© Nottingham University Press
Proceedings of the International Conference on
Computing in Civil and Building Engineering
W Tizani (Editor)

Agent-oriented development of hybrid wind turbine monitoring systems

K. Smarsly
Stanford University, Stanford, CA, USA

D. Hartmann
Ruhr-University Bochum, Bochum, Germany

Abstract

Structural health monitoring is a growing field of research and of paramount practical relevance. Backed by the recent advances in engineering technologies and computing methods, structural health monitoring has the potential to reliably identify damages and deteriorations of structures, at an early stage, resulting in significantly reduced repair and maintenance costs. Considering the emerging field of wind turbine monitoring, this paper presents an innovative hybrid (i.e. software-hardware) monitoring system based on multi-agent technology. Since this technology represents a novel approach towards developing fully decentralized and autonomous monitoring systems, adequate agent-oriented development strategies for structural health monitoring have not yet been proposed. Thus, a suitable development methodology for agent-based monitoring systems is introduced in this paper, which is exemplified by means of a prototype of a wind turbine monitoring system (WTMS). Focusing mainly on the software part of the WTMS prototype, various re-usable system models being part of the methodology are to be illuminated in detail. To this respect, requirements analysis, design, implementation and application of the monitoring system will be demonstrated. Also, an evaluation of the already implemented prototype using actual field data obtained from a wind turbine in operation is carried out.

Keywords: wind turbine, monitoring, multi-agent, agent-oriented, smart structures, autonomous

1 Introduction

Wind energy technology is one of the fastest growing industrial sectors worldwide. Providing a relatively cheap and clean energy source, wind energy has become a strong contender in the global market being competitive with conventional sources. Driven by environmental as well as economic developments, substantial progress has been made in wind energy research in the last years. For example, the power output of a state-of-the-art wind turbine has increased by a factor of more than 200 since the beginning of the 1980s (see Fig. 1, EWEA 2009a). However, the recent advancements in the wind energy sector are resulting in sophisticated technological challenges requiring innovative solutions. In this context, wind turbines need to be observed permanently in order to ensure a reliable operation and to avoid expensive parking times or even downtimes. For that purpose, modern structural health monitoring systems can be applied. Integrating a variety of advanced engineering technologies, these systems are capable of collecting continuously structural as well as environmental data through automated data acquisition subsystems deploying sensor networks installed in the wind turbine. Simultaneously, the acquired data streams can be easily transferred to a central computer system for assessing the structural condition in real-time.

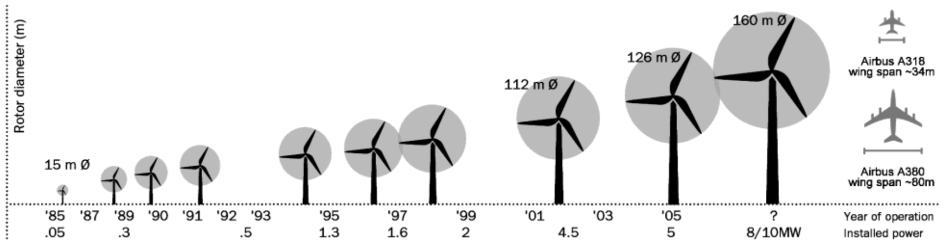


Figure 1. Size evolution and installed power of wind turbines (source: EWEA 2009b, modified).

In the last years, various innovative approaches, stemming from different scientific disciplines, have been explored for creating low-cost, automated and highly reliable monitoring systems. By way of example, Lynch (2007) has introduced a prototypical real-time wireless monitoring system based on embedded computing. Furthermore, new sensing technologies and enhanced measurement systems for damage identification have been proposed and turned into practice in the recent past (Smith & Kripakaran, 2009; Zimmerman, 2008). In particular, multi-agent technology represents a powerful basis for developing autonomous structural health monitoring systems that operate in a fully decentralized fashion. As demonstrated by Smarsly and Hartmann (2004), robust and cost-efficient monitoring systems can be created and practically be applied for taking software agents as a basis. Thereby, the total monitoring problem is decomposed into well-defined sub-problems such as data acquisition, data storage, data analysis, etc. Each sub-problem is proactively solved by a single software agent representing a specialized "problem solving entity". In total, the overall monitoring task is autonomously handled through cooperation and collaboration of the distributed, artificial agent society.

In multi-agent systems research it is well known that agent-oriented software engineering is a challenging task with respect to uncertain environmental dynamics, heterogeneous agent behaviors and spatio-temporally distributed agent interactions that need to be synchronized reliably (Bordini, et al., 2007; Wooldridge & Jennings, 1998). Accordingly, various methodologies have been proposed for supporting a professional and efficient development of multi-agent systems to be applied in diverse areas of applications. However, in the field of structural health monitoring, adequate design strategies are still in their infancy concerning the development of monitoring systems being (i) agent-based, (ii) safety-relevant and (iii) hybrid (i.e. composed of interconnected hardware and software subsystems). For that reason, this paper presents a novel approach towards holistically designing structural health monitoring systems applied to the field of wind turbine monitoring. To this end, a new agent-oriented systems engineering methodology is used, which has been developed at the Institute for Computational Engineering for supporting the efficient design and implementation of robust as well as reliable agent-based monitoring systems. Focusing on the software part of the hybrid wind turbine monitoring system in more detail, the development process is elucidated by presenting exemplarily specific system models created within the global development process. Thereupon, the evaluation of the implemented prototype is demonstrated using actual field data obtained from a wind turbine in operation as well as generated test data simulated in the laboratory.

2 System development using an agent-oriented monitoring systems engineering methodology

The wind turbine monitoring system is designed based on the Agent-Oriented Monitoring Systems Engineering (AGEME) methodology. Developed at the authors' institute, AGEME is intended to support the creation of agent-based, hybrid monitoring systems comprising various collaborating software agents as well as intelligent hardware agents that are realized as sensor nodes to be distributed in the observed structure. In order to ensure reusability, modularity and platform-independence, well-established design methodologies and existing specifications are synergistically integrated, such as Gaia (Wooldridge, et al., 2000), ASEME (Moraitis & Spanoudakis 2004, 2006, 2007), SPEM (OMG 2008), Prometheus (Padgham & Winikoff, 2002a, 2002b; Padgham, et al., 2005), Tropos (Giorgini et al., 2005), etc. AGEME comprises five interlinked development phases, taking into account the specific requirements of structural health monitoring:

i. requirements analysis
ii. agent-based system analysis
iii. agent-based system design
iv. object-oriented implementation
v. system evaluation and optimization

Within each development phase, increasingly detailed system models are created. Applying explicit transformation rules and monitoring knowledge to the models of a development phase *n*, the models of the phase *n*+1 are iteratively generated.

2.1 Requirements analysis

In the first development phase, the requirements and the expected functionalities of the wind turbine monitoring system (WTMS) are defined by a set of interconnected system models. These are

a *scenario model* (SM) that defines distinctive monitoring scenarios to be covered by the WTMS,
an *actor category model* (ACM), deduced from the SM, that defines abstract categories of participating artificial as well as human actors, and
a *computer independent model* (CIM) that defines the system requirements by means of actor diagrams containing the identified actors and their individual goals (see also Spanoudakis & Moraitis, 2007; Giorgini, et al., 2005).

Fig. 2 shows an extract of the created CIM, exemplarily illustrating a simple actor diagram for "*obtaining measured values*" that are stored in database systems: thus, a (human) actor "*user*" requests measurements in order to compute structural analyses. Through the artificial (i.e. computational) actor "*personal assistant*", the request is forwarded to a "*database wrapper*" actor, which is responsible for providing access to the collected measurements. The "*database wrapper*" finally handles the request and extracts the required data sets out of the database system.

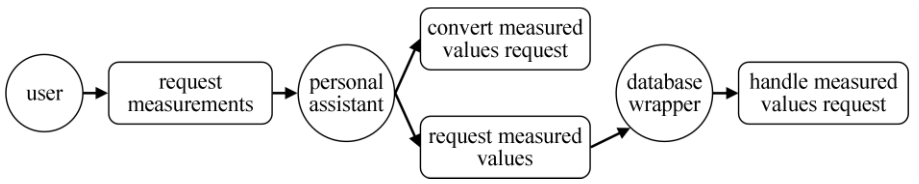


Figure 2. Extract of the CIM model: actor diagram for obtaining measured values.

2.2 Agent-based system analysis

Taking the SM, the ACM and the CIM model as a basis, the generated actor diagrams are coherently transformed into use case diagrams defining the agent capabilities. Based upon an iterative and monitoring-specific transformation process, the actors are transformed to roles and the goals of the actors are transformed to use cases. Corresponding to the actor diagram introduced in Fig. 2, the achieved use case diagram is illustrated in Fig. 3. According to the use case diagram, the artificial "*supervisor assistant*" is identified being a main role of the system, responsible for supporting its human counterpart “*supervisor*” proactively. Furthermore, the “*database wrapper*” and the dependencies between the roles necessary to obtain measured values, are defined and formally specified in the use case diagram.

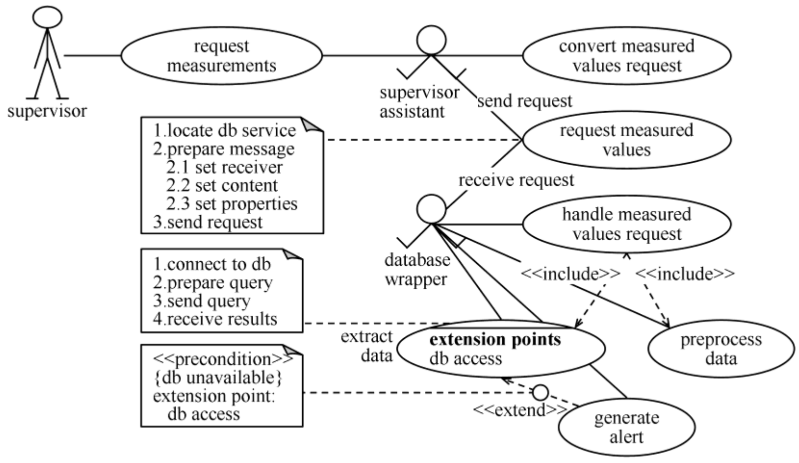


Figure 3. Use case diagram for obtaining measured values.

Using the previously developed system models, two further models, known from conventional agent-oriented development strategies, are generated on another abstraction level: the *roles model* (RM) and the *interaction model* (IM). Both RM and IM are inspired by the Gaia methodology for agent-oriented analysis and design (see Wooldridge, et al., 2000). The RM identifies the expected functions of the roles as well as their permissions and responsibilities in terms of roles schemata. The IM captures the interaction patterns between the roles, represented as a set of interaction protocols. To give an example, Fig. 4 shows the interaction protocols “*request measured values*” and “*generate alert*” that are associated with the “*database wrapper*” role. In these interaction protocols, all participating roles, purposes, parameters and interaction processes are defined. At this, each interaction process is composed of a set of activities and protocols. Activities, written in underlined font, are to be performed by an agent without involving interactions with other agents. By contrast, protocols, written in regular font, are activities that necessarily require interaction between two or more agents.

protocol	request measured values	generate alert
initiator(s)	supervisor assistant	database wrapper
receiver(s)	database wrapper	supervisor assistant
purpose, parameters	specific measurements from a database system are requested *input:* specification of request *output:* extracted data	alerts are generated in case of a database exception (e.g. database not available) *input:* database message *output:* generated alert

process	requestMeasuredValues = locateDBService. prepareMVRMessage.sendMVRequest. receiveMVResults
	prepareMVRMessage = setMVRReceiver. setMVRContent.setMVRProperties
		generateAlert = prepareDBAMessage. sendDBAMessage
		prepareDBAMessage = setDBAReceiver. setDBAContent.setDBAProperties

Figure 4. Interaction protocols associated with the “*database wrapper*” role.

2.3 *Agent-based system design*

The design phase aims at transforming the preceding models into a sufficiently low level of abstraction that is required to appropriately implement the agent-based WTMS using traditional object-oriented techniques. Thus, inter-agent control and intra-agent control are main aspects considered in the design phase. As an output of the design phase, reusable and encapsulated *platform independent models* (PIM) are created.

Accordingly, Fig. 5 illustrates an extract of the generated „*requestMeasuredValues*“ protocol in terms of a state machine. In Fig. 5, the previously defined activities (which do not require agent interaction) are represented as monitoring states in rounded rectangles written in regular font, e.g. “*handleMeasuredValuesRequest*”. In addition, those monitoring states that originate from protocols (which do require agent interaction) are represented in rounded boxes written in italics, such as “*sendResults*”. Inter-agent message exchange is specified using directed graphs (arrows). Attached to the arrows, information concerning performatives, senders and receivers of messages is provided in compliance with the specifications of the Foundation for Intelligent Physical Agents (FIPA 2002).

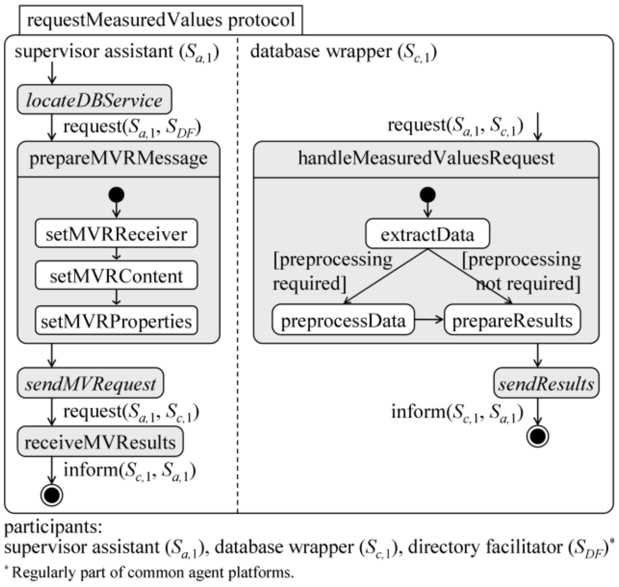
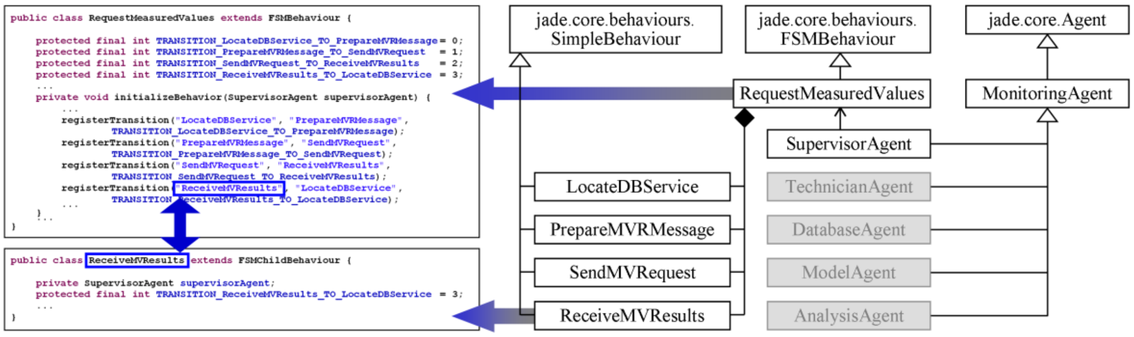

requestMeasuredValues protocol
supervisor assistant ($S_{a,1}$)
database wrapper ($S_{c,1}$)
locateDBService
request($S_{a,1}$, S_{DF})
prepareMVRMessage
setMVRReceiver
setMVRContent
setMVRProperties
sendMVRequest
request($S_{a,1}$, $S_{c,1}$)
receiveMVResults
inform($S_{c,1}$, $S_{a,1}$)
request($S_{a,1}$, $S_{c,1}$)
handleMeasuredValuesRequest
extractData
[preprocessing required]
[preprocessing not required]
preprocessData
prepareResults
sendResults
inform($S_{c,1}$, $S_{a,1}$)
participants:
supervisor assistant ($S_{a,1}$), database wrapper ($S_{c,1}$), directory facilitator (S_{DF})*
* Regularly part of common agent platforms.

Figure 5. Extract of the inter-agent control model: the „*requestMeasuredValues*“ protocol.

2.4 Object-oriented implementation

In the implementation phase, platform-specific system models are defined necessary for implementing the multi-agent system. In this respect, an *agent model* AM is designed, which finally leads to an explicit *class model* CM that strongly depends on the agent platform used. Here, the agent platform JADE is utilized, a Java-based software framework for implementing multi-agent systems through a middleware compliant with the FIPA specifications (Telecom Italia, 2009). Representing a crucial part of the implementation phase, Fig. 6 shows an UML class diagram illustrating an extract of the CM. As can be seen from Fig. 6, several monitoring-specific agent behaviors are attached to the agents. Thereby, the previously identified monitoring states serve as a basis for defining the heterogeneous cluster of agent behaviors. To handle the complexity of the dynamic system appropriately, a one-to-one correspondence between monitoring states and agent behaviors is realized. As a result, one agent behavior is introduced for handling one specific monitoring state (embodied in the classes “*LocateDBService*”, “*PrepareMVRMessage*”, “*SendMVRequest*”, “*Receive MVResults*”). Applying the concept of a finite state machine, each monitoring state – i.e. each agent behavior – is permanently controlled by the supervisor assistant agent through a superior composite behavior that runs autonomously and encapsulates the logic (class “*RequestMeasuredValues*”, see Fig. 6).


```
public class RequestMeasuredValues extends FSMBehaviour {

    protected final int TRANSITION_LocateDBService_TO_PrepareMVRMessage= 0;
    protected final int TRANSITION_PrepareMVRMessage_TO_SendMVRequest   = 1;
    protected final int TRANSITION_SendMVRequest_TO_ReceiveMVResults    = 2;
    protected final int TRANSITION_ReceiveMVResults_TO_LocateDBService  = 3;
    ...
    private void initializeBehavior(SupervisorAgent supervisorAgent) {
        ...
        registerTransition("LocateDBService", "PrepareMVRMessage",
                TRANSITION_LocateDBService_TO_PrepareMVRMessage);
        registerTransition("PrepareMVRMessage", "SendMVRequest",
                TRANSITION_PrepareMVRMessage_TO_SendMVRequest);
        registerTransition("SendMVRequest", "ReceiveMVResults",
                TRANSITION_SendMVRequest_TO_ReceiveMVResults);
        registerTransition("ReceiveMVResults", "LocateDBService",
                TRANSITION_ReceiveMVResults_TO_LocateDBService);
        ...
    }
    ...
}
```

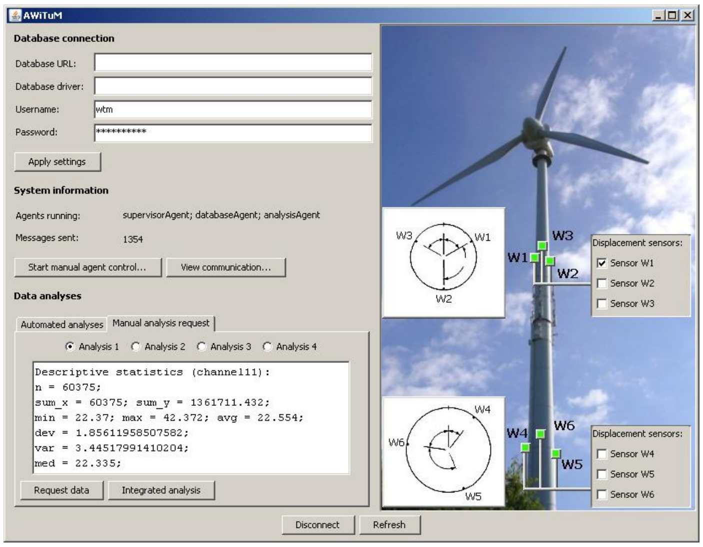
```
public class ReceiveMVResults extends FSMChildBehaviour {

    private SupervisorAgent supervisorAgent;
    protected final int TRANSITION_ReceiveMVResults_TO_LocateDBService = 3;
    ...
}
```

Figure 6. Implementation example: extract of the UML class diagram and corresponding Java-based implementation.

3 Application and evaluation of the agent-based approach

Subsequent to the implementation phase, the WTMS prototype is achieved. Fig.7 shows a preliminary graphical user interface (GUI) of the WTMS as provided by the supervisor assistant agent. The GUI allows manual user interactions with the WTMS, e.g. to obtain specific structural data or to request agent-based data analyses. Different analysis algorithms, to be executed by the WTMS in real-time, are applied serving as test procedures for system evaluation. To this purpose, statistical approaches for detecting outliers within the collected data sets are prototypically implemented to discover anomalies in time.


AWiTuM

Database connection

Database URL:

Database driver:

Username: wtm

Password: **********

System information

Agents running: supervisorAgent; databaseAgent; analysisAgent

Messages sent: 1354

Data analyses

Automated analyses | Manual analysis request

[✓ Analysis 1] Analysis 2 Analysis 3 Analysis 4

```
Descriptive statistics (channel11):
n = 60375;
sum_x = 60375; sum_y = 1361711.432;
min = 22.37; max = 42.372; avg = 22.554;
dev = 1.85611958507582;
var = 3.44517991410204;
med = 22.335;
```

Displacement sensors: [✓ Sensor W1] Sensor W2 Sensor W3

Displacement sensors: Sensor W4 Sensor W5 Sensor W6

Figure 7. GUI generated by the supervisor assistant agent.

Assuming normally distributed data, an outlier is defined as a measured value x_m that exceeds the interval $[\mu-\lambda\cdot\sigma, \mu+\lambda\cdot\sigma]$ where σ is the standard deviation and μ the mean value of the data set considered. The quantity λ is the sensitivity specified by the human experts in charge with monitoring. Thereby, λ=3 is predefined by the supervisor assistant agent. As a second test procedure for detecting outliers within the data sets, IQR (inter-quartile-range) computation is implemented and deployed by the supervisor assistant agent. Similar to a box plot, the first quartile q_1, the third quartile q_3, and the inter-quartile range *IQR*, representing the dispersion of the data set, is computed. Thereupon, a measured value x_m is considered an outlier if located outside the interval $[q_1-\lambda\cdot IQR, q_3+\lambda\cdot IQR]$ using λ=1.5 as a predefined standard value. Further outlier test procedures are currently being realized (e.g.

Hampel's test, Nalimov's test, Dean-Dixon-Test, etc., see Sachs, 2006). In addition, decentralized agent-based analysis strategies using distributed-cooperative knowledge processing are being integrated into the agent-based WTMS. For system evaluation, comprehensive data sets are used that are obtained from a 500 kW wind turbine located in Dortmund, Germany.

Fig. 8 shows on the left-hand side a plot of the temperature measurements T_1 and T_4, acquired inside the shaft of the wind turbine at the levels of 42.7 m (T_1) and 21.7 m (T_4). Evidently, within this actual field data no outliers could be detected that might indicate anomalies, inconsistencies or malfunctions of the sensing devices. Thus, additional test data containing simulated outliers is generated in the laboratory based on the collected field data. As illustrated in the right plot of Fig. 8, the outliers are generated by means of zero values that represent simulated malfunctions of the sensing devices.

As a result, all outliers representing simulated sensing malfunctions are detected reliably through the implemented outlier test procedures by the agent. Also, performance tests are conducted analyzing the time elapsed for executing the agent-based test procedures. In average, approximately 70,000 measurements per second could be analyzed by the agent, which is equivalent to $1.4 \cdot 10^{-5}$ seconds per analyzed measured value[1]. Within this performance study, the time consumed for essential agent-based interaction tasks, necessary to autonomously accomplish the procedures in a fully decentralized fashion, is already incorporated. These tasks conducted by the participating agents include, e.g., the creation of messages m_i as well as self-contained setting of performatives, conversation IDs, senders, receivers, etc.

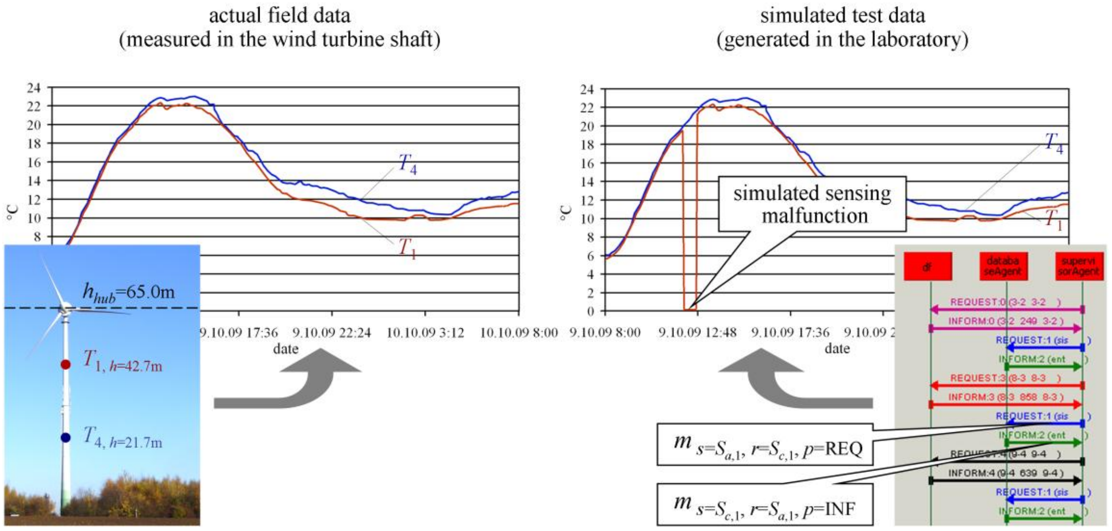


Figure 8. Actual temperature data measured in the wind turbine and simulated test data for system evaluation.

4 Conclusions

In this paper, a novel methodology for developing agent-based structural health monitoring systems has been introduced and applied to the emerging field of wind turbine monitoring in real world scenarios. To this end, the individual development phases of the proposed methodology have been elucidated by means of selected system models. Particularly, the phases of requirements analysis, agent-based analysis and design, object-oriented implementation and system evaluation have been explained.

Representing a substantial enhancement compared to existing agent-based development approaches, substantial conceptual issues could be integrated into the development methodology,

[1] Performed on a 2.16 GHz Apple MacBook, 992 MB RAM, Java version 6, update 16.

which results in a more efficient systems development process compared with conventional development strategies. As an example, the concept of monitoring scenarios has been introduced within the requirements analysis phase. Furthermore, with respect to an appropriate multi-layered agent interaction design, an explicit differentiation between protocols and activities has been proposed at an early development stage (i.e. in the analysis phase). Also, reusable and platform-independent system models could be created on different abstraction levels in order to avoid duplicate development efforts in designing monitoring systems.

Although the preliminary results, as explored in the experimental study, clearly indicate the enormous potential of the proposed approach, it should be emphasized that additional intensive field tests are still necessary for further improvements of the performance and reliability of the implemented prototype. For that purpose, the area of bridge monitoring is to be considered soon. At this, further intelligent agent behaviors as well as monitoring-specific interaction patterns but also enhanced distributed as well as proactive knowledge processing are to be incorporated into the monitoring system.

Acknowledgements

The authors gratefully acknowledge the financial support of the German Research Foundation (DFG) through the research grants SM 281/1-1 (principal investigator Kay Smarsly) and HA 1463/20-1 (principal investigator Dietrich Hartmann).

References

BORDINI, R. H., DASTANI, M. & WINIKOFF, M., 2007. Current Issues in Multi-Agent Systems Development. In J. G. Carbonell & J. Siekmann, eds. Engineering Societies in the Agents World VII. Berlin: Springer, 2007, pp.38-61.

EWEA, (The European Wind Energy Association), 2009a. *Wind Energy and Research*. [Online] Available at: http://www.ewea.org.

EWEA, (The European Wind Energy Association), 2009b. *Wind Power Technology*. [Online] Available at: http://www.ewea.org.

GIORGINI, P. et al., 2005. The Tropos Metamodel and its Use. Informatica, 29(4), pp.401-408.

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS (FIPA), 2002. SC00037J *FIPA Communicative Act Library Specification*. Geneva, Switzerland: Foundation for Intelligent Physical Agents.

LYNCH, J. P., et al., 2007. Decentralized Civil Structural Control using Real-time Wireless Sensing and Embedded Computing. *Smart Structures and Systems*, Techno Press, 3(3), pp.321-340.

MORAITIS, P. & SPANOUDAKIS, N., 2004. Combining Gaia and JADE for Multi-Agent Systems Development. In EMCSR 2004, (Austrian Society for Cybernetic Studies) The 17th European Meeting on Cybernetics and Systems Research. University of Vienna, Austria, April 6 - 9, 2010. Austrian Society for Cybernetic Studies: Vienna, Austria.

PADGHAM, L. & WINIKOFF, M., 2002a. Prometheus: A Methodology for Developing Intelligent Agents. In F. Giunchiglia, J. Odell & G. Weiß, eds. Agent-Oriented Software Engineering III. Berlin: Springer, 2002, pp.174-185.

PADGHAM, L. & WINIKOFF, M., 2002b. Prometheus: A Pragmatic Methodology for Engineering Intelligent Agents. In *The 17th Conf. on Object-Oriented Progr., Systems, Languages, and Appl.* Seattle, WA, USA, 2002. ACM: New York, NY, USA.

PADGHAM, L., THANGARAJAH, J., WINIKOFF, M., 2005. Tool support for agent development using the Prometheus methodology. In *The 5th International Conference on Quality Software*. Melbourne, Australia, 2005.

SACHS, L. & HEDDERICH, J., 2006. *Angewandte Statistik*. 12th ed. Berlin, Germany: Springer.

SMARSLY, K. & HARTMANN, D., 2004. Autonome Überwachung sicherheitsrelevanter Ingenieurbauwerke. In J. Zimmermann, ed. *Forum Bauinformatik 2004*. Braunschweig, Germany, 2004, Shaker Publ.: Aachen, Germany.

SPANOUDAKIS, N. & MORAITIS, P., 2006. The Gaia2Jade Process for MAS Development. *Applied AI*, 20, pp.251–273.

SPANOUDAKIS, N. & MORAITIS, P., 2007. The Agent Systems Methodology (ASEME): A Preliminary Report. In EUMAS 07, 5th European Workshop on Multi-Agent Systems. Hammamet, Tunisia, December 13-14, 2007.

SMITH, I. & KRIPAKARAN, P., 2009. Configuring and enhancing meas. systems for damage identification. *AEI*, 23, pp.424–432.

TELECOM ITALIA S.P.A. 2009. *JADE - Java Agent DEvelopment Framework, Version 3.7*. [Online] Telecom Italia S.p.A. Available at: http://jade.tilab.com [Accessed October 20, 2009].

THE OBJECT MANAGEMENT GROUP (OMG), 2008. Software & Systems Process Engineering Meta-Model Specification, Version 2. OMG Document Number: formal/2008-04-01. Available at: http://www.omg.org/spec/ SPEM/2.0/PDF.

WOOLDRIDGE, M. & JENNINGS, N. R., 1998. Pitfalls of Agent-Oriented Development. In K. Sycara-Cyranski. *The Second Int. Conf. on Autonomous Agents.* Minneapolis, MI, USA, 1998, Assoc. for Computing Machinery (ACM): New York, NY, USA.

WOOLDRIDGE, M. et al., 2000. The Gaia Methodology for Agent-Oriented An. and Des. Aut. Agents and MAS, 3(3), pp.285-312.

ZIMMERMANN, A. T., et al., 2008. Automated Modal Parameter Estimation by Parallel Processing within Wireless Monitoring Systems. *Journal of Infrastructure Systems*, ASCE, 14(1), pp.102-113.