Towards Model Transformation from a CBM Model to CEP Rules to Support Predictive Maintenance

Alexandre Sarazin, Sébastien Truptil, Aurelie Montarnal, Jacques Lamothe, Julien Commanay, Laurent Sagaspe

Towards Model Transformation from a CBM Model to CEP Rules to Support Predictive Maintenance

Alexandre Sarazin[1,2], Sébastien Truptil[1], Aurélie Montarnal[1], Jacques Lamothe[1], Julien Commanay[2] and Laurent Sagaspe[2]

[1]*Industrial Engineering Center, IMT Mines Albi, Albi, France*

[2]*Digital & Software Department, APSYS, Blagnac, France*

Keywords: Maintenance, Knowledge Base, Model Transformation.

Abstract: Over the past decades, the development of predictive maintenance strategies, like Prognostics and Health Management (PHM), have brought new opportunities to the maintenance domain. However, implementing such systems addresses several challenges. First, all information related to the system description and failure definition must be collected and processed. In this regard, using an expert system (ES) seems interesting. The second challenge, when monitoring complex systems, is to deal with the high volume and velocity of the input data. To reduce them, Complex Event Processing (CEP) can be used to identify relevant events, based on predefined rules. These rules can be extracted from the ES knowledge base using model transformation. This process consists in transforming some concepts from a source to a target model using transformation rules. In this paper, we propose to transform a part of the knowledge from a condition-based maintenance (CBM) model into CEP rules. After further explaining the motivations behind this work and defining the principles behind model-driven architecture and model transformation, the transformation from a CBM model to a "generic rules" model will be proposed. This model will then be transformed into an Event Processing Language (EPL) model. Examples will be given as illustrations for each transformation.

1 INTRODUCTION

According to european stantards (AFNOR, 2018), maintenance is defined as the "combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function". When a maintenance action is carried out in order to assess and/or to mitigate the degradation and reduce the probability of failure of an item, the maintenance is said preventive. Condition-based maintenance (CBM) is hence defined as a specific kind of preventive maintenance including the assessment of the systems physical conditions, their analysis and the determination of possible ensuing maintenance actions.

Condition-based maintenance has become a very dynamic research topic these past few years, motivated by the constant increase in the systems complexity, the higher quality and safety requirements and, consequently, the augmentation of time-based preventive maintenance actions cost (Jardine et al., 2006). In particular, the concept of PHM has emerged as "a method that permits the reliability of a system to be evaluated in its actual life-cycle conditions, to determine the advent of failure, and mitigate the system risks" (Vichare and Pecht, 2006). It is based on the observation of the systems sensor data in order to provide a complete, real-time vision of the systems health to anticipate potential degradations. The early warning signs detected from the sensor data allow to performing failure detection and estimate the Remaining Useful Life (RUL) of the system as the degradation duration before a failure occurs (Blanchard et al., 1995; Lee et al., 2014).

A PHM approach is comprised of seven main steps (Lebold et al., 2003) :

1. Sensor-based data acquisition
2. Data preprocessing
3. Condition assessment / Anomaly detection
4. Failure identification / Diagnostic
5. Prognostics / Remaining useful life (RUL) estima-

Sarazin, A., Truptil, S., Montarnal, A., Lamothe, J., Commanay, J. and Sagaspe, L.
Towards Model Transformation from a CBM Model to CEP Rules to Support Predictive Maintenance.
DOI: 10.5220/0008880302050215
In *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2020)*, pages 205-215
ISBN: 978-989-758-400-8

tion

6. Decision support
7. Human-Machine Interface

The data acquisition step is dedicated to the monitoring of the system in its actual life cycle conditions. Before further analysis, a first processing has to be performed in order to improve data quality through cleaning or transform the raw data into more relevant variables. Once the data have been preprocessed, the health of the system has to be assessed by anomaly detection. The detected anomalies are then analysed in order to identify the root cause and failure. This step is called diagnostic. If the anomalies detected are related to the future occurrence of a failure, the RUL of the system can be evaluated during the prognostics phase. The results of these analysis are then transmitted to a decision support system and communicated to the end user through a human-machine interface.

2 OVERVIEW OF THE PROPOSED PHM APPROACH

Among the numerous solutions proposed to implement a PHM approach (Jardine et al., 2006; Lee et al., 2014; DePold and Gass, 1999; Gertler, 1998; Vachtsevanos et al., 2006; Lee et al., 2017), a possible solution is to use an expert system (ES). An ES is a computer program in which expert knowledge is implemented for a specific topic in order to solve problems or provide some advice (Jackson, 1998). The purpose of these systems is to mimic the experts reasoning by performing an analysis based on facts to deliver a conclusion (Levine and Pomerol, 1990). This approach is hence based on the formalization of human knowledge and the mechanisms allowing to perform its exploitation. For maintenance applications where experts knowledge is often available and always valuable, there is great value in incorporating the capitalisation of this experience in maintenance solutions like PHM. ES are composed of three main components : a user interface, an inference engine and a knowledge base (Liebowitz, 1995). The user interface allows the user to interact with the expert system. The knowledge base is a set of facts and rules related to the domain. This component may be composed of a static and dynamic database (Kalogirou, 2003). The static database contains the domain knowledge implemented by the human expert which is stable although new facts and rules may be added or modified. The main interest of using an ES lies in the content of this database, as such, its must be complete, consistent and accurate. The dynamic one, however, changes on a higher frequency as it is used to "store all information obtained from the user, as well as intermediate conclusions (facts) that are inferred during the reasoning" (Kalogirou, 2003). Its content is lost at the end of each execution. This knowledge base is processed by an inference engine in order to reach a conclusion. It serves as a "control structure [...] that allows the expert to use search strategies to test different hypotheses to arrive at expert system conclusions" (Liebowitz, 1995). Using an ES can thus be considered a solution to capitalize and exploit the available knowledge on the system. In the maintenance context, the rules for anomaly detection, diagnostic and prognostic must be applied to the input data in order to identify failures and estimate the RUL.

However, managing the input data is one of the challenges in implementing CBM, especially to complex systems. Complex systems, like aircrafts, require many different sensors to perform an acceptable monitoring. The volume and velocity of these inputs are characteristics of big data problems defined through the 5V (Jin et al., 2015) : volume of the collected data, velocity of its update, veracity of the information, variety of the sources and value of the information. In order to reduce the volume and velocity of the input data and processe them, a solution is to use Complex Event Processing (CEP). CEP are designed to monitor large amounts of data in real time and detect patterns based on predefined rules. This solution is relevant in reducing the flow of data to allow further analysis such as diagnostics or prognostics. The main requirement of this solution is the ability to provide observation rules as configuration. These observation rules being present in the ES knowledge base, a critical step is to automatically extract and transform them in an Event Processing Language (EPL) to make it usable by the CEP (Fig. 1).

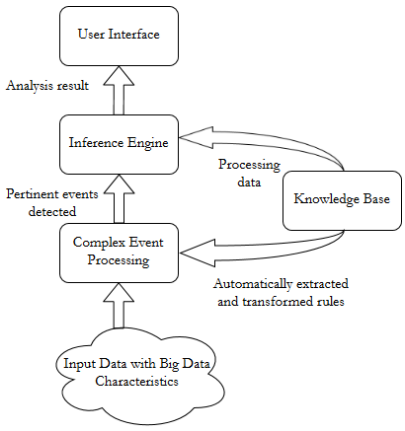

Figure 1: Relation between CEP and Expert System.

To adress this problem, this paper proposes a model

transformation from a CBM model to CEP rules. This transformation will be performed in two steps : The first step is to transform the CBM model into "generic rules" and from "generic rules" to CEP rules. The contribution here lies in the definition of the "generic rules" using a metamodel and the description of the transformation rules between the three models. This methodology has been conducted in accordance with the principles of Model-Driven Architecture (MDA)(Belaunde et al., 2003).

3 MODEL-DRIVEN ARCHITECTURE AND MODEL TRANSFORMATION

A model is defined as a "formal specification of the function, structure and/or behavior of an application or system" (Belaunde et al., 2003). A critical aspect of a model is the point of view chosen to represent it. Indeed, a single system can be represented by a multitude of different models as long as the point of view changes (Bezivin and Briot, 2004). A model is most of the time created in accordance with a metamodel, which is an "explicit specification of an abstraction" (Bezivin and Gerbe, 2001) or a "model of models" (Belaunde et al., 2003). It is used to define the concepts manipulated in a model and the relations between them. A model can thus be considered as an instance of a metamodel.

The main motivation behind the Model Driven Architecture (MDA) approach is to differentiate the business concepts manipulated from the technological platform used to implement them. In software development, this allows to differentiate the steps of design from the business logic to the technical implementation. According to MDA guide, this methodology improves the "portability, interoperability and reusability" of the end result. The MDA methodology is centred around four main categories of models (Belaunde et al., 2003) :

- The Computation Independant Model (CIM)
- The Platform Independent Model (PIM)
- The Platform Model (PM)
- The Platform Specific Model (PSM)

The CIM is the model designed by a domain expert with its own vocabulary and without any kind of implementation solution. The PIM includes a first level of specification making it suitable for different platforms of the same kind. The PM describes the platform used to implement the model. It describes all the different parts of the platform and the services related to it. Finally, the PSM is defined as a "view of a system from the platform specific viewpoint". It is the step beyond the PIM that includes some specificities of a particular platform (Belaunde et al., 2003).

MDA methodology consists of passing from CIM to PIM and from PIM to PSM. The process of converting a model into another of the same system is called model transformation (Belaunde et al., 2003). A model transformation can also be defined as "a transformation operation Mt taking a model Ma as the source model and producing a model Mb as the target model". All models must conform to their own metamodels, conforming themselves to a metametamodel (Bezivin et al., 2006). In a model transformation approach, all concepts are not commonly shared by the source and target models. In fact, the first step of a model transformation is to identify, in each model, the shared concepts, where lies the domain of the transformation, and the specific concepts, which are not shared (Fig. 2). The shared concepts are then transformed from the source to the target model using transformation rules (mapping rules). In the source model, the specific part can thus be assimilated to capitalized knowledge while the specific concepts of the target model correspond to additional knowledge that has to be implemented from external sources(Truptil et al., 2010).

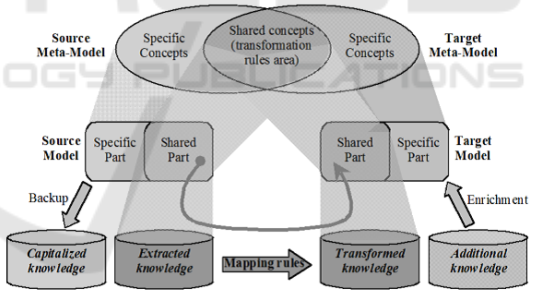

Figure 2: "Model transformation principle" (Truptil et al., 2010).

In this paper, the CBM model and the "generic rules" model can be assimilated to CIMs, while the model for the CEP rules refers to a PIM as EPL is a commun to different CEP languages such as EQL, CQL, SteamSQL or CCL. The point of this paper is to define the "generic rules" CIM and explicit the transformation rules between the two CIMs and from the "generic rules" to the PSM (Fig 3).

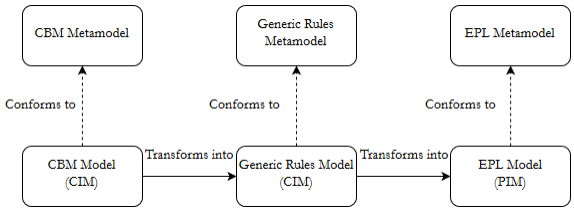

Figure 3: Model transformation from a CBM to EPL rules.

4 FROM A CBM MODEL TO GENERIC RULES

4.1 Presentation of the CBM Metamodel

The metamodel chosen to define the CIM has been designed by Guillèn et al. (Guillen et al., 2016). This metamodel describes the different parts of a CBM solution with concepts defined from ISO standards (Fig. 4). In this metamodel, a CBM solution is divided into 5 parts:

- Physical Description
- Functional Description
- Information Sources
- Symptom Analysis
- Maintenance Decision-Making

The "Physical Description" part regroups all information related to the systems composition. It describes the different components to the maintainable items level which are "the group of parts of the equipment unit that are commonly maintained (repaired/restored) as a whole" (ISO, 2017).

The "Functional Description" part lists the functions assured by the equipment units. For each function, the functional failures associated are also described and the failure modes for each functional failure are provided.

The "Information Sources" block describes the elements related to information gathering such as sensor data, variables, and measurement techniques. These elements are used to generate monitoring variables designed as basis for analysis.

The forth part is the "Symptom Analysis" block which defines the descriptors, symptoms and information rules. A descriptor is a "feature, data item derived from raw or processed parameters or external observation" (ISO, 2012). They are obtained from the processing of one or several monitoring variables. A symptom is a "perception, made by means of human observations and measurements (descriptors), which may indicate the presence of one or more faults with a certain probability" (ISO, 2012). Finally, an interpretation rule is "the description of how the descriptor values have to be interpreted or treated in order to get the monitoring outputs (detection, diagnosis, prognosis) for a failure mode" (Guillen et al., 2016).

The last part, the "Maintenance Decision-Making" block, supports the anomaly detection, diagnosis and prognosis activities. These actions are triggered by the activation of the interpretation rules and provide a list of potential maintenance decisions. The detection element is designed to analyse the state of the system and detect abnormal behaviours. Its purpose is to confirm the occurrence of a failure and reduce the rate of false positives. The diagnosis element refers to the failure identification activity. It is defined as a "conclusion or group of conclusions drawn about a system or unit under test" (ISO, 2012). Its purpose is to identify the deficient component and the source and/or nature of the failure. The prognosis is the "estimation of time to failure and risk for one or more incipient failure modes" (ISO, 2012). Its purpose is to estimate the remaining duration before a failure occurs also called Remaining Useful Life (RUL) and anticipate the emergence of new risks.

The relevance of this model comes from the exhaustiveness of the concepts, all based on standards, and the relations between these concepts required to perform CBM.

4.2 Presentation of the "Generic Rules" Metamodel

As illustrated in Fig.3 the target model of the first model transformation describes the structure of generic rules. This model serves as a medium between the CBM model and the EPL model. There are three purposes behind this model.

First, it reduces the scope of the CBM to observation rules only, discarding any maintenance specific concept. This way, the model allows generic rules to be defined for a larger scope of models and may not be restricted to maintenance only. As it is independent from the source model, it is also resilient to any modification of the CBM model and only the transformation rules would be affected by such modifications.

The second purpose is related to the second transformation. As the generic rules are not event-oriented, they are also independent from the EPL model, improving the architectures flexibility by allowing no-EPL rule-based solutions to be more easily implemented.

The third purpose for designing this model for generic rules is to allow 1-to-n transformations for the second transformation, meaning that a single generic

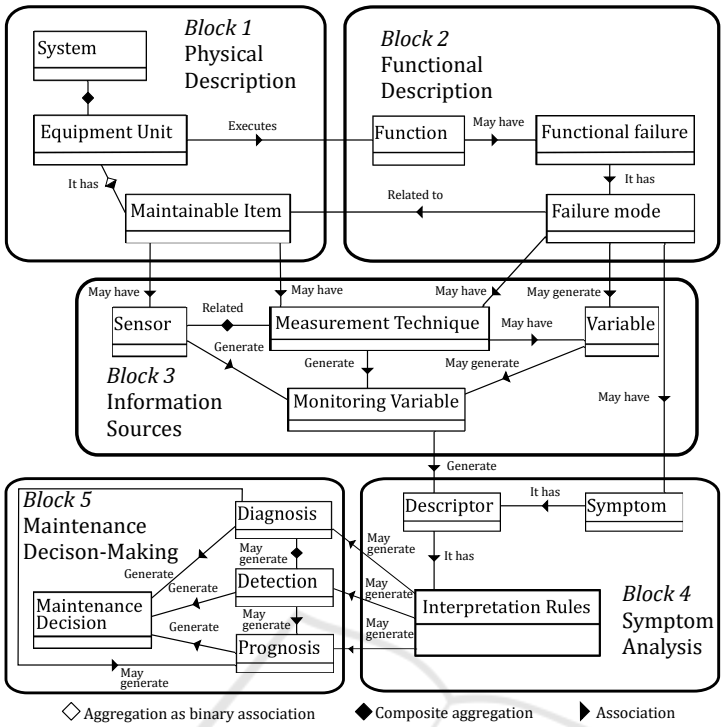

Figure 4: "Basic structure for the CBM solution presented in an UML diagram" (Guillen et al., 2016).

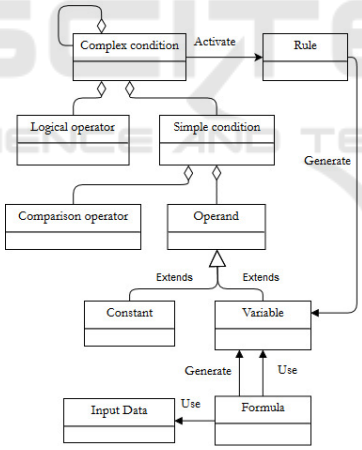

Figure 5: Generic Rules Model.

rule can generate several EPL rules.

In this model (Fig. 5), a generic rule is assimilated to a complex condition. This complex condition can be generated by aggregation of other conditions, either complex or simple, with logical operators (for instance and, or, not). A simple condition, which is the elementary form of condition, is composed of operands linked by a comparison operator (equal, greater than etc...). An operand can be a constant or a variable generated from input data processing. The processing is described here as the generation of a variable by a formula using input data and/or other variables as input. When all conditions are fulfilled, the rule generates one or several output variables.

4.3 Transformation Rules and Illustration

Once the source and target models and defined, the model transformation can begin. As previously mentioned, the first step is to identify the concepts shared by both models. In the source model, the relevant elements for a rule definition are:

- the monitoring variable which can be considered as inputs
- the descriptors which define how the monitoring data should be processed
- the symptoms for rule characterization
- the interpretation rules to define the activation conditions including operators and threshold values
- the detection, diagnosis and prognosis elements to indicate which actions should be triggered by the rule activation

In order to connect these elements to concepts of the target model, transformation rules have been generated and summed up in Table 1. According to this

Table 1: Transformation Rules from a CBM Model into Generic Rules.

CBM Model Concept	Generic Rule Model Concept
Monitoring Variable	Input Data
Descriptor	Variable
Descriptor	Formula
Symptoms	Rule
Interpretation Rule	Complex Condition
Detection	Variable
Diagnosis	Variable
Prognosis	Variable

mapping, the monitoring variables correspond to the input data as they refer to elemental signals or information. The descriptors are related to two of the targets concepts. In the source model, this component is a "data item derived from raw or processed parameters". In order to define it, intermediate variables must be generated using functions. It is therefore related to the Formula and Variable Operand elements of the target model. The Symptom component provides business logic on the state of the system depending on the activation of the conditions activation. It can thus be mapped to the target models Rule. The Interpretation Rules function is to compare the descriptors to threshold values. As such it is composed of Comparison Operators and Constant Operands. The Detection, Diagnosis and Prognosis elements are the rules output. In the source model, they refer to the actions performed once the conditions of the Interpretation Rule are fulfilled. In the target model, they are assimilated as Variables indicating the nature of the actions to perform. In the global architecture, these actions should then be performed by the expert systems inference engine.

Two illustrations of this transformation are proposed based on the use case of an industrial power transformer. These examples have been defined by Guillèn et al. in order to illustrate the CBM model. They will here be transformed into generic rules and into EPL rules in the next chapter in order to illustrate the second model transformation. Following this approach, these examples will be processed by both model transformations, turning the elements of the knowledge base into EPL rules. For both examples, the shared concepts and related failure modes are described in Table 2.

The first example describes the detection of the failure mode "lack of outflow" of the maintainable item "Refrigeration". The symptom to be detected is an abnormal correlation between the oil temperature and the current output of the power transformer. In order to detect this symptom, the monitored variables are the upper and lower oil layer temperature, the load current in output and the number of maintenance interventions. These monitoring variables are then processed using a reliability function to generate a descriptor. When the descriptor, alias the value the reliability function, becomes inferior to 20%, a prognosis action is then triggered. A model of this example according to the source model is represented in Fig 6. Applying the above mentioned transformation rules to this model, the corresponding target model has been designed in Fig 7. The particularity of this example is the aggregation a several monitoring data into a single descriptor using complex formulas. As a reminder, these formulas are hidden inside the Descriptor element of the source model.

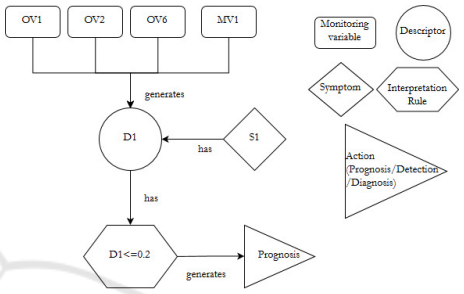

Figure 6: First Example of a CBM model adapted from (Guillen et al., 2016).

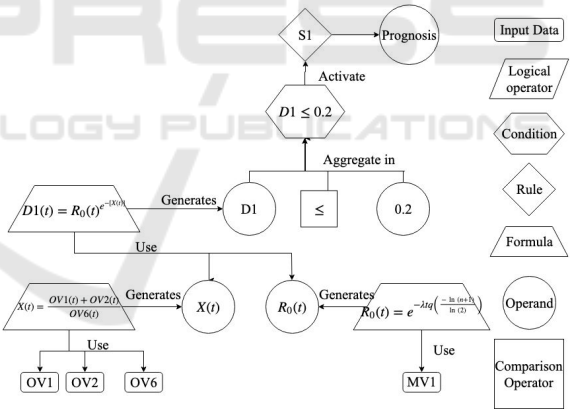

Figure 7: First Example of a Generic Rule Model.

The second example concerns the detection of the failure mode "short-circuit" of the power transformers core. the symptom observed is the combination of an anomaly between the over-current and service current and the presence of hydrogen and CO generated by the short-circuit. The monitored variables are the quantity of hydrogen compared to normal service values (OV5), the load current(OV6) and the quantity of CO (OV8). The particularity of this example is that each monitoring variable is a descriptor. The interpretation rule states that OV5 must be above 50% and the load current and CO level must be positive in order to activate the rule and perform detection and diagnosis

Table 2: Presentation of the Shared Concepts for two Examples of Failure Modes adapted from (Guillen et al., 2016).

Failure Mode	Symptom	Monitoring Variable	Decriptor	Action	IR
Lack of outflow	S1 - Relation of the oil temperature and the current output in the power transformer	OV1 - Upper oil layer temp. (°C) OV2 - Lower oil layer temp. (°C) OV6 - Load current (A) MV1 - Number of maintenance interventions	D1 = R(t,OV1, OV2, OV6, MV1)	Prognosis	D1≤0.2
Short circuit	S2 - Over-current considerably higher than service current, confirmed also by the presence of hydrogen and CO as a result of the arc	OV5 - Hydrogen level (%) OV6 - Load current(A) OV8 - CO level (%)	D5 = OV5 D6 = OV6 D7 = OV7	Detection, Diagnosis	D6 ≥0.5 & D5 ≥ 0 & D7 ≥ 0

activities. The instantiation of the source model with this example is represented in Fig. 8. Its transformation into a generic rule, according to the transformation rules in Table 1, is described in Fig. 9.

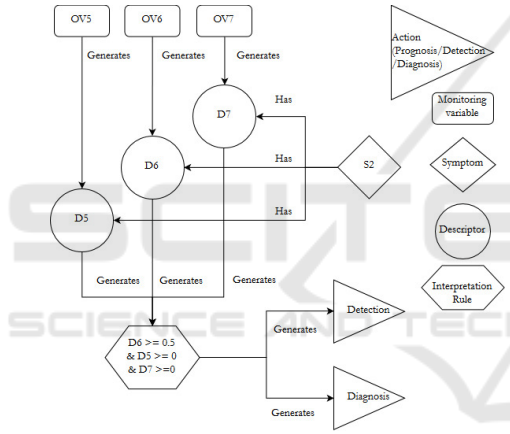

Figure 8: Second Example of a CBM Model adapted from (Guillen et al., 2016).

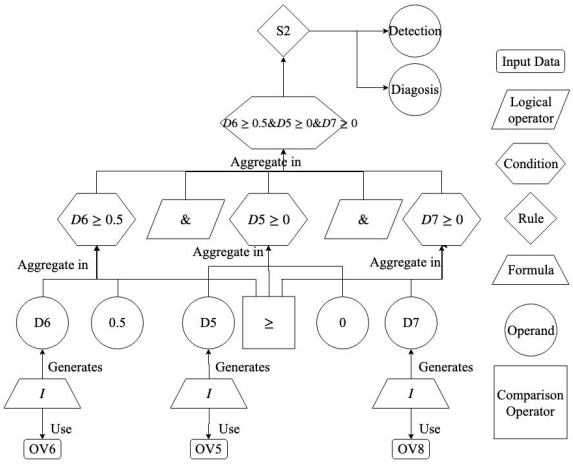

Figure 9: Second Example of a Generic Rule Model.

These examples illustrate how the mapping rules presented in Table 1 can be used to perform the transformation process from an expert system knowledge base, defined here using the CBM model of Guillèn et al. , into generic rules.

5 FROM GENERIC RULES TO AN EPL MODEL

5.1 Presentation of the EPL Metamodel

Once the generic rules are defined, they have to be transformed to comply with Event Processing Languages (EPL). EPLs are SQL-like languages designed support CEP solutions by defining events, conditions and patterns in order to detect interesting behaviors in the data (Boubeta-Puig et al., 2014). Esper or Siddhi are some examples of well-known EPL-based solutions.

According to Boubeta-Puig et al. (Boubeta-Puig et al., 2014), one of the downsides of these systems is their first hand complexity. In order to ease the domain experts work in implementing CEP solutions despite the lack of EPL knowledge, a model for EPL and an automatic model-to-code solution have been designed (Boubeta-Puig et al., 2014). This model-to-code implementation can be assimilated as a PIM to PSM transformation, the PIM being the EPL model and the PSM being the generated code. In order to rely on the above mentioned work, the target model for the model transformation is the model presented in Fig.10.

In this model, the EPL is composed of three main types of components : the “SearchConditions” component, the “Pattern” component and the “Output” component. The “Link” component is designed to establish the relationships between the elements contained in the three main components.

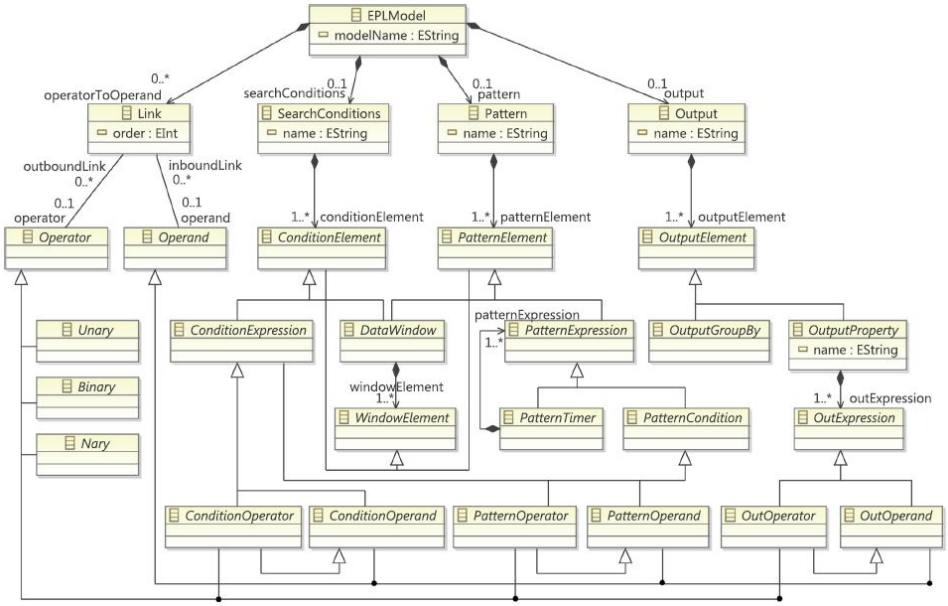

Figure 10: "EPL Metamodel" (Boubeta-Puig et al., 2014).

The "Link" component is divided into operands and operators. Operators correspond here to the operation being performed on one or several operands. These operators can either be Unary, Binary or N-ary depending on the number of operands them can be applied on.

The "SearchConditions" component regroups the "ConditionsElements". These elements are designed to join event streams or filter events and are composed of data windows and condition expressions. The data windows consist of "bounded set of events from an event stream" while condition expressions define the type of operation to be performed and the type of operand they must be performed on. The condition operators can either be comparison operators (equal, greater than, etc...), arithmetic operators (addition, subtraction etc...) or logical (and, not, or).

The "Pattern" component is defined as "a template specifying conditions which can match sets of related events". It is composed of at least one "PatternElement" which associate a pattern expression with a DataWindow element. The pattern expression is a set of particular conditions specific to pattern management including pattern timers. These components are not being shared by the generic rules. The implementation of these elements should be performed using additional knowledge according to Fig. 2.

Finally, the "Output" component specify the features of the complex event generated by the rules activation. It can be composed of several events each possessing properties and generated using expressions.

Further details about the elements of this model are available in Boubeta-Puig et al. (Boubeta-Puig et al., 2014).

5.2 Transformation Rules and Illustration

As mentioned above, the scope of shared concepts between the generic rule model and the EPL model does not include the event patterns as events are not defined in the source model. The transformation rules from the generic rule to EPL are described in Table 3.

The concept of Input Data in the source model can be easily assimilated to Window Elements as they correspond to the lowest level of monitoring data. However, the concept of generating variables using formulas is less simple. The solution proposed is to create an OutputElement for each variable, consequently considering a formula as an OutExpression. This OutExpression generates an Output property which is the value of the variable. These OutputElement can then be integrated into another EPL rule as a WindowElement. Here lies one of the purposes of using the generic rule model as it allows 1-to-n transformations from a single generic rule into several EPL rules. The complex conditions of the source model are a set of simple conditions aggregated together using logical operators. This structure is similar to the EPL concepts of SearchConditions as they are defined as a set ConditionElements related to each others by operators. However, in opposition with variables which are assimilated to WindowElements as they are dynamically generated, the constant operands are stable

Table 3: Transformation Rules from Generic Rules to EPL.

Generic Rule Model	EPL Model
Input Data	WindowElement
Formula	OutExpression
Variable	WindowElement/ OutputElement
Constant	ConditionOperand
Simple Condition	ConditionElement
Multiple Condition	SearchConditions
Comparison Operator	Operator
Logical Operator	Operator

and should then be considered as a part of the condition. For this reason, constant operands are mapped as ConditionOperands.

This mapping is illustrated by the two examples defined in the previous chapter. Considering the first example, the result of the transformation from generic rule (Fig.7) into EPL rules, using the mapping presented above, is represented in Fig. 11.

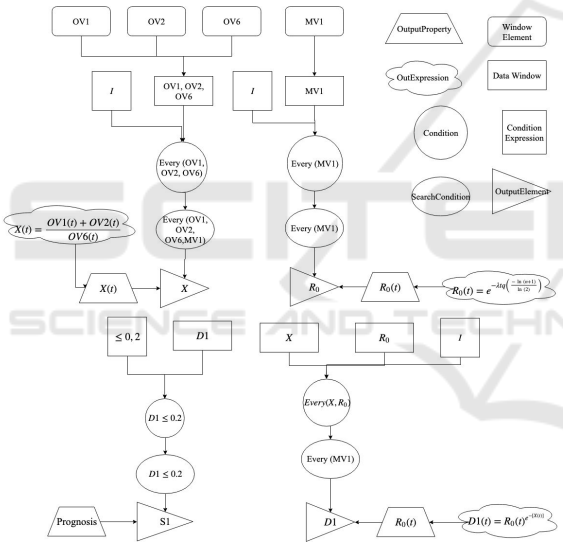

Figure 11: First Example of an EPL Model.

In this example, four OutputElements are genrated by four different EPL rules. The first rule creates an OutputElement X every time a set of WindowElements OV1, OV2 and OV8 is collected. The second example creates the OutputElement R_0 from the number of maintenance activities MV1. These OutpuElements are then aggregated in a third rule to produce the descriptor D1 as a third OutputElement. Finally, a fourth rule check the value of D1 using the ConditionElement "≤ 0.2" to produce or not an OutputEvent S1 with the OutputProperty "Prognosis", meaning that the symptom as been detected and indicating that a prognosis action has to be performed.

The result of the transformation for the second example is represented in Fig.12.

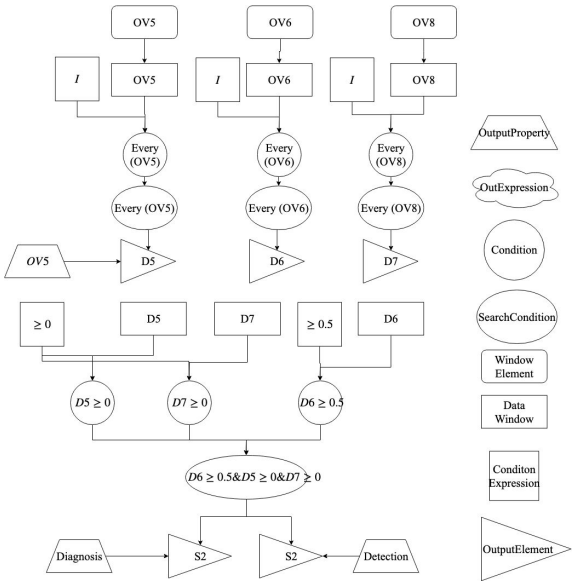

Figure 12: Second Example of an EPL Model.

In this example four rules are also required as three descriptors have to be generated before being tested and protentially producing an OutputElement for the symptom. The first rule produces the OutputElement D5 for each WindowElement OV5 received. The same process is used to create the OutputElement D6 from the WindowElement OV6 and to create the OutputElement D7 from the WindowElement OV8. These OutputElements are then used as WindowsElement for the fourth rule. In this rule, three ConditionElements check a DataWindow each. These ConditionElements are aggregated in a more complex SearchConditions element and produce two OutputElements with detection and diagnosis OutputProperties.

These examples show how the mapping presented in Table 3 can be used to produce EPL rules form a generic rule. A limit of this transformation may be the large number of event produced considering that each variable generation requires a rule to be defined and an event to be created. This limit may affect the CEP performance also the impact of this influence as yet to be measured.

6 PERSPECTIVES AND FUTURE WORK

This paper discusses about the interoperability of Expert Systems (ES) with Complex Event Processing (CEP) solutions. In the maintenance domain, the multiplication of the data sources have boosted the development of condition-based maintenance strate-

gies and given birth to new approaches such as Prognostics and Health Management (PHM). At the same time, data management has also become a challenge that can be neglected no more. As we believe that ES are relevant solutions for implementing PHM solutions due to their ability to capitalize and process available knowledge on a systems and its failures, we propose to combine this systems with CEP solutions. The purpose of using CEP is to filter to flow of input data in order to detect the relevant events for further analysis. In order to match these systems we propose to implement a model transformation approach to extract knowledge from the ES knowledge base and transform it into CEP rules. This transformation is divided in two steps. The first phase consists in transforming the relevant concepts of the knowledge base into generic rules. The second phase transforms these generic rules into CEP rules conforming to the Event Processing Language. The purpose of defining generic rules lies in the improved flexibility granted to the transformation and the possibility to perform 1-to-n transformations between these generic rules and EQL rules.

The limit of this approach is the current lack of computing implementation in a real case, which our future work will focus on.

REFERENCES

AFNOR (2018). NF EN 13306 - Maintenance — Terminologie de la maintenance.

Belaunde, M., Casanave, C., DSouza, D., Duddy, K., El Kaim, W., Kennedy, A., Frank, W., Frankel, D., Hauch, R., and Hendryx, S. (2003). *MDA Guide Version 1.0. 1*.

Bezivin, J. and Briot, J.-P. (2004). Sur les principes de base de l'ingénierie des modèles. *L'OBJET*, 10(4):145–157.

Bezivin, J., Büttner, F., Gogolla, M., Jouault, F., Kurtev, I., and Lindow, A. (2006). Model Transformations? Transformation Models! In Nierstrasz, O., Whittle, J., Harel, D., and Reggio, G., editors, *Model Driven Engineering Languages and Systems*, Lecture Notes in Computer Science, pages 440–453. Springer Berlin Heidelberg.

Bezivin, J. and Gerbe, O. (2001). Towards a precise definition of the OMG/MDA framework. In *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, pages 273–280.

Blanchard, B. S., Verma, D. C., and Peterson, E. L. (1995). *Maintainability : a key to effective serviceability and maintenance management*. New York : John Wiley & Sons, Inc.

Boubeta-Puig, J., Ortiz, G., and Medina-Bulo, I. (2014). A model-driven approach for facilitating user-friendly design of complex event patterns. *Expert Systems with Applications*, 41(2):445–456.

DePold, H. R. and Gass, F. D. (1999). The Application of Expert Systems and Neural Networks to Gas Turbine Prognostics and Diagnostics. *Journal of Engineering for Gas Turbines and Power*, 121(4):607–612.

Gertler, J. (1998). *Fault Detection and Diagnosis in Engineering Systems*. CRC Press. Google-Books-ID: fmPyTbbqKFIC.

Guillen, A. J., Crespo, A., Gómez, J. F., and Sanz, M. D. (2016). A framework for effective management of condition based maintenance programs in the context of industrial development of E-Maintenance strategies. *Computers in Industry*, 82:170–185.

ISO (2012). ISO 13372, Surveillance et diagnostic des machines — Vocabulaire.

ISO (2017). NF EN ISO 14224 - Petroleum, petrochemical and natural gas industries - Collection and exchange of reliability and maintenance data for equipment.

Jackson, P. (1998). *Introduction to Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition.

Jardine, A. K. S., Lin, D., and Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510.

Jin, X., Wah, B. W., Cheng, X., and Wang, Y. (2015). Significance and Challenges of Big Data Research. *Big Data Research*, 2(2):59–64.

Kalogirou, S. A. (2003). Artificial intelligence for the modeling and control of combustion processes: a review. *Progress in Energy and Combustion Science*, 29(6):515–566.

Lebold, M., Reichard, K., and Boylan, D. (2003). Utilizing dcom in an open system architecture framework for machinery monitoring and diagnostics. In *2003 IEEE Aerospace Conference Proceedings (Cat. No.03TH8652)*, volume 3, pages 3_1227–3_1236.

Lee, J., Jin, C., Liu, Z., and Ardakani, H. D. (2017). Introduction to data-driven methodologies for prognostics and health management. In *Probabilistic prognostics and health management of energy systems*, pages 9–32. Springer.

Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., and Siegel, D. (2014). Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 42(1):314–334.

Levine, P. and Pomerol, J.-C. (1990). *Systèmes interactifs d'aide à la décision et systèmes experts*. Hermès.

Liebowitz, J. (1995). Expert systems: A short introduction. *Engineering Fracture Mechanics*, 50(5):601–607.

Truptil, S., Bénaben, F., Salatge, N., Hanachi, C., Chapurlat, V., Pignon, J.-P., and Pingaud, H. (2010). Mediation Information System Engineering for Interoperability Support in Crisis Management. In Popplewell, K., Harding, J., Poler, R., and Chalmeta, R., editors, *Enterprise Interoperability IV*, pages 187–197. Springer London.

Vachtsevanos, G., Lewis, F., Roemer, M., Hess, A., and Wu, B. (2006). Systems Approach to CBM/PHM. In *Intel-*

ligent Fault Diagnosis and Prognosis for Engineering Systems, pages 13–55. John Wiley & Sons, Inc.

Vichare, N. M. and Pecht, M. G. (2006). Prognostics and health management of electronics. *IEEE Transactions on Components and Packaging Technologies*, 29(1):222–229.