Ontology-based Fault Diagnosis for Industrial Control Applications

Martin Melik-Merkumians, Alois Zoitl
Automation and Control Institute
Vienna University of Technology
Gußhausstr. 27-29/E376
A-1040 Vienna, Austria
{Melik-Merkumians, Zoitl}@acin.tuwien.ac.at

Thomas Moser
Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstrasse 9-11/188
A-1040 Vienna, Austria
thomas.moser@tuwien.ac.at

Abstract

Traditional fault detection systems in industrial control applications are just able to report occurring faults. Fault diagnosis systems are more desirable for plant operators, as such systems are capable to reduce the number of occurring alarms by elimination of consecutive alarms and prioritization of critical alarms. The disadvantage of those systems is that they have to be implemented anew for every control application, as the system dependencies vary from application to application. Ontology-based fault diagnosis systems do not have this disadvantage. Only the ontology has to be created for the new system, which greatly reduces time and effort for new systems, as old ontologies can be reused for the new system.

1. Introduction

Alarm management, Supervisory Control and Data Acquisition (SCADA) performance data acquisition and visualization, and performance data storage are becoming more and more complicated, as plant and process complexity is increasing. Current solutions for these problems are insufficient. It is not uncommon, that thousand alarms are reported simultaneously by a production facility to the control room personnel. Only highly skilled and experienced personnel is able to identify the most important alarms and root causes in such an alarm flood. SCADA performance data acquisition is configured singular one tag after another and in a similar cumbersome way the human-machine-interface (HMI) is generated. Granted, HMI wizards reduce effort of HMI generation, but the configuration effort tends to be the same. Performance data storage is especially important for food manufacturers and pharma industry, who must be able to provide all important plant data in case of contamination of their products. Right now this data exists on data acquisition and storage server, but the reconstruction of the actual plant processes on basis of this data is nigh to impossible.

With the help of ontologies and reasoning most of those problems can be addressed, especially if the ontologies are implemented in the automation devices themselves and if the performance data storage is able to save the data accordingly to the ontology. Thereby it is possible to store the performance data in a structured and comprehensible way. Performance data request can be issued for specific devices, groups, and parts of devices and groups in a simple way. For example the performance data of a tank can be requested by a single request. With this in mind the data visualization can also be simplified as the ontology of an automation device can be projected on a HMI with little effort.

But the most important aspect of this paper is the possibility to perform reasoning on ontological models as an early-alarm system for critical failures, fault tree analysis, and alarm prioritization and reduction. The ability of ontologies to model correlations and constraints of different system parts and also for specific system states is suited to describe system dependencies. This correlations, constraints, and dependencies can be modeled very easily by several rules that apply to the ontology. A software system, called Reasoner, can be used to check if the model and also the current state of the model is in consistency with the imposed rule set. Occurring inconsistencies, and therefore potential system faults, are reported by the Reasoner.

The goal of this paper is to propose an ontology-based fault diagnosis system, which is able to identify the root cause of faults or at least can reduce the number of possible causes. Furthermore such systems are able to provide additional information for the maintenance personnel, e.g., which system parts have to be checked or replaced.

2. State of the Art

System faults of control application systems can lead to severe damage to the plant and personnel. Therefore it is of utmost importance that sources of such catastrophic faults are identified, so that such events can be identified and, if possible, prevented. The failure mode and effect analysis (FMEA), presented in [6] and [9], is a common technique to identify all possible faults and their consequences. Additionally, the faults are classified by the severity of the consequences associated with this fault. A complementary tool of FMEA is the fault tree analysis (FTA) [9]. The goal of FTA is to identify which sets of basic events, will produce a certain high level faults (e.g., which events must occur for a tank to overflow). These connections and constraints, which are explicit analyzed and described through FMEA and FTA, can be modeled with ontologies.

2.1. Failure Mode and Effect Analysis

The failure mode and effect analysis is an inductive bottom up approach to determine the effects of each failure mode of each system component on the rest of the system. To assure the systematic and thorough coverage of all failure modes the information is normally arranged in tabular format with at least three columns. The first column is reserved for the component name or another unique component identifier. The second column is a description of the currently regarded failure mode, and the third column describes the effects and consequences on the rest of the system. Usually additional columns for the failure detection method, corrective actions, and criticality are introduced. [6] defines four levels of criticality:

- Safe—No major system degradation, no equipment damage or personnel injury
- Marginal—System degradation can be countered or controlled without major damage or injury to personnel
- Critical—Degradation of system performance, damage to equipment, or hazard requiring immediate corrective actions for personnel or equipment survival
- Catastrophic—Severe system degradation with subsequent equipment loss an/or death or multiple injuries to personnel

Afterwards the failure modes identified to be critical or catastrophic are summarized in a critical item list. FMEA can be either hardware or functionality oriented. The hardware oriented approach regards the failure modes of singular hardware items, such as sensors, actuators, interface devices, I/O cards, etc. For example the analysis of a valve would consider the effects on the system if the valve is stuck in “open” position or stuck in “closed” position. The functional approach considers the failure modes of the sub-systems, such as interlocks, trips or control loops. For example, if the function of a control loop is to control the temperature of a liquid the failure modes could be:

- The control loop heats the liquid to a higher temperature than specified
- The control loop heats the liquid to a lower temperature than specified
- The temperature sensor of the control loop reports incorrect temperature values
- The heater of the control loop is defect

2.2. Fault Tree Analysis

The fault tree analysis is a top down approach, where one of the identified failure modes is selected as top event under which the fault tree is developed. All possible cases are identified by a deduction process, until a set of basic events is identified whereby the tree is developed. The combination of those basic events are described by the means of logical AND and OR connections. The size of the tree is influenced by the decision which external causes, such as power failures, air and water supply disruption, are included in consideration.

A combination of events which will cause the top level event to occur is called a cut set. A minimum cut set is the minimal event set which will cause the top level event to occur. Minimum cut sets are identified by using the absorption an idempotence rules of Boolean algebra. It should be noted that there is no guarantee that the minimum cut sets can be identified if the NOT operator, or if an exclusive OR operator is used as an switchable NOT operator, is included in the fault tree. However, it is usually possible to restructure the fault tree so that the use of NOT operators is avoided. If systems get more complex, the size of the fault trees increases and it is not uncommon that a fault tree consists of several thousand cut sets, with sets containing dozens of basic events. Therefore it is common practice to truncate fault trees. Usually large cut sets or cut sets with a probability under a certain threshold are eliminated.

The fault tree synthesis process is essentially a manual task, whereas the analysis, cut set generation, truncation, identification of redundant events, and elimination of redundant events are tool-supported automated tasks. Such tools are usually able to evaluate the probabilities of top level events.

2.3. Ontologies

In general, ontologies are a main part of the semantic web technology and are used like a knowledge representation of the real world or only part of it. Ontologies are formal models of a specific application domain, and primarily used to facilitate the exchange and partitioning of knowledge. More precisely, an ontology is a data model that represents a set of concepts within a domain and their relationships. The word ontology has its origin from the Greek words ontos (=being) and logos (=word). From a philosophical point of view an ontology refers to the subject of existence, that is the study of being as such [3]. Gruber [4] defines an ontology as an explicit specification of a conceptualization. Where a conceptualization illustrates an abstract, simplified picture of the world used for representation and designation. Each knowledge representation follows a certain degree of conceptualization, either explicitly or implicitly. Moreover ontologies can effectively support software development processes, primarily by providing a continuous data model [1].

The emerging field of semantic web technologies promises new stimulus for (Software+) Engineering research. However, since the underlying concepts of the semantic web have a long tradition in the knowledge engineering field, it is sometimes hard for engineers to overlook the variety of ontology-enabled approaches to (Software+) Engineering. Happel and Seedorf [5] propose a simple classification schema that allows a better differentiation among the various ideas of using ontologies in (Software+) Engineering. The Ontology Driven Architecture (ODA) note at W3C serves as a starting point to elaborate a systematic categorization of the approaches and to derive more clearly defined acronyms [11]. Happel and Seedorf propose two dimensions of comparison to achieve a more precise classification. First, they distinguish the role of ontologies in the context of (Software+) Engineering between usage at run-time and development time. Second, they look at the kind of knowledge the ontology actually compromises. Here, they distinguish between the problem domain that the software system tries to tackle, and infrastructure aspects to make the software or its development more convenient.

Semantic systems offer a convenient way for the representation of distributed and linked explicit as well as tacit knowledge. The usage of ontologies for knowledge representation, sharing and high-level reasoning could be seen as a major step towards the area of agent-based control solutions [10]. Exploitation of semantics and ontologies in the area of agent-based industrial systems has become one of the hot topics in the last few years, primarily because of the success and promotion of semantic web technologies to enable better communication between machines and people [12]. Ontologies are considered here as an essential technology for semantic web development guaranteeing data and information interoperability in heterogeneous and content-rich environments [7].

3. Example of Use

As example of use the educational tank model (shown in Fig. 1) of the Odo Struger Laboratory of the Automation and Control Institute of the Vienna University of Technology was used. The first step was a classic UML-based object-oriented analysis to identify the essential objects and their attributes of the tank model components.

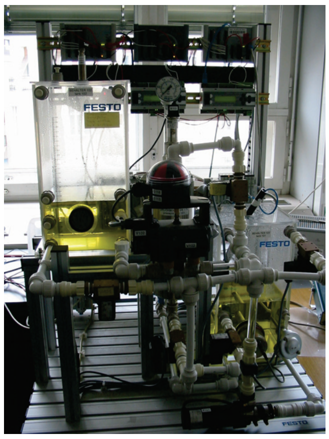
Figure 1. Educational Tank Model

Based on this analysis a basic vocabulary has been identified which is used to declare the core concepts of the ontology. The ontology been created with the help of Protégé[1] (shown in Fig. 2). The identified core concepts are “pipe”, “control”, “actuator”, “sensor”, “tank”, and “pipe connector”.

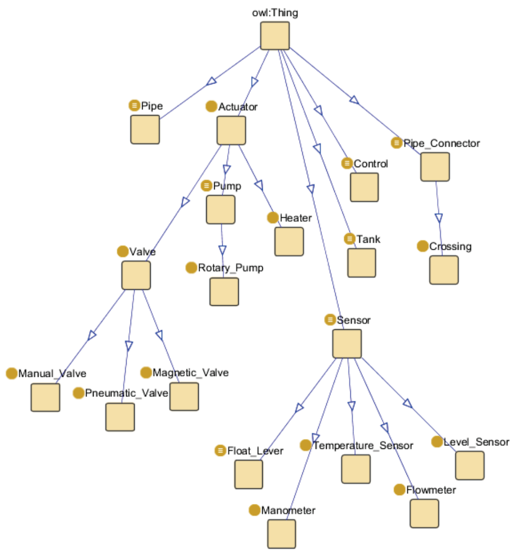

Figure 2. Tank Model Ontology

More specialized concepts are derived from these basic ones (e.g., “pump” is a specialized “actuator”). After the definition of all necessary entity concepts, properties were defined to model the entity relationships. For example the property “contains” defines which entity instance

[1] Available at http://protege.stanford.edu/

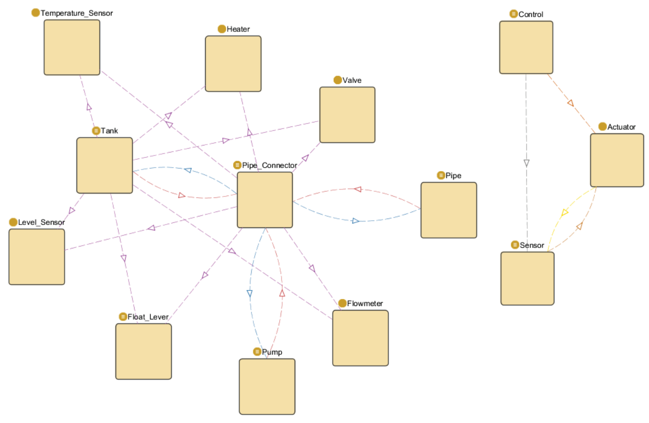

Figure 3. Entity Associations

can be contained in other entity instances. For instance a “tank” can contain a “temperature sensor”, or a “level sensor”, or a “pipe connector” can contain a “valve” in our ontology. However such properties can not differentiate between different entities and therefore it is also possible that a tank can contain a valve. This makes no sense as a tank can be connected to a valve, but will never have one inside it. Therefore entity rules have been defined which prevent such associations. Without going into further detail, Figure 3 shows that even is such a simple ontology the number of associations is high, which represents a large amount of information. The last step was to model the tank model based on the developed ontology. With the help of this minimalistic model we are already capable to detect design faults, such as unconnected pipes.

4. Further Work

The next steps in our work are to enhance the level of detail of the model and to perform a FTA and a FMEA for our tank model. Based on these analysis we will derive the necessary attributes and properties to incorporate the findings into the ontological model. Therefore we will also consider and analyze the work of [2] who has analyzed the core elements of FMEA and already created a first FMEA ontology. [8] already tried a basic approach of ontology-based FMEA for soldering processes. Both works can provide basic insights for our further work. But in contrast to those works which concentrate on the ontological representation of FMEA, our goal is to enable automation devices and the system as a whole to identify its error condition based on the basic error events and to priorize the occurring alarms to support the control room personnel. After that, our main goal will be to integrate our ontology-based fault detection systems onto automation devices, to enable them to analyze their own operational mode. Because of limited processing power, memory size, and run-time constraints it will be necessary to reduce the ontology to a level the device can handle, which will need further study. Such an distributed fault detection approach increases reliability, and simultaneously decreases hardware costs. The future goal for such systems is the ability to predict and prevent their own and system wide failure modes on basis of these ontologies. If an imminent failure mode is detected by the reasoner, appropriate counter measures shall be initiated to prevent the failure mode to occur or the minimize the consequences of such failure modes.

Acknowledgment

We want to thank our partners from academia and industry in the Christian Doppler Laboratory for “Software Engineering Integration for Flexible Automation Systems” for their valuable discussions and feedback. Parts of this work were funded by the Christian Doppler Society and the BMWFJ, Austria.

References

[1] C. Calero, F. Ruiz, and M. Piattini. *Ontologies for Software Engineering and Software Technology*. Springer-Verlag, Berlin, Heidelberg, 2006.

[2] V. Ebrahimipour, K. Rezaie, and S. Shokravi. An ontology approach to support FMEA studies. In *Reliability and Maintainability Symposium, 2009. RAMS 2009. Annual*, pages 407 –411, 26-29 2009.

[3] D. Gasevic, D. Djuric, V. Devedzic, and B. Selic. *Model Driven Architecture and Ontology Development*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5-6):907 – 928, 1995.

[5] H. J. Happel and S. Seedorf. Applications of ontologies in software engineering. In *Proc. of Workshop on Sematic Web Enabled Software Engineering (SWESE) at the ISWC*, pages 5–9, 2006.

[6] J. Love. *Process Automation Handbook: A Guide to Theory and Practice*. Springer-Verlag London, 2007.

[7] M. Merdan. *Knowledge-based Multi-Agent Architecture Applied in the Assembly Domain*. PhD thesis, Vienna University of Technology, 2009.

[8] M. Molhanec and P. Mach. The ontology based FMEA of lead free soldering process. In *Electronics Technology, 2009. ISSE 2009. 32nd International Spring Seminar on*, pages 1 –4, 13-17 2009.

[9] S. Y. Nof. *Springer Handbook of Automation*. Springer Publishing Company, Incorporated, 2009.

[10] M. Obitko and V. Marík. Ontologies for Multi-Agent Systems in Manufacturing Domain. In *DEXA ’02: Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, pages 597–602, Washington, DC, USA, 2002. IEEE Computer Society.

[11] P. Tetlow, J. Z. Pan, D. Oberle, E. Wallace, M. Uschold, and E. Kendall. Ontology driven architectures and potential uses of the semantic web in systems and software engineering. *W3C Working Draft*, 2005.

[12] J. H. Tim Berners-Lee and O. Lassita. The semantic web. *Scientific American*, 284:34–43, 2001.