classification agents and for deciding between the location agents. The symmetry of the situation does not warrant introducing further possibilities. We want an abstract design with

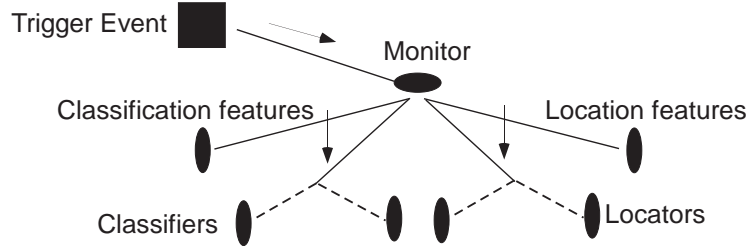


Fig. 2. Verifying Concurrent Activity of Specialty Agents

several properties. First of all, diagnosis should complete indefinitely often. Also, whenever the extraction of features for classification is finished, classification itself should eventually finish. Finally, if classification has not been completed, then it should not complete until feature extraction for it has completed. We can express these specifications in a mixture of LTL and English as follows:

- ◇ *Diagnosis complete*,
- (*Feature extraction done* → ◇ *Classification done*),
- (! *Classification done* → (! *Classification done* U *feature extraction done*))

The specifications for classification and the feature-extraction tasks it depends on could be repeated for location, but symmetry makes one case representative.

The PROMELA model has ten processes representing eight agents; two dispatch processes represent aspects of the behavior of the monitor agent. All the processes execute infinite loops. Sequential constraints on the executions of processes are mostly established with boolean variables used as guards. When a process finishes its work for the current event, it sets its controlling variable to false and sets the variables for its one or more sequels to true. To express the above specifications in a form that can be used by SPIN, we must replace the English expressions with expressions over PROMELA variables. Variable *diagend* is initially true and becomes true when the diagnosis of an event finishes; it enables the supervisor process, which sets it to false. *finish1* is set to true by the feature extraction process for classification; it enables classification and is set to false by the monitor just before it enables classification. Finally, *done1* is set to true by the dispatch process for classification and is set to false by the monitor just before it waits for the next event. The specifications, then, are

- $\square \diamond \textit{diagend}$,
- $\square (\textit{finish1} \rightarrow \diamond \textit{done1})$,
- $\square (! \textit{done1} \rightarrow (! \textit{done1} U \textit{finish1}))$.

4.3 Establishing a Monitor Hierarchy

Regarding Initiation, when a monitor agent gets a task from a supervisor agent, it may pass on the task to another monitor agent that handles a higher task. The communication in the contract net protocol and the tasks of the monitor agents are abstracted out to capture just the establishment of the hierarchy and ensuring that the hierarchy is maintained. Following the three cases in Figure 3 from left to right, initially the supervisor has a task that it announces to three monitor agents. One monitor is awarded the task, and it in turn announces the task to the remaining two monitors, one of which is awarded the task. One monitor is free, left out of the hierarchy. We want to guarantee that the hierarchy persists. One specified property is that, if the hierarchy is established, then the free monitor never gets the award. Another is that, whenever the hierarchy is established, then the top monitor eventually is awarded the task. We can express these properties in English as:

- Hierarchy set* $\rightarrow \square ! \textit{Free agent gets award}$,
- $\square (\textit{Hierarchy set} \rightarrow \diamond \textit{Parent monitor handles the task})$.

The PROMELA model consists of four processes, each representing an agent. One is the supervisor. There are three monitors, all instances of the same process type (with the same code). All processes execute infinite loops. Most of the synchronization is achieved with rendezvous channels. The process identifiers (pids) of processes are used for identification. In the chain of awards that sets up the hierarchy, monitor processes pass their pids with their reports, which allows them to be identified for subsequent awards. Selection of monitor process for award is non-deterministic during setup. Several global variables are maintained for model checking. Boolean variable *sbsq* indicates the beginning of any diagnostic

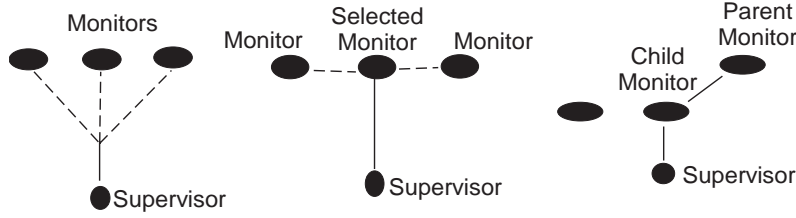


Fig. 3. Establishing a Monitor Hierarchy

task except the first. It is initially false, is set to true each time but the first that the supervisor executes, and is set to false when the child monitor receives an award. Variable *freepid* is set to the pid of the free monitor when it receives a reject during setup, and *toppid* is set to the pid of the parent monitor during setup. Finally, *curpid* is set to the pid of the parent monitor when it receives a reward after setup. With these variables, we can now express the above LTL specifications as: $sbsq \rightarrow \square ! p$ and $\square (sbsq \rightarrow \diamond q)$, where p is defined to be $curpid == freepid$ and q is defined to be $curpid == toppid$.

5 Prototype

We have implemented a prototype of our architecture using JADE, Matlab, and JMatLink. It uses acoustic emission (AE) signals to classify events related to fatigue crack growth in materials. It has seven agents: a supervisor agent, two feature extraction agents, two classification agents, and two monitor agents. Current work is adding location agents and the necessary additional feature-extraction agents. Various parameters (features) of AE signals are used to classify their sources. AE signals from mode I fatigue crack growth normal to the direction of the load axis in fabric composite materials as well as modes I and II delamination growths are considered. Simple coupon specimens of glass fabric/epoxy composite laminate are used to generate the different types of AE signals [18]. Specimens are loaded in an MTS (material testing system) machine and the AE signals are recorded with a digital oscilloscope. The winning feature-extraction agent runs a Matlab script on the waveforms thus obtained to extract eight features, namely, symmetric amplitude and duration and asymmetric amplitude and duration for both shear and normal sensors. Two different feature-extraction agents are obtained by different instantiations of the feature extraction agent class. The values of these features are supplied to the winning classification agent, which runs a neural network for which supervised training was used. Classification identifies one of the three failure modes of the composite material. Two different classification agents were obtained by using different sets of training data.

Concerning the overall structure and operation of the prototype, each agent registers itself with the Directory Facilitator of the JADE runtime environment. An agent looking for a service finds the agents providing it, allowing the contract

net protocol to use focused addressing. In the prototype, a potential contractor responds to an announcement with a randomly-generated number thought of as its estimate of the time to perform the announced task. A diagnosis task is initiated when the user provides the supervisor agent with an ordinal number identifying the data set for the simulated fault; this action is the triggering event for the supervisor agent. The supervisor agent then selects a monitor agent through the contract net protocol. With the award, the supervisor agent passes to the monitor agent the ordinal. The monitor agent then announces a feature-extraction task, and the selected feature extraction agent is passed the ordinal of the data set. The feature extraction agent runs a Matlab script on the data set and stores the feature values in a file whose name is reported back to the monitor agent, which then announces a classification task. The selected classification agent connects to Matlab, inputs the file with feature values to the trained neural network, and reports the classification back to the monitor agent, which reports it to the user.

6 Agent Ecology and Societies

A health monitoring system benefits from sensors distributed over the structure to whose data we can apply data fusion, “the process of combining data or information to estimate or predict entity states [19]”. The OODA model was an early data-fusion model that divided the fusion process into four stages: observe, orient, decide, and act. The waterfall model represented a bottom-up tree from sensor observations to decision making. Both were too rigid and circumscribed [20] and have been largely superseded by the JDL Data Fusion Model [19], a functional model that does not dictate a processing sequence. Instead, it recognizes five functional levels:

- Level 0 (Signal Refinement) involves sub-object data assessment that addresses pixel/signal-level data.
- Level 1 (Object Refinement) estimates and predicts entity states and includes tracking and classifying objects.
- Level 2 (Situation Refinement) infers relations among entities
- Level 3 (Threat Refinement) assesses the potential impact of the situation based on a priori knowledge and prognosis.
- Level 4 (Process Refinement) manages resources to give a form of adaptive data acquisition and processing to support mission objectives.

In health monitoring, it is often more natural to speak of events rather than objects. The levels of the JDL model correspond nicely to the functions of our multiagent health monitoring framework. At Level 0, we have feature extraction, at Level 1, local classification and location, at Level 2, more global classification and location (involving hierarchies of monitor agents), and, at Level 3, prognosis. With a multiagent system, Level 4 occurs spontaneously and continuously through negotiation.

The JDL model embodies some strong realist assumptions, especially in distinguishing between Level 0, where information is handled without regard to its referents, and Level 1, where events and objects in the real world are located and classified. Despite this realism, much of the data-fusion literature pays little regard to real-world referents. As an antidote to this myopia, we suggest a realist picture of perception, such as Gibson’s ecological theory of visual perception [21]. What we directly see according to Gibson is the layout of objects by virtue of the stimulus information in the structure of ambient light. This realist position relates to JDL Level 2. A central concept for Gibson is that of the *affordances* of the environment: what behavioral possibilities it furnishes for an animal. In perceiving affordances, Gibson maintains, we perceive values: some offerings are beneficial, some injurious. This realist interpretation of Level 3 is particularly relevant to health monitoring.

We have agents distributed on a platform, perceiving features of events taking place in it and collaborating through speech acts (messages with performatives) to arrive at a diagnosis. To disentangle the threads here, consider Dennett’s account of various strategies used to predict the behavior of systems [22]. The physical strategy views a system in terms of the physical sciences. If a system is too complex for the physical strategy, we may use the design strategy: predict that the system will behave as it is designed to behave. Finally, when even the design stance is impractical, there is still the intentional stance, treating the system as a rational agent: determine its beliefs and desires, then predict behaviors that will further its goals. Although the intentional stance (applied to computational entities) has been a cornerstone of agent research, our focus is different from Dennett’s. While Dennett is concerned with the status of beliefs, we are interested in agent coordination hence in (socially situated) communication acts. (Still, such acts express beliefs and desires.) All three strategies dovetail together in our framework. The physical strategy is appropriate for addressing physical events and materials, the design strategy applies because a design is instantiated in the structure and known by the agents (in fact, it provides affordances), and the intentional strategy comes in because we are concerned with communication acts that express beliefs and desires.

Focusing on the social aspect, the key issue is how the agents coordinate to reach a diagnosis given the events they perceive together and the information they share about the design of the platform. Now, common knowledge can be shown to be a necessary condition for coordination [23]. Where ϕ is a proposition and G is a group, it is common knowledge in G that ϕ if everyone in G knows that ϕ , everyone in G knows that everyone in G knows that ϕ , and so on for arbitrarily deep nestings of the operator “every one in G knows that.” Clark and Carlson [24] have presented heuristics for inferring common knowledge in terms of situations shared by agents A and B . The physical co-presence heuristic comes into play when an object is located between A and B and both see (or otherwise perceive) the object and each other simultaneously. With the linguistic co-presence heuristic, there is a triple co-presence of A , B , and the linguistic positing of the object of common knowledge. Finally, by the community

membership heuristic, if A finds that B is in the same community as A , then A may assume that B has that community's common knowledge. All three of these heuristics apply to our agents. Community membership is designed into the system, and the agent communication language is designed to achieve linguistic co-presence. Physical co-presence applies because several agents can perceive an event and, by design or arrangement, they have information about their neighbors' locations. Realism regarding perception makes it appropriate to apply this heuristic. Model checking fits nicely into this picture: the synchronization that is central to the abstract software design being verified is an abstraction from the system that is the object of the common knowledge that makes the coordination among the agents possible.

7 Conclusion

We formulated a multiagent structural health monitoring architecture that includes system agents, data-management agents, and alarm-raising supervisor agents bound to groups of sensors. There are also monitor agents, which coordinate the diagnosis task and configure themselves hierarchically. Specialty agents provide specific services by incorporating specialized software. The contract net protocol is used both to assign responsibilities and to assign problems on the fly. Since this highly concurrent architecture is too complex and uncontrollable for thorough testing, we used model checking to verify abstract designs of critical aspects against LTL specifications. Finally, a prototype to classify events related to fatigue crack growth was built. We related the tasks in our framework to data-fusion levels and Gibson's realist position on direct perception of objects and affordances. We showed how a system consisting of a multiagent system and the monitored platform exhibits behavior at several different levels, from the physics of acoustic emissions to inter-agent conversations expressing desires and beliefs. Communication, perception of public events, and system design conspire to provide the common knowledge in the agent community needed to coordinate diagnostic tasks. We intend to work on a multi-level interpretation with systematic integration of social and ecological concepts with each other and with concepts at the physical and design levels. This will reduce the conceptual distance between the physical events and awareness of the impact of the current situation on the success of the mission. Future work will also involve more extensive prototypes, and more aspects of the architecture will be model checked.

References

1. Garga, A., Campbell, R., Byington, C., Kasmala, G., Lang, D., Lebold, M., Banks, J.: Diagnostic reasoning agents development for HUMS systems. In: American Helicopter Society 57th Annual Forum, Washington, D.C. (2001) 1268–1275
2. Garga, A., McClintic, K., Campbell, R., Yang, C., Lebold, M.: Hybrid reasoning for prognostic learning in CBM systems. In: IEEE Aerospace Conference, Big Sky, MT (2001) 2957–69

3. Roemer, M., Vachtsenavos, G., Byington, C., Kacprzyński, G.: Two-day PHM/CBM design course (2004) Orlando, FL.
4. Faas, P., Schroeder, J., Smith, G.: Vehicle health management research for legacy and future operational environments. *Aerospace and Electronic Systems Magazine IEEE* **17** (2002) 10–16
5. Esterline, A., Gandluri, B., Sundaresan, M., Sankar, J.: Verified models of multiagent systems for vehicle health management. In: 12th SPIE Annual International Symposium on Smart Structures and Materials, San Diego, CA (2005)
6. Wooldridge, M.: *Intelligent Agents*. In: *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT press, Cambridge, MA (2000) 27–78
7. Huhns, M., Stephens, L.: *Multiagent Systems and Societies of Agents*. In: *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press (2000) 79–120
8. Mabry, S., Schneringer, T., Eppers, T., Edwards, N.: Intelligent agents for patient monitoring and diagnostics. In: *Proceedings of 2003 ACM Symposium on Applied Computing*, Melbourne, FL (2003) 257–262
9. Lam, S., Cao, J., Fan, H.: Development of an intelligent agent for airport gate assignment. *Journal of Airport Transportation* **7** (2002) 103–114
10. Logan, K.: Prognostic software agents for machinery health monitoring. In: *IEEE Aerospace Conference*. (2003) 3213–3225
11. Franke, J., Satterfield, B., Czajkowski, M., Jameson, S.: Self-awareness for vehicle safety and mission success. Technical report, Lockheed Martin Advanced Technology Laboratories, Camden, NJ (2002)
12. Smith, R.: The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* **C-29(12)** (1980) 1104–1113
13. Shen, W., Norrie, D.: An agent-based approach for dynamic manufacturing scheduling. In Nwana, H., Ndumu, D., eds.: *3rd Int. Conf. on the Practical Applications of Agents and Multi-Agent Systems*, London, UK (1998) 533–548
14. FIPA: Welcome to FIPA! <http://www.fipa.org/> (2005)
15. TILab: Java agent development environment. <http://jade.tilab.com/> (2005)
16. Mller, S.: JMatLink: Matlab Java classes. <http://www.held-mueller.de/JMatLink> (2005)
17. Holzmann, G.: *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley, Boston, MA (2004)
18. Sundaresan, M., Nkrumah, F., Grandhi, G., Derriso, M.: Identification of failure modes in composite materials using a continuous ae sensor system. In: *IMECE 2004: ASME Int. Mechanical Engineering Congress*, Anaheim, CA (2004)
19. Steinberg, A., Bowman, C.: Revisions to the JDL Data Fusion Model. In: *Handbook of Multisensor Data Fusion*. CRC Press, Washington, D.C. (2001) 2–1–18
20. Elmenreich, W.: *Sensor Fusion in Time-Triggered Systems*. PhD thesis, Technical University of Vienna, Vienna, Austria (2002)
21. Gibson, J.: *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, Hillsdale, NJ (1986)
22. Dennet, D.: True Believers: The Intentional Strategy and Why It Works. In: *The Intentional Stance*. The MIT Press, Cambridge, MA (1987) 13–35
23. R. Fagin, J. Y. Halpern, Y.M., Vardi, M.Y.: *Reasoning About Knowledge*. MIT Press, Cambridge, MA (2003)
24. Clark, H., Carlson, T.: *Speech Acts and Hearers' Beliefs*. In: *Mutual Knowledge*. Academic Press, London (1982) 1–36