18th International Conference on the Application of Computer
Science and Mathematics in Architecture and Civil Engineering
K. Gürlebeck and C. Könke (eds.)
Weimar, Germany, 07–09 July 2009

REAL-TIME MONITORING OF WIND ENERGY CONVERTERS BASED ON SOFTWARE AGENTS

K. Smarsly * and D. Hartmann

* *Institute for Computational Engineering (ICE), Ruhr-University Bochum*
Universitätsstr. 150, D-44780 Bochum, Germany
E-mail: *kay.smarsly@rub.de*

Keywords: Wind energy converters, structural health monitoring, safety assessment, damage detection, sensing networks, agent technology, artificial intelligence.

Abstract. *Due to increasing numbers of wind energy converters, the accurate assessment of the lifespan of their structural parts and the entire converter system is becoming more and more paramount. Lifespan-oriented design, inspections and remedial maintenance are challenging because of their complex dynamic behavior. Wind energy converters are subjected to stochastic turbulent wind loading causing corresponding stochastic structural response and vibrations associated with an extreme number of stress cycles (up to 10^9 according to the rotation of the blades). Currently, wind energy converters are constructed for a service life of about 20 years. However, this estimation is more or less made by rule of thumb and not backed by profound scientific analyses or accurate simulations. By contrast, modern structural health monitoring systems allow an improved identification of deteriorations and, thereupon, to drastically advance the lifespan assessment of wind energy converters. In particular, monitoring systems based on artificial intelligence techniques represent a promising approach towards cost-efficient and reliable real-time monitoring. Therefore, an innovative real-time structural health monitoring concept based on software agents is introduced in this contribution. For a short time, this concept is also turned into a real-world monitoring system developed in a DFG joint research project in the authors' institute at the Ruhr-University Bochum. In this paper, primarily the agent-based development, implementation and application of the monitoring system is addressed, focusing on the real-time monitoring tasks in the deserved detail.*

1 INTRODUCTION

The growing demand for renewable energy sources urges to foster "clean" power production. To this respect, wind energy is a strong competitor compared to traditional alternatives because of its relative cost-effectiveness [1]. Fig. 1 shows a prognosis of power production costs comparing wind-generated electricity with conventionally generated electricity in Germany [2, 3]. By now, wind energy converters generate electricity at a price of less than 0.08 €/kWh, whereas the costs of conventional electricity generation are continuously rising. Similar situations apply in other industrial countries than Germany.

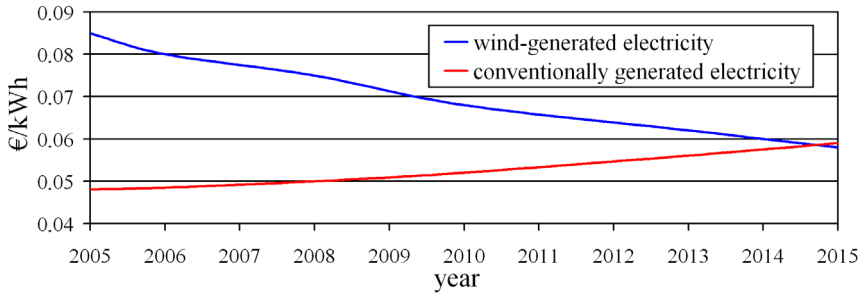

Figure 1: Prognosis of electricity production costs in Germany (source [2, 3]).

As another remarkable trend, current state-of-the-art wind energy converters consistently grow in size. Consequently, maintenance activities as well as revitalization work become more challenging. Hence, expenditures for safety assessment and condition monitoring are increasing, which underpins the urgent need for cost-effective and reliable monitoring systems.

By applying automated monitoring systems, unnecessary replacements of components as well as down time can be avoided, leading to decreased lifecycle costs, reduced inspection time and increased performance. In addition, the frequency of breakdowns can significantly be lowered. Recently, the necessity of applying innovative monitoring systems has been recognized in various engineering disciplines. Moreover, in practice and in politics monitoring of wind energy converters is regarded as a promising field worth to be supported by considerable financial investments [4, 5]. Accordingly, in numerous countries large wind energy farms are being erected with federal assistance. Also in science, considerable efforts have been being undertaken in the past years. Damage detection algorithms applied to wind energy converters are being improved, in parallel, novel sensing devices as well as sophisticated monitoring techniques are established. In addition, the application of simulation-based computational models is fostered for achieving advanced safety assessment strategies [6-14].

However, most approaches towards monitoring still cause tremendous financial expenses. Furthermore, cost-intensive application of man power and equipment is required, whereby human experts charged with monitoring need to precisely analyze the structural condition in periodic intervals. Thus, a considerable improvement of current monitoring approaches is the application of "intelligent" monitoring concepts which are able to support the human actors in a pro-active fashion. For that purpose, a novel multi-scale monitoring concept based on software agents is introduced in this paper. Software agents – representing self-contained computational entities – are capable of autonomously processing monitoring tasks in order to support the human actors to a large extend.

In this contribution, the development and the application of a wind energy converter monitoring system is addressed, which is one part of an ongoing research project funded by the German Research Foundation (DFG [15]). Considering the "real-time" monitoring aspects of this system in more detail, the actual system development is elucidated by scrutinizing the "data conversion" in monitoring. To this end, two alternative strategies are prototypically implemented for solving the data conversion problem appropriately. With the aid of a study, both strategies are compared with respect to their performance and applicability. Finally, the results are discussed and a concise review is given on future potential research.

2 REAL-TIME MONITORING CONCEPT BASED ON SOFTWARE AGENTS

To permanently observe structural conditions and to provide comprehensive data for an accurate assessment of the lifespan, a monitoring concept based on software agents is introduced. Operating on different monitoring layers, two spatially distributed subsystems are created, forming the total monitoring system (s. fig. 2). The first subsystem is the Multi-Sensor System *MSS* to be installed in the wind energy converter for continuously collecting structural and environmental data. Hereby, the *MSS* is responsible for automated signal processing and steering of the installed sensing network (i.e. accelerometers, displacement and temperature sensors, 3d ultrasonic anemometers, etc.).

A distributed Multi-Agent System *MAS*, representing the second subsystem, controls the multi-sensor system *MSS*. Furthermore, the *MAS* provides an autonomous accomplishment of monitoring tasks, such as real-time data analyses, data conversion or data storage, and embodies several cooperating software agents. Deduced from distributed artificial intelligence, a software agent can be seen as a piece of software that acts for a user or program (e.g. another software agent). Each software agent applied is designed for explicitly solving one monitoring task. Compared to other modern monitoring concepts, software agents allow an efficient decomposition of the total monitoring problem into individual subproblems, which can adequately be solved. Consequently, a scalable upgrading, an extension or a modification of the total system becomes possible because of the autonomous nature of the agents.

The software agents applied within the *MAS* are divided into two agent categories:

(i) *Online agents* are responsible for autonomously performing real-time monitoring tasks that must be executed permanently. Examples are data acquisition and pre-processing of the collected data.

(ii) *Offline agents* are utilized by the human experts on demand. Offline agents are, e.g., responsible for accomplishing detailed system identification by applying a finite element model along with the collected measurements.

Both subsystems are intended to work independently: The *MSS* and the *MAS* are linked via a network connection. As a result, exchangeability of each subsystem is achieved. By introducing independent subsystems, the overall system architecture is in harmony with current monitoring standards for wind energy converters which recommend a client/server-like pattern as global system architecture (IEC 61400-25, [16-18]).

In the following sections, the development of the *MAS* is considered, particularly focusing on those design issues that are related to the online monitoring aspects to be processed in real-time.

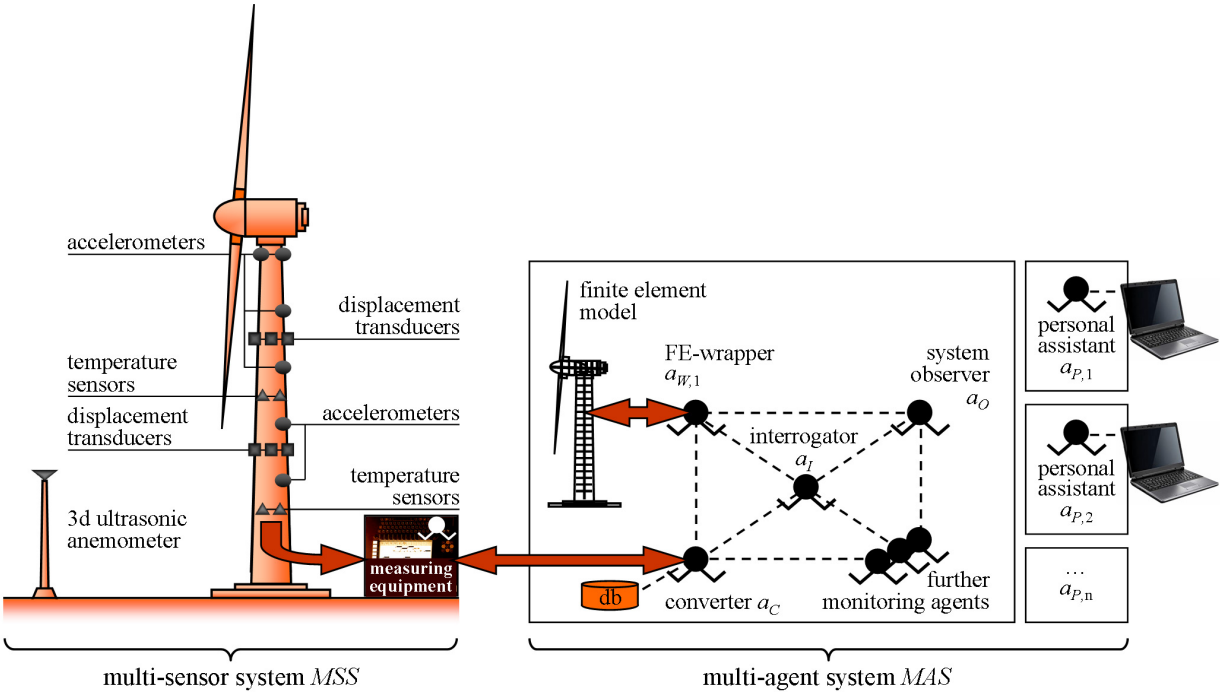

Figure 2: Structural health monitoring concept – multi-level approach.

2.1 Agent-based system analysis and design

Requirements to be met by the *MAS* are the following:

- to reliably collect relevant monitoring data that is also designated as "primary data"
- to autonomously pre-process the primary data (i.e. to backup and to check the data with regard to inconsistencies) and to create alerts in the case of detected anomalies
- to permanently convert the primary data into "secondary data" which is then interpreted and utilized for further processing by both human experts and software agents, respectively
- to persistently store the secondary data in a central monitoring database

Furthermore, the *MAS* is designed such that a fault-tolerant system behavior can be established. For the purpose of consistency, a novel design methodology is used for creating agent-based monitoring systems (see also [19]): As a key part of the methodology, a set of roles is defined. Hereby, a role represents an abstract description of an organizational unit defining a specific system function to be implemented.

All roles within the *MAS* are based on the above listed requirements. Consequently, the set of roles e.g. includes a "backup role" r_B, an "interrogator role" r_I, an "alert transmitter role" r_A, a "finite element-wrapper" role $r_{W,1}$, roles for supporting the human actors $r_{P,i}$, an "observer role" r_O and a "converter role" r_C. The converter role r_C is considered as crucial within the *MAS* because reliable and efficient data conversion from primary into secondary data has to be accomplished. The converter role r_C is defined according to the schema depicted in fig. 3 which describes expected functions of the role r_C using both permissions and responsibilities of the role [20]. Responsibilities are divided further into two subcategories: Safety and liveness responsibilities that are defined by specific activities and protocols associated with a role. Besides a multitude of further monitoring specific protocols defined for the *MAS*, the protocol

doConversion is responsible for the intelligent, self-contained conversion and the processing of the collected primary data.

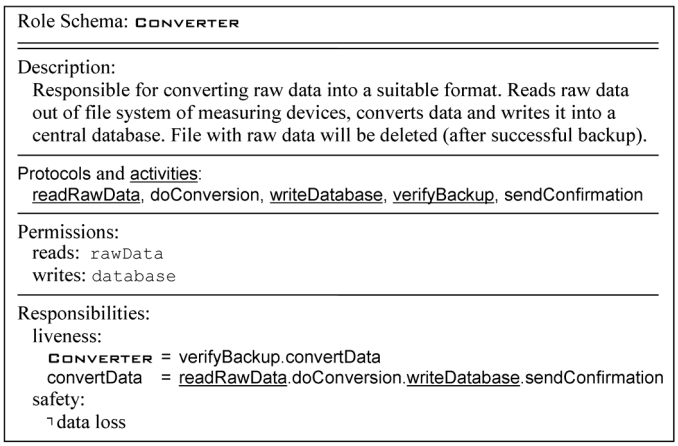

Figure 3: Schema for converter role r_C.

The protocol definition of the doConversion protocol is illustrated in fig. 4. As can be seen from fig. 4, the protocol is subdivided into a request and a response part (ConvertingRequest and ConvertingResponse). This subdivision ensures an appropriate design and implementation of the heterogeneous agent system in the succeeding steps.

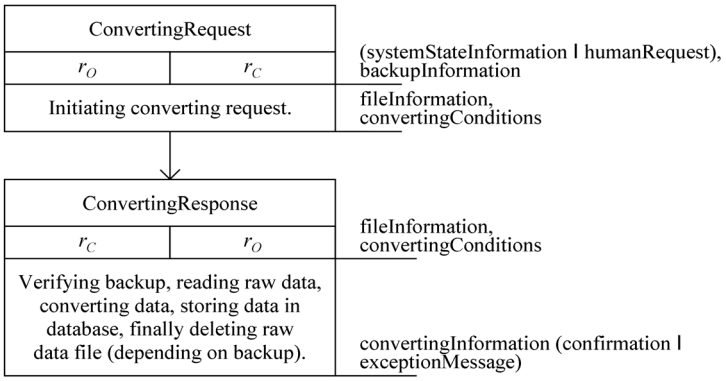

Figure 4: Protocol definitions related to data conversion.

Based on the defined roles and interaction protocols, an agent model for the *MAS* is created. Hereby, the roles are aggregated in so called *agent types* that finally make up the *MAS*. In agent research, a one-to-one correspondence between roles and agent types is frequently achieved. However, due to the high degree of interdependence between the converter role r_C and the backup role r_B, both roles are grouped together in one single agent type (r_O and r_A as well). Fig. 5 shows the roles aggregated into agent types forming the *MAS*.

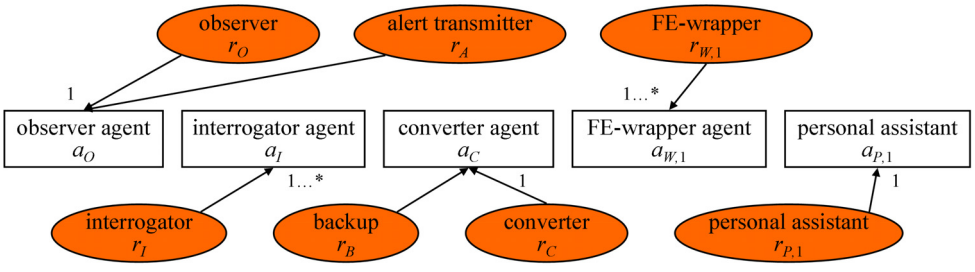

Figure 5: Agent model of the multi-agent system.

2.2 Behavior-based modeling and object-oriented implementation

Having defined various monitoring services incorporated into the agents (such as "conversion service") and having identified the communication paths between the agents, behavior-based modeling is accomplished. Unlike objects in object-oriented programming (OOP), which are defined in terms of attributes, methods and associations, a software agent is further defined by its behavior(s). Following the behavior-based modeling, all behaviors and agents are consistently implemented into several classes.

Taking into account the major functions of the converter agent a_C, its main behaviors are denoted `Converter` and `Backup`. As a matter of course, the `Converter` behavior controls and executes automated data conversion continuously, and the `Backup` behavior is applied for cyclic backups of the collected monitoring data. Key classes related to the converter agent's `Converter` and `Backup` behavior are shown in fig. 6[1].

In order to investigate performance and applicability of different data conversion approaches, two different, but concurrently usable conversion strategies are implemented. Accordingly, the abstract behavior class `Converter`, providing semantics necessary for data conversion, serves as a super class for specific implementations of converter behaviors. On this, the abstract class `Converter` is extended by the specific converter classes `PDIConverter` and `CustomConverter` as shown in fig. 6. Each converter class epitomizes a particular conversion strategy to be applied by the converter agent a_C: On the one hand, the `PDIConverter` integrates ("wraps") commercial software into the *MAS* for data conversion. On the other hand, the `CustomConverter` conducts data conversion by utilizing internal agent capabilities and knowledge. Using this adapter-like modeling approach, additional conversion strategies can be integrated in the same manner by adding further classes later on.

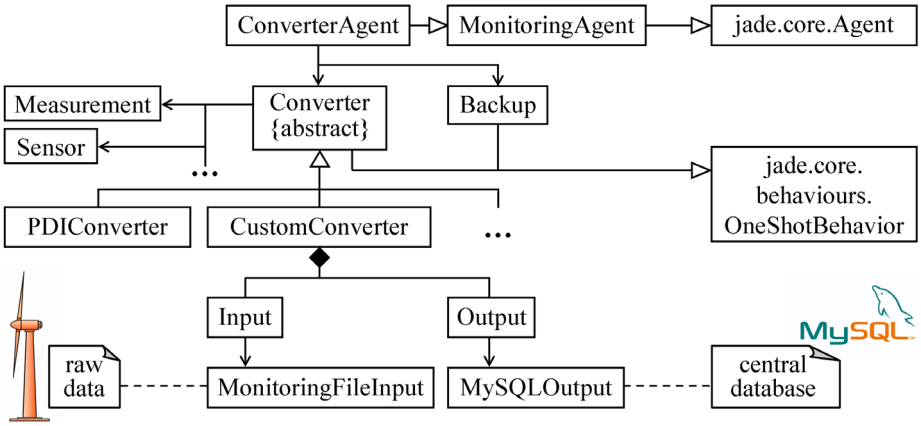

Figure 6: Extract of the behavior-related class structure (converter agent).

3 APPLICATION EXAMPLES AND SYSTEM EVALUATION

The implementation of both the *agent-internal* conversion and the *PDI* conversion are briefly demonstrated in the following examples. Hereon, performance of both implementations is evaluated.

[1] For implementing the agent system, the "Java Agent Development Framework" JADE is applied (http://jade.tilab.com).

3.1 Agent-internal conversion

For the agent-internal conversion of data, the class `CustomConverter` is implemented using a modular concept. As depicted in fig. 6, this concept comprises an input module (class `Input`) and an output module (class `Output`). By that, the converter agent a_C is capable of handling different input data formats ("primary data"). Vice versa, generating of various output data formats ("secondary data") is possible.

The input data format prototypically applied is a predefined vendor-specific format (`MonitoringFileInput`). This data format is determined by the specific measuring devices installed in the wind energy converter. To illustrate the input data format given, fig. 7 shows a representative part of the input file generated by a temperature logger within laboratory tests. For generating output data, the relational database management system MySQL is utilized (`MySQLOutput`).

```
Datum→    →   Zeit   →   Channel·1→ Channel·2→ Channel·3→ Channel·4→ Channel·1→ Channel·2→ Channe
   →      →        →        °C   →     °C   →     °C   →     °C   →     °C   →     °C   →     °C
   →      →        →         →         →         →         →         →         →         →
01.06.2009 → 07:59:28→ 22,378  →  ******  →  ******  →  ******  →  ******  →  22,692  →  22,726
01.06.2009 → 07:59:29→ 22,378  →  ******  →  ******  →  ******  →  ******  →  22,692  →  22,726
01.06.2009 → 07:59:30→ 22,377  →  ******  →  ******  →  ******  →  ******  →  22,692  →  22,725
01.06.2009 → 07:59:31→ 22,377  →  ******  →  ******  →  ******  →  ******  →  22,692  →  22,725
01.06.2009 → 07:59:32→ 22,377  →  ******  →  ******  →  ******  →  ******  →  22,692  →  22,725
01.06.2009 → 07:59:33→ 22,380  →  ******  →  ******  →  ******  →  ******  →  22,687  →  22,727
01.06.2009 → 07:59:34→ 22,380  →  ******  →  ******  →  ******  →  ******  →  22,687  →  22,727
01.06.2009 → 07:59:35→ 22,380  →  ******  →  ******  →  ******  →  ******  →  22,687  →  22,727
01.06.2009 → 07:59:36→ 22,378  →  ******  →  ******  →  ******  →  ******  →  22,688  →  22,729
01.06.2009 → 07:59:37→ 22,378  →  ******  →  ******  →  ******  →  ******  →  22,688  →  22,729
```

Figure 7: Data file containing primary data generated by a 3×4-channel temperature logger.

In order to accomplish data conversion efficiently, the converter agent a_C autonomously identifies all input variables relevant to the conversion. Simultaneously, monitoring meta data is attached by the agent to the primary data given (such as definition of installed sensors, logger IDs, output specification details, time formats). Through this, input variables are allocated to different output tables, rows, etc. Integrating further vectors and fields, the corresponding MySQL tables are created (or updated respectively) containing the generated secondary data[2]. As an example, fig. 8 presents the secondary data autonomously generated by converter agent a_C. Stored in the central database, secondary data can be accessed by the human experts and by the software agents applied in the agent system as well.

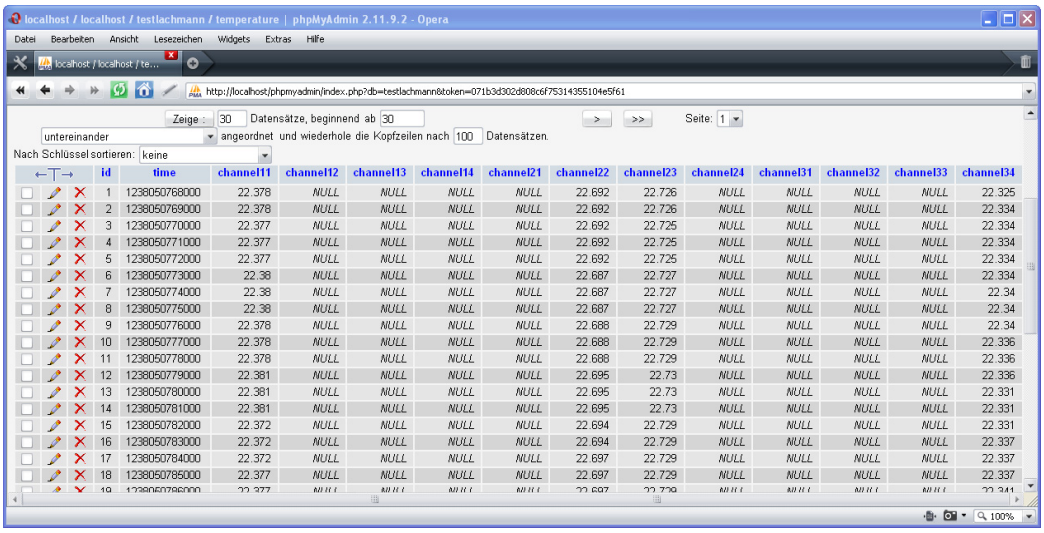

Figure 8: Visualization of secondary data.

[2] For autonomous communication between online agent system and MySQL database system, the JDBC application programming interface, which provides methods for querying and updating data in a database system, is utilized by the converter agent.

3.2 PDI conversion

Besides agent-internal data conversion, an "external" conversion approach is pursued and materialized in another behavior class named `PDIConverter`. This class integrates the open-source ETL[3] tool "PDI" (Pentaho Data Integration) into the *MAS*. As commercial software, the PDI tool offers powerful data extraction, transformation and loading capabilities through a standards-based architecture.

For processing data conversion, the agent a_C defines a chain of conversion subtasks that are transferred to the external PDI tool used for executing the conversion. Fig. 9 illustrates the definition of these conversion subtasks and the corresponding visualization in the PDI tool. In detail, the chain comprises subtasks performing (i) start of the conversion service, (ii) data processing and (iii) shifting of input files for backup.

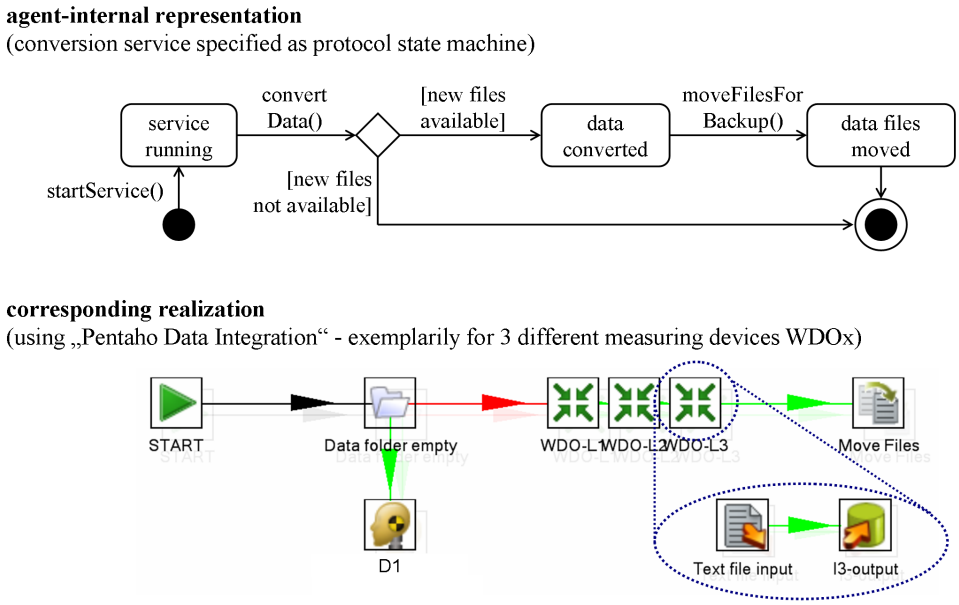

Figure 9: PDI-based conversion approach.

3.3 Comparison and evaluation

Since large data volume is the most critical aspect in handling and processing monitoring data, the time consumed for converting primary data into secondary data is considered as a key criterion for evaluating the performance of agent-internal and PDI conversion.

In total, data sets ranging from 1 to 1,000,000 simulated measurement readings are applied for performance tests. Fig. 10 shows a graphical comparison of the consumed time periods. Evidently, the number of converted measurement data scales linearly with respect to time. This linearity indicates that the quantity of processed data does not affect the performance of the *MAS*. Also, fig. 10 illustrates that the agent-internal conversion $f_{a\text{-}i}(m)$ (blue line) is more performant than the external PDI conversion $f_{PDI}(m)$ (red line). Reasons are the high quality of the agent-internal implementation and, secondly, the fact that the external PDI tool, wrapped into the system, has not specifically been designed for solving this particular conversion problem. Different from the agent-internal implementation, it is rather a commercial tool for processing a wide variety of diverse data conversion and integration problems.

[3] ETL = extract, transform, load.

In summary, both implementations represent powerful and efficient variants for real-time processing of monitoring data: The agent-internal implementation consumes 0.014 ms in average for conversion of a single measurement, the PDI approach takes 0.018 ms (performed on a 2.16 GHz Apple MacBook[4]).

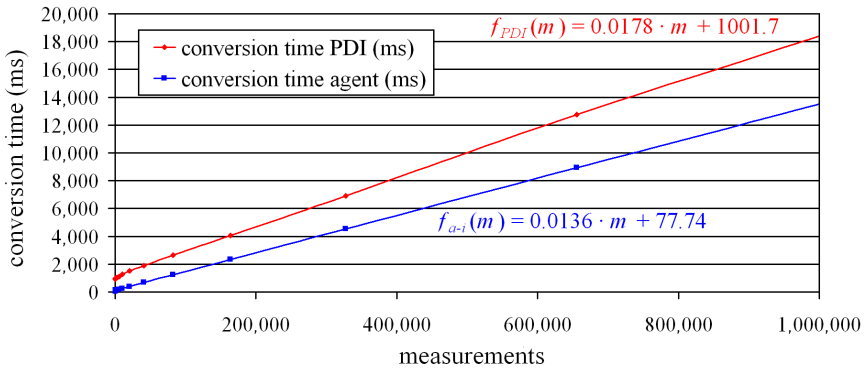

Figure 10: Comparison of agent-internal and PDI approach.

Finally, the agent-internal implementation is integrated into the *MAS*. Fig. 11 shows the corresponding monitoring system GUI prototypically generated by converter agent a_C. Besides *autonomous* processing, data conversion can also be launched *manually* by explicitly selecting a conversion method and by specifying input and output data formats. Thereby, additional conversion information can be defined, such as location of input files or specification of database drivers, URLs or further meta data.

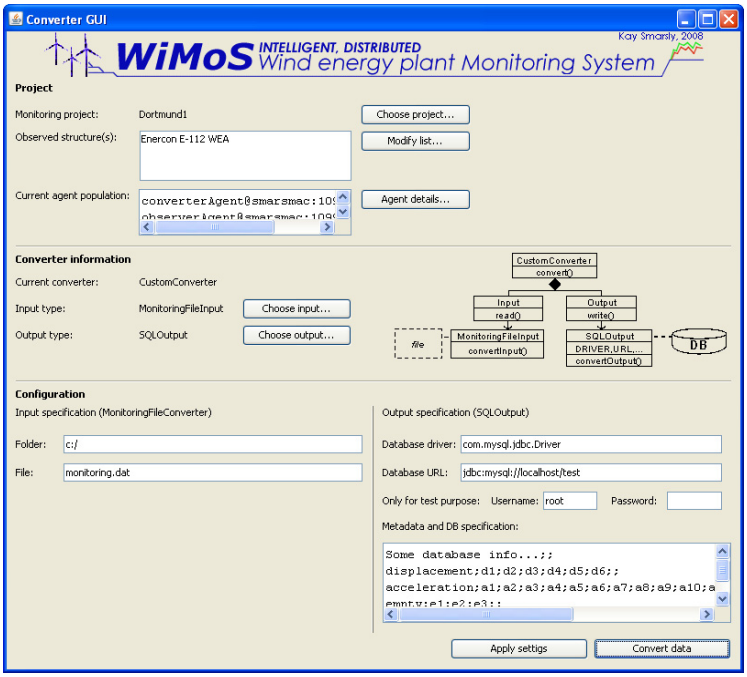

Figure 11: GUI of the online agent system for data conversion.

[4] 2.16 GHz Intel Core 2 Duo (T7400), 992 MB RAM. Java version 6 update 11.

4 CONCLUSIONS AND FUTURE WORK

It could be shown that sophisticated monitoring concepts based on artificial intelligence and autonomous data conversion are capable of improving current structural health monitoring strategies considerably. Agent technology, stemming from artificial intelligence, has successfully been applied to the engineering problem of wind energy converter monitoring. In this context, agent-based analysis and design as well as behavior-based modeling and object-oriented implementation of a monitoring system have been demonstrated. Taking the monitoring task of autonomous data conversion as an example, two concurrent implementations have been elucidated that are utilized for wind energy converter monitoring. In total, both implementation concepts are well suited for the application in advanced monitoring practice in association with the autonomy and the self-contained nature of software agents.

Since the beginning of 2009, this concept is also turned into a real-world monitoring system developed within a DFG joint research project in the authors' institute at the Ruhr-University Bochum. The developed monitoring system is currently being mounted on a 500 kW wind energy converter for intensive field tests. With respect to future work, a promising aspect is to integrate further research issues into the monitoring system, which are currently being investigated in this field. Examples include wireless sensing, sensor and actuator placement optimization strategies, photogrammetric concepts, novel networking approaches for communication between nacelle and rotating turbine as well as novel blade monitoring, simulation and system identification concepts.

REFERENCES

[1] Ciang, C. C., Lee J. R. and H. J. Bang. 2008. "Structural health monitoring for a wind turbine system: a review of damage detection methods," Meas. Sci. Technol. 19 122001.

[2] German Wind Energy Association. 2005. "Perspektiven der Windkraftwirtschaft in NRW," report, Berlin, Germany.

[3] Institute for Solar Energy Supply Technology (ISET). 2005. "Effizienz der Windenergie," report, Kassel, Germany.

[4] German Federal Ministry of Justice. 2009. "EEG – Gesetz für den Vorrang erneuerbarer Energien," (BGBl. I S. 2074), geändert durch Artikel 5 des Gesetzes vom 28. März 2009 (BGBl. I S. 643), Berlin, Germany.

[5] Ministry of Innovation, Science, Research and Technology of the State of North Rhine-Westphalia. 2009. "Das Cluster EnergieForschung.NRW," report, Düsseldorf, Germany.

[6] Farrar, C. R., Worden, K., Todd, M. D., Park, G., Nichols, J, Adams, D. E., Bement, M. T., and K. Farinholt. 2007. "Nonlinear System Identification for Damage Detection," report, LA-14353, Los Alamos National Laboratories, Los Alamos, NM, USA.

[7] James, G. 1996. "Development of Structural Health Monitoring Techniques using Dynamics Testing," SAND96-0810, Sandia National Laboratories, Albuquerque, NM, USA.

[8] Sutherland, H., Beattie, A., Hansche, B., Musial, W., Alread, J., Johnson, J., and M. Summers. 1994. "The Application Of Non-Destructive Techniques to the Testing of a Wind Turbine Blade," SAND93-1380, Sandia National Laboratories, Albuquerque, NM, USA.

[9] Jüngert, A. 2008. “Damage Detection in wind turbine blades using two different acoustic techniques,” The NDT Database & Journal (NDT).

[10] Lading, L., et al. 2002. “Fundamentals for Remote Structural Health Monitoring of Wind Turbine Blades – a Preproject,” Risø National Laboratory, Roskilde, Denmark.

[11] Rumsey, M. A. and J. A. Paquette. 2008. “Structural health monitoring of wind turbine blades,” Smart Sensor Phenomena, Technology, Networks, and Systems 2008. Proceedings of the SPIE, 6933, pp. 69330E-69330E-15.

[12] Wörmann, R. and Reinhard R. Harte. 2003. “Structural degradation of wind-turbine towers under combined wind and thermic action,” in second MIT Conference on Computational Fluid and Solid Mechanics, Cambridge, MA, USA.

[13] Fisher, K. 2004. “Integrated Ice Protection and Self-Monitoring Rotor System,” in proceedings of the Sandia National Laboratory Wind Turbine Blade Workshop 2004, Albuquerque, NM, USA.

[14] Zerbst, S., Reetz, J., Gerasch, W.-J., and R. Rolfes. 2007. “Methoden zur Schadenserkennung an Tragstrukturen von Windenergieanlagen,” in proceedings of the fifth Symposium Offshore Wind Energy, Hannover, Germany.

[15] Hartmann, D. 2009. “Lebensdauerabschätzung von Windenergieanlagen mit fortlaufend durch Systemidentifikation aktualisierten numerischen Modellen,” DFG – Individual Grants Program Ha 1463/20-1, Institute for Computational Engineering, Ruhr-University Bochum, Bochum, Germany.

[16] International standard IEC 61400-25 (all parts). 2006. Communications for monitoring and control of wind power converters, TC 88. 2006-12. International Electrotechnical Commission, Geneva, Switzerland.

[17] Giebel, G., Gehrke, O., McGugan, M., and K. K. Borum. 2006. “Common access to wind turbine data for condition monitoring. The IEC 61400-25 family of standards,” in proceedings of the 27th Risø International Symposium on Materials Science, Risø National Laboratory, Denmark.

[18] Giebhardt, J., et al. 2004. “Predictive Condition Monitoring for Offshore Wind Energy Converters with Respect to the IEC 61400-25 Standard,” in proceedings of the International Technical Wind Energy Conference (DEWEK), Wilhelmshaven, Germany.

[19] Smarsly, K. 2008. “Autonome Überwachung sicherheitsrelevanter Ingenieurbauwerke,” doctoral thesis. Shaker Publishing, ISBN 978-3-8322-7562-4, ISSN 1614-4384, Aachen, Germany.

[20] Wooldridge, M., Jennings, N. R., Kinny, D. 2000. “The Gaia Methodology For Agent-Oriented Analysis And Design.” Journal of Autonomous Agents and Multi-Agent Systems, Vol. 3, No. 3, pp. 285-312.