An Ontology-based model for providing Semantic Maintenance.

Aristeidis Matsokis, Mohamed Hedi Karray, Brigitte Chebel-Morello, Dimitris Kiritsis

An Ontology-based Model for providing Semantic Maintenance

Aristeidis Matsokis*, Mohamed Hedi Karray,**
Brigitte Chebel-Morello, Dimitris Kiritsis***

** Ecole Polytechnique Fédérale de Lausanne, Lausanne, CH-1015 Switzerland (Tel: +41216934359; e-mail: {aristeidis.matsokis, dimitris.kiritsis}@ epfl.ch).*
*** Femto-st, Besançon, F-25000 France (e-mail: {hedi.karray, brigitte.morello}@femto-st.fr)*

Abstract: Maintenance is becoming more and more crucial in Asset Lifecycle Management information models. Issues such as collecting, handling and using the asset data produced during its lifecycle in a lean and efficient manner are on top of today's research. Customer satisfaction, compliance with environmental friendly legislation, product quality, high performance and reliability are only a few of the benefits improved maintenance methods and tools may provide to enterprises. In this work we combine the benefits of two previous developed models and we develop a model for Semantic Maintenance. The first model we are based on is the PROMISE semantic object model which was made for supporting Closed-Loop Product Lifecycle Management. The second model is the semantic model of e-maintenance developed in PROTEUS project. The new model described in this paper is named "SMAC-Model". Its aim is to provide advanced maintenance services as well as feedback for the Beginning of Life and input for End of Life. The model is generic and may be used in various Asset Lifecycle Management cases. It is developed to facilitate complex physical assets and to work in industrial environment.

Keywords: Maintenance engineering, Life cycles,

1. INTRODUCTION

Along the lifecycle of products and assets a huge amount of information is being generated which if exploited and analysed, could be beneficial for the product itself during its usage period, for the new generations of the product as well as for accumulating the useful components of the product when it arrives at its end of life.

In this work we have developed an ontology model for Semantic Maintenance focusing on the collection and the analysis of the maintenance data in order to provide advanced maintenance methods which are beneficial in many ways such as to provide new services, to improve customer satisfaction, to acquire compliance with environmental friendly legislation, and to achieve higher product quality, higher performance and reliability. In order to develop our model, we combined the advantages of two previous developed models. The first model we are based on is the PROMISE Semantic Object Model (SOM) which was made for supporting Closed-Loop Product Lifecycle Management. In this way the data and the information produced from the asset during its Middle Of Life (MOL) is collected and processed to be used as input for improvement of Beginning Of Life (BOL) activities (design, production), and End Of Life (EOL) activities (recycling, remanufacturing, reuse, etc.). Thus, this model allows closing the information loop between the different phases of the lifecycle. The second model is the one developed in PROTEUS project. This model was developed with the aim of being a maintenance-oriented platform providing support for integrating maintenance activities of the MOL and allowing maintenance agents to have remote access to data, to remotely submit requests for diagnosis and to communicate with intelligent systems for better management of maintenance activities.

The new model described in this paper is named "SMAC-Model" and it is an ontology information model developed in the framework of the Interreg Project "SMAC" (Semantic Maintenance and Life Cycle) applied on a lathe machine. Its aim is to provide advanced maintenance services as well as feedback for BOL and input for EOL. The model is generic and may be used in various Asset Lifecycle Management cases. It is developed to facilitate complex physical assets and to work in industrial environment.

The model is developed in the Web Ontology Language with Description Logics (OWL-DL) with the aim of implementing into the model ontology advantages and features which include: data structure layout, description logic (DL) reasoning (Baader et al. (2003)), semantic web rules including Semantic Web Rule Language (SWRL) and Jess rule engine. With the use of these technologies the developed information model is now executable, dynamic and flexible.

In the rest of this work, the following naming conventions are used: names of classes are written in boldface and capitalized/lower case Arial (i.e. **Product_BOL_Supply, Product_MOL, Product_EOL,** etc). Names of attributes are capitalised /lower case Courier New (i.e. `hasParent`) while names of instances are in italics Arial (i.e. *Screw_1*).

The structure of the paper is as follows. Section 2 describes briefly previous works on the PROMISE SOM and the semantic model of e-maintenance of PROTEUS. In section 3 there is a brief description of the ontology SMAC-Model.

2. PREVIOUS WORK

The first model used is a modified version of the SOM developed in the PROMISE FP6 project, developed by Matsokis et al. (2009). The aim of the modified model was to make the model an OWL-DL ontology and use the benefits of the DL reasoner and DL rules. The use of an ontology language makes the model dynamic and allows to record, store and process data-information about a number of systems in a single source. In order to facilitate these characteristics, some modifications to the OWL version of the model were performed. In order to facilitate these new characteristics a number of modifications to the model were performed. The schema of the developed ontology model containing all the classes, sub-classes, attributes and relationships as well as a methodology for extending the model can be found in. For the easier comprehension of the rest of this paragraph, please refer to Fig. 1 presenting the SMAC-Model where the main classes of the SOM are preserved.

The functionality of the SOM is quite simple. Firstly, the list of the physical products is stored in the **Physical_Product** class. The physical products may be complex products which consist of many parts such as vehicles or simple which consist of only one part such as screw. The complexity of the product and its parts is described through a "physical product to physical product" object property `hasParent` and its inverse `isParentOf`. Depending on the requirements of the application the level of detail which is considered as "simple" may vary. Even for the same product, in different cases, we might have different level of detail: i.e. the level of detail is different for products of a fleet management company and different for a single user who might be interested to have more detailed model for the one product he is using. The properties of the products are stored in the **Property** class and the **ID_Info** class (and its sub-classes). Furthermore, each physical product is related to the **Life_Cycle_Phase** class which enables each product to be related to one or more instances of a lifecycle phase i.e. to multiple instances of the MOL. This relationship combined with the object properties `hasParent/isParentOf` allows the information system to track information about the product through its different phases (and types of usage) and therefore, preserve continuity of information about the physical product. Thus, the model stores information about which data is related to the product for each of its use. During its lifecycle the product is monitored with sensors which collect valuable data of different types such as temperature, pressure, velocity, viscosity etc. in various measurement units such as Celsius, bar, m/sec, Pascal-second respectively. The different sensors related to the product are stored in the **Field_Data_Source** class and the types of the data collected are stored in the **Valid_Field_Data_Type** class. The collected data from the sensors is stored in the **Field_Data** class and in documents if necessary. In the **Condition** class it is stored a list of the required or recommended conditions for the well-functioning of the products. These conditions may vary depending on the product and are adjusted according to various criteria. Then, the data of the **Field_Data** class is compared with the conditions. If one or more conditions are not met, one or more events are created and stored in the **Event** class. Events depending on their severity may trigger activities such as maintenance, part replacement etc. which are stored in the **Activity** class. To perform activities several resources are used. The available resources are in the **Resource** class. Finally, activities may cause events (i.e. start, finish, etc.). This part of the model combining activities, events and resources is the part which supports the actual maintenance. Finally, activities cause events (i.e. start, finish, etc.). This part of the model combining activities, events and resources is the part which supports the actual maintenance.

The second model used is a modified version of the semantic model of PROTEUS project (Bangemann et al. (2004) and Rasovska et al. (2006)), developed by Karray et al. (2009). The modifications where judged necessary in order to cover the whole field of maintenance. The final UML ontological model of maintenance consists of twelve parts corresponding to both the structure of the enterprise information system and the maintenance process. These are: the *monitoring management model*; the *site management model*; the *equipment expertise management model*; the *resource management model*; the *intervention management model* which focuses on the maintenance intervention to remedy the equipment failure and is described by an intervention report. It is composed by maintenance activities performed by maintenance actors and create reports for future use; the *maintenance strategy management model* which depends on technical and financial indicators of the maintenance contract for each equipment; the *maintenance management model* which manages the different types of maintenance (corrective, preventive, predictive); the *equipment states model* which has as possible states: Normal state, Degraded state, Failure state, Programmed stop state; the *historic management model* which contains the main data related to the equipment maintenance.; the *document management model*; the *functional management model* which describes the function of the equipment or of the component; the *dysfunctional management model* which stores the characteristics of different failure states of the equipment.

The Proteus model was designed to be a model describing maintenance actions, activities, processes etc and the SOM was designed to provide information continuity among the different life cycle phases. In the next section, the two models are combined to develop the SMAC-Model with the aim of providing semantic maintenance.

3. DEVELOPED SMAC-MODEL

The concept is to combine the Closed Loop-PLM SOM with the Proteus e-maintenance platform in order to provide more complete maintenance model applicable in the entire lifecycle. The SMAC-Model contains classes and relationships from both previous models. Moreover, it

includes new classes, relationships (object properties) and attributes (datatype properties) in order to increase the capabilities of the model. These new elements derive from the fact that after combining the two models new opportunities were created, and hence, the model was extended to support them. The SMAC-Model developed is shown in Fig. 1. The most important extensions in the model are described in the following paragraphs.

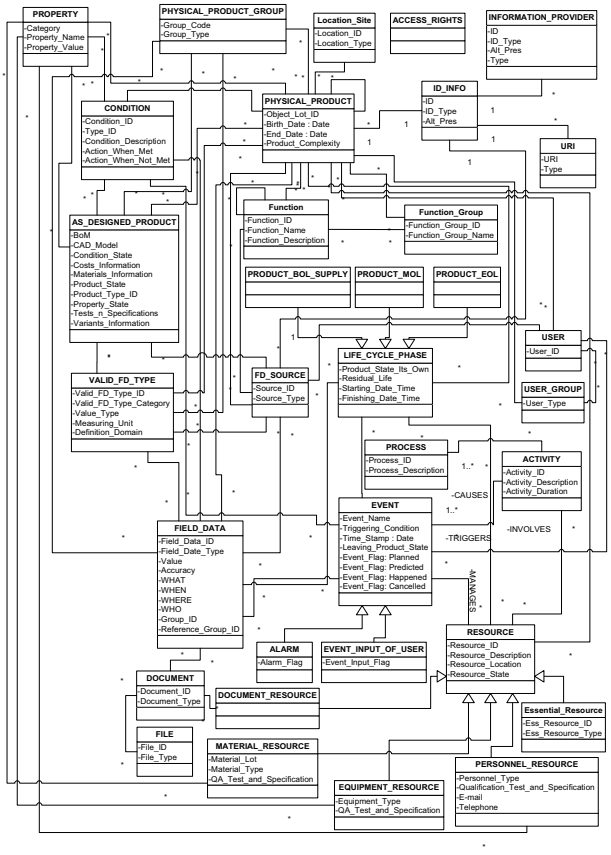

Fig. 1 The developed SMAC-Model.

3.1 Expansion in Classes

Each class in the model describes a concept of the real life. Therefore, the model was extended in classes in order to increase the described concepts. The classes added to the model were **Location_Site**, **Essential_Need**, **User**, **User_Group**, **Function**, **Function_Group**, **Alarm** (as critical event), **Event_Input_of_User** and **Process**. Their usefulness is described in this paragraph.

The **Location_Site** class was added. It is related to **Physical_Product** via the object property `Location_Site2Physical_Product` (and its inverse). Its datatype attributes are `Location_ID` and `Location_Type`. The information stored in this class is for instance the geographical location of a manufacturing plant. This is important for the better management of resources during maintenance, since we know the location of the product and we can use the closest maintenance facilities possible. Furthermore, any maintenance activities will have to be compliant with local regulations such as noise levels, pollution levels, recycling, etc.

The **Essential_Resource** class was added as a sub-class of **Resource** class. Its datatype attribute is `Ess_Resource_Type`. This class describes the requirements of the physical product regarding infrastructure and its environment in order to be ready to perform its functions. Such needs could be power supply, water supply, gas supply, oil supply, sunlight, etc.

The **User** and **User_Group** were added. They are related to each other through the object property `User2User_Group` (and its inverse), and to **Physical_Product** via `User2Physical_Product`, `User_Group2Physical_Product` (and their inverse relationships) respectively.

- The idea behind the **User** class is that the user will be the "client" or "customer" who is buying the service of using the physical product/machine on contract: i.e. when one rents a car from a car rental provider, he is the user for a certain time period or/and a limit in km. Similarly, a company may “rent” a product for certain working hours with leasing and perhaps adjust it to the needs of the user.

- The **User_Group** describes elements such as the type of maintenance performed by the user. Thus, we can have groups according to their maintenance contract type or the type of industry the machine is used in (for the use of the machine) i.e. form steel or aluminum parts. In the previous model this was only referred as a **Document_Resource** class and it was declared through a datatype attribute.

The **Function** and **Function_Group** classes were added. They are related to each other through the object property `Function2Function_Group` (and its inverse), and to **Physical_Product** via `Function2Physical_Product`, `Function_Group2Physical_Product` (and their inverse relationships) respectively.

- The idea behind the **Function** class is that each physical product may have one or more functions (i.e. rotation, linear movement, store coolant liquid, etc.). Therefore, whenever a physical product is degraded one or more of its functions are affected. Through the connection of this class with the **Physical_Product** we are able to know which functions are related to each individual physical product and they are or may be affected during its degradation.

- The **Function_Group** is used to describe functions at a generic level i.e. group all material of heat isolation of the product, group all rotating parts of the product, etc.

The **Alarm** class was added as a sub-class of **Event** class. This class contains only the critical events. The system issues events some of which depending on their criticality may be

evaluated as alarms. Alarms may cause the breakdown of a function of the physical product.

Event_Input_of_User class was added as a sub-class of **Event** class. This class contains only Events which are input by a user. These events may have been caused by:

- The fact that there is place for improvement in the monitoring system i.e. we have not installed a sensor at a place where we should have it. In that case it gives us feedback for possible weaknesses of the system.
- The slow response of the system in an abnormal situation.
- An external factor i.e. in case of flood or fire in the building.

Finally, the **Process** class was added. It is related to **Activity** via the object property `Process2Activity` (and its inverse). The meaning of a process in this context is that a process consists of one or more activities and its task is to group together the maintenance activities. This was required in order to accumulate knowledge about which activities are performed per process and make the system capable of automatically listing the activities needed depending on the events.

3.2 Expansion in Relationships

After extending the domain coverage of the model with the new classes, some more relationships were added to describe the links between the classes. These relationships compose the active network of communication of the model since they are used like verbs of the sentences in a structure noun-verb-noun which in the model is class-relationship-class. In a later stage they may be used as the basis for introducing DL rules in the model.

The **Condition** class was related to **Event** via the relationships `Condition2Event` and its inverse `Event2Condition`. Thus, the model can describe the condition(s) that trigger an event and relate them directly.

The **User** class was related to **Event** and to **Field_Data_Source** via `User2Event`, `User2FD_Source` (and their inverse relationships) respectively.

- The relationship `User2Event` describes the case where a user notices some malfunction and makes an action. In this case an event instance is created and it is related to a user instance. This may provide feedback and may be a source for reporting bugs of the monitoring system.
- The relationship `User2FD_Source` describes the case where a user notices some malfunction and acts as a field data source. In this case the user inputs data at the **Field_Data** class. Then this data will be evaluated by the system and an event instance may be created depending on the conditions.

The **Function** class was related to **Field_Data_Source** via `Function2FD_Source` and its inverse relationship. Moreover, the **Function** class is related to itself with the relationship `FunctionIsComposedBy` and its inverse `FunctionComposes`.

- The relationship `Function2FD_Source` is used to relate the function with the sensor of the **Field_Data_Source**. Thus, when an alarm is created from the field data collected by a specific sensor, the system knows which functions are or may be affected.
- The relationship `FunctionIsComposedBy` describes the case where a function is composed by a number of sub-functions. Similarly its inverse describes which sub-functions are composing more complicated functions.

3.3 Expansion in Attributes

After making the changes in the classes and the relationships, some more datatype attributes were added to describe various requirements. These were:

Attributes `Group_Code` to facilitate the unique number of the group and `Group_Type` to facilitate the description of the group were added to the **Physical_Product_Group** class.

Attribute `Condition_Description` was added to the **Condition** class. This attribute contains a short description of the condition.

In the **Alarm** class, for the better management of alarms, `Alarm_Flag` was added. There are two levels of alarm described through the datatype attribute `Alarm_Flag`:

- Yellow alarm (the function is likely to fail ~50-75%).
- Red alarm (the function is likely to fail >75%).

  These likelihoods are estimated on real time and should be coordinated with time. For example the likelihood of ~50% for the Axis X1 drive to fail in the next 5 working hours might be a red alarm, whereas a likelihood of ~50% for the Axis X1 drive to fail in the next 500 working hours might be a yellow alarm.

In the **Event_Input_of_User** class the attribute `Event_Input_Flag` was added and it may have the following values:

- Weakness Factor: this describes the fact that there is place for improvement in the monitoring system either by adding new sensors or by performing activities for improving the response of the system in abnormal situations. It should be noted that the user is not an expert and therefore the exact weakness factor has to be defined by the cause/fault/data analysis.
- External Factor: this describes a factor coming from the environment of the product i.e. in case of flood or fire in the building, black-out etc.

Attributes `Function_Group_ID` and `Function_Group_Name` were added to the **Function_Group** class in order to describe the elements of this class.

The attributes added to the **Function** class were `Function_ID`, `Function_Name` and `Function_Description`.

Attribute `User_Group_ID` and `User_Group_Type` were added to the **User_Group** class in order to describe the elements of this class.

The attributes added to the **User** class were `User_ID` and `User_Type`.

The attributes added to the **Process** class were `Process_ID` and `Process_Description`.

All these new attributes are supposed to describe better the various aspects of the maintenance of the product and like the relationships they may be used for introducing DL rules in the model.

4. DISCUSSION-CONCLUSIONS

The SMAC-Model is developed in OWL-DL and it is extended in comparison to the previous models in all aspects: classes, relationships and attributes. This is performed in order to describe the MOL of the products with higher accuracy. The model is generic and extensible to fulfil the user's requirements.

In the near future a case study will be performed to demonstrate how the user may extend the generic classes of the model with sub-classes to describe better his needs and at the same time maintain data integration and interoperability among the variations of the initial model. The user is required to extend the model using DL rules according to the methodology developed by Matsokis et al. (2009). Thus, the concept described by every new sub-class is defined by its own DL rules in a machine-understandable manner. Then, the DL reasoner may be used in order to find equivalencies and consistency of the classes and to re-classify the class hierarchy. When applied on the instances (such as products, field data, etc.) the reasoner categorises them under the right classes in the model.

Future work on the model includes the mapping of the SMAC-Model with MIMOSA OSA-CBM which will possibly increase its re-usability.

ACKNOWLEDGEMENTS

This work was carried out and funded in the framework of SMAC project (Semantic-maintenance and life cycle), supported by Interreg IV programme between France and Switzerland. We also appreciate the support of the research project IMS 2020 (Intelligent Manufacturing Systems for 2020).

REFERENCES

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press.

Bangemann, T., Rebeuf, X., Reboul, D., Schulze, A., Szymanski, J., Thomesse, J.P., Thron, M. and Zerhouni, N. (2006). PROTEUS-Creating distributed maintenance systems through an integration platform. *Computers in Industry,* volume 57, issue 6, pp. 539-551

Byington, C.S., Kalgren, P.W., Dunkin, B.K. and Donovan, B.P. (2004). Advanced diagnostic/prognostic reasoning and evidence transformation techniques for improved avionics maintenance. *Proc. of the IEEE Aerospace Conference*; 13 March, 2004, Big Sky, Montana, USA, volume 5, pp. 3424-3434.

Chidambaram, B., Gilbertson, D.D. and Keller, K. (2005). Condition-based monitoring of an electro-hydraulic system using open software architectures. *Proc. of the IEEE Aerospace Conference;* 5-12 March, 2005, Big Sky, Montana, USA, pp. 3532 - 3539.

Karray, M.H., Morello-Chebel, B. and Zerhouni, N. (2009). Towards a maintenance semantic architecture. *The 4th World Congress on Engineering Asset Management (WCEAM),* 28-30 Sep. 2009 Athens, Unpublished.

Mathew, A., Zhang, L., Zhang, S. and Ma, L. (2006). A Review of the MIMOSA OSA-EAI Database for Condition Monitoring Systems. *Proc. of the 1st World Congress on Engineering Asset Management (WCEAM)*; 11 – 14 July 2006, Gold Coast, Australia, pp. 837-846

Matsokis, A. and Kiritsis, D. (2009). An Ontology-based Approach for Product Lifecycle Management. *Computers in Industry. Special Issue, Semantic Web Computing in Industry,* Unpublished.

Matsokis, A., Zamofing, S. and Kiritsis, D. (2010). A Ontology-based Modelling for Complex Industrial Asset Lifecycle Management: a Case Study. *In Proc. the 7th International Product Lifecycle Management Conference*; 12-14 July, 2010, Bremen, Germany, Unpublished.

MIMOSA (Machinery Information Management Open Systems Alliance) www.mimosa.org, March 2010.

Muller, A., Crespo Marquez, A. and Iung, B. (2008). On the concept of e-maintenance: Review and current research. *Reliability Engineering and System Safety*; volume 93, issue 8, pp. 1165-1187

PROMISE FP6 project: www.promise.no, December 2009.

Rasovska, I., Chebel-Morello, B. and Zerhouni, N. (2004). A conceptual model of maintenance process in unified modeling language. *Proc. of the 11th IFAC Symposium on Information Control Problems in Manufacturing.* INCOM'04 5-7 April 2004, Salvador, Brazil

Rasovska, I., Chebel-Morello, B. and Zerhouni, N. (2005). Process of s-maintenance: decision support system for maintenance intervention. *Proc. of the 10th IEEE conference on emerging technologies and factory automation*; volume 2, Catania, Italy. pp. 679-686.

Voisin, A., Levrat, E., Cocheteux, P. and Iung, B. (2010). Generic prognosis model for proactive maintenance decision support- application to pre-industrial emaintenance test bed. *Journal of Intelligent Manufacturing*; volume 21, issue 2, pp. 177-193.