Model Transformation from CBM to EPL Rules to Detect Failure Symptoms

Alexandre Sarazin, Sébastien Truptil, Aurelie Montarnal, Jérémy Bascans, Xavier Lorca

Model Transformation from CBM to EPL Rules to Detect Failure Symptoms

Alexandre Sarazin[1,2(✉)], Sebastien Truptil[3(✉)], Aurélie Montarnal[2(✉)], Jérémy Bascans[1(✉)], and Xavier Lorca[2(✉)]

[1] APSYS, 36 Rue Raymond Grimaud, 31700 Blagnac, France
alexandre-m.sarazin@mines-albi.fr, jeremy.bascans@apsys-airbus.com
[2] IMT Mines Albi, Centre de Génie Industriel, Allée des Sciences, 81000 Albi, France
{aurelie.montarnal,xavier.lorca}@mines-albi.fr
[3] CEA, CEA Tech Occitanie, 51 Rue de l'Innovation, 31670 Labège, France
sebastien.truptil@cea.fr

Abstract. The increasing complexity of modern systems, cost reduction policies and ever increasing safety requirements are bringing new challenges to the maintenance domain. In many fields, periodic maintenance actions become either insufficient or too expensive. In this context, Condition-Based Maintenance (CBM) strategies, and Prognostics and Health Management (PHM) in particular, are offering an interesting alternative by allowing systems to be maintained only when needed. These strategies rely on a constant monitoring and analysis of the systems operating conditions in order to detect and identify a failure when it occurs and even sometimes beforehand.

Nowadays, two main approaches are explored to detect failures in PHM solutions: one based on machine learning, the other based on expertise and capitalised system knowledge. This work proposes to combine a Complex Event Processing (CEP), to manage incoming data's volumetry and velocity, with an Expert System (ES) in charge of exploiting the capitalized knowledge. This paper focuses on the configuration of a CEP from rules contained in a CBM ES using a Model Driven Architecture (MDA). This configuration is a challenge, especially regarding the management of rules with temporal parameters and the need for intermediate results to deal with the rule's complexity.

Keywords: Maintenance · Knowledge base · Model transformation

1 Introduction

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" [1]. In particular, preventive maintenance describes maintenance action carried out to assess and/or to mitigate the degradation and reduce the probability of failure of an item. Condition-based maintenance (CBM) is a specific kind of preventive

maintenance assessing the systems physical conditions and analysing them to identify possible ensuing maintenance actions. Among the many solutions used for CBM data analysis, the two main strategies are data-driven and Expert Systems (ES).

Data-driven strategies, mostly consisting of machine learning algorithms, classify failures from past experience. The drawbacks of these approaches are the difficulty to explain the result and the large amount of reliable and relevant failure records required to train the learning algorithm. In fields like aeronautics where systems reliability is already strong, collecting a large amount of records of the same failure on identical systems is a very challenging task. However, the main advantage is the limited domain knowledge required to implement them.

ES approaches reproduce an expert reasoning by exploiting a base of facts and rules created from capitalised expert knowledge. However, according to [30], ES have "common defects in efficiency, scalability and applicability".

In order to solve the scalability issue, in particular from the data ingestion perspective, [25] proposed to use Complex Event Processing (CEP) to monitor and process the incoming data. The monitoring rules should be provided by the ES base of facts and transformed into generic rules and Event Processing Language (EPL) rules according to the Model Driven Architecture (MDA) methodology. However, the transformations proposed do not manage rules activated over a timeframe observation. For instance, a rule is activated if a condition is fulfilled continuously or repeats itself several times in a predefined timeframe. Yet, these kinds of rules can be written in EPL. In order to integrate these rules, this paper proposes an updated version of the transformations defined in [25].

In Sect. 2, the notions of PHM, ES and CEP will be defined and the motivation behind their combination will be explained. In Sect. 3, the CBM, generic rules and EPL metamodels will be presented. In the 4th section, the model transformations from these models will be detailed according to the MDA methodology. Finally, in Sect. 5, a representative case study is used to illustrate these transformations while stressing the need for considering timeframe based rules.

2 Use PHM Approach to Detect Failure Symptoms

2.1 Prognostics and Health Management

Modern maintenance is confronted to many challenges. Firstly, the systems become more and more complex which raises the difficulty in identifying and preventing failures. Secondly, the safety and availability requirements are getting increasingly demanding. As such, systems reliability is constantly being challenged. Moreover, cost reduction has become a strategic stake in many industrial fields and maintenance is not spared.

In this context, periodic maintenance, also known as time-based maintenance, is becoming insufficient as unnecessary actions become too expensive and unexpected failures affect availability and reliability [17]. In order to tackle these challenges, maintenance needs to be performed only when needed. Consequently, monitoring the systems working conditions in real time should be performed to detect and identify failures the moment they occur or, in best cases, beforehand. Prognostics and Health Management is a CBM strategy defined by [29] 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". It is composed of 7 main steps designed to collect, monitor and process sensor data in order to identify failures and estimate the Remaining Useful Life (RUL) of the defective system (Fig. 1). This information can then be processed by a decision support system and displayed to the end user [6, 22] .

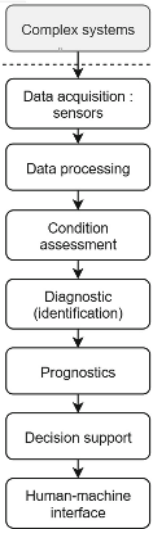

Fig. 1. PHM 7 steps [19].

The first step is to collect data in the system's actual life cycle conditions. This step is critical, as the relevance and quality of the collected data have a major impact on the quality of any further analysis. The second step is the data processing. It is meant to clean and transform collected data into more relevant variables before analysis. The third step is to assess the system's health status through anomaly detection based on the processed data. Next, potential failures and root causes should be identified. Depending on the identified failure, a degradation model should be chosen and applied to estimate the defective systems RUL in the prognostics phase. Finally, this information should be provided to a decision support system and displayed to the end user.

Although PHM architecture addresses modern maintenance issues, it also raises several technical challenges. In particular, ingesting and processing a large amount of incoming sensor data in an acceptable time is no simple task.

2.2 Expert System

In order to assess the system's health and diagnose failures, two main strategies can be adopted: data-driven or Expert Systems (ES). Data-driven technologies mostly rely on learning algorithms to detect anomalies and identify failures from a set of past records. However, this approach has benefits and drawbacks. This approach could detect maintenance needs even if the cause or the reason of the malfunction is unknown or not explainable. Nevertheless, the results obtained from these algorithms are hardly explainable and the confidence on the result depends on the number and the representativeness of the learning dataset. Depending on the observed system, collecting the learning dataset can be challenging, especially in fields like aeronautics where reliability is a major concern.

This work focuses on system without enough learning dataset but with available expert knowledge and safety documentation. To exploit this knowledge, we believe that an Expert System (ES) is interesting as explained in [9,12,17,21,22,28]. 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 [16]. ES are composed of a user interface, an inference engine and a knowledge base [23]. The user interface is meant to allow the user to interact with the system. The knowledge base structures a set of facts and rules describing the monitored system as well as the symptom and failures which can affect it. This component can be implemented using a static and dynamic database. The static database is meant to collect domain expert knowledge on the system. This database is stable, even though facts and rules may be added or modified. It should also be complete, consistent and accurate for the ES to perform acceptable analysis. The dynamic database, however, is used to "store all information obtained from the user, as well as intermediate conclusions (facts) that are inferred during the reasoning" [20]. Its content is lost at the end of each execution. An inference engine processes this knowledge base and reaches a conclusion. It can be used as a "control structure [...] that allows the expert to use search strategies to test different hypotheses to arrive at expert system conclusions" [23]. 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.

Although ES offers a good solution for processing maintenance data, this solution has scalability limits and can not easily process a large amount of incoming data [30]. Complex Event Processing (CEP) can be used to fill this gap.

2.3 Complex Event Processing

Managing the inflow of data is one of the main issues in implementing PHM, especially on complex systems. As an acceptable monitoring can only be performed when many sensors of different types are set on the system, the volume and velocity of these inputs are challenging to process. These processing issues share common characteristics with big data problems defined through the 5V [18]: volume of the collected data, velocity of its update, veracity of the information, variety of the sources and value of the information. To process the input data with reduced volume and velocity, a CEP can be used.

As described by [11], a CEP engine aims at processing data efficiently to immediately recognise patterns when they occur. It was first introduced by Luckham and Fransca [24] to process events at multiple levels of abstraction. It enables a system to reach passive context-awareness, however, unlike expert systems, it does not take decisions or recommendations. A CEP engine is based on a set of complex event processing rules. According to [8, 10], each rule enables to:

- Detect the occurrence of patterns based on presence or absence of linked events (e.g. incoming data)
- Filter events thanks to conditions;
- Generate new events, called complex events, based on incoming events. These complex events can be processed as new incoming events.

Depending on the language used to describe the complex event processing rules, the condition used to filter conditions could be simple (comparison to a threshold) or more complex with some functionality using temporal windows. Regarding the needs of the PHM approach, the use of temporal windows to detect abnormal situation is a requirement. Thus CEP using rules implementing Event Processing Languages (EPL) is a suitable option. EPLs are SQL-like languages designed support CEP solutions by defining events, conditions and patterns in order to detect interesting behaviors in the data [7]. ESPERTech[1] or Siddhi[2] are some examples of well-known EPL-based solutions.

In conclusion, to support PHM, ES and CEP are both useful and complementary. Indeed, CEP are designed to monitor large amounts of data in real time and detect patterns based on predefined rules whereas ES allow further analysis such as diagnostics or prognostics.

2.4 Using CEP and Expert System to Support PHM Approach

As detailed in [25], CEP can be combined with an ES in a PHM architecture. Indeed, as explained in previous sections, CEP and ES both have advantages and drawbacks. A CEP is designed to monitor large amounts of data in real time and detect patterns. Thus a CEP could reduce the flow of data but it could not

[1] http://www.espertech.com/.
[2] https://docs.wso2.com/display/CEP300/Siddhi+Language+Specification.

perform a diagnosis. An ES analyses data (to provide diagnostics or prognostics) and explains the analysis result but the processing could be slow. Therefore, the combination of the two technologies is relevant in a PHM application. To combine the two technologies, the anomaly detection rules implemented in the ES knowledge base should be transformed into EPL rules to be applied by the CEP according to Figure 2.

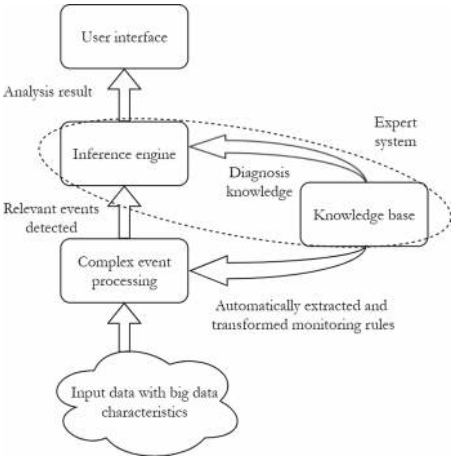

Fig. 2. Relation between CEP and expert system adapted from [25].

In this architecture, incoming sensor data with high velocity and volume should be injected in the CEP. Used as a filter, the CEP then detects the relevant anomalies based on rules extracted from the ES knowledge base. The detected events can then be processed by an inference engine to identify the related failure. This information can then be displayed to the end-user through an interface.

The main issue in combining CEP and ES, is to transform CBM rules into EPL. Moreover, according to [7], one of the downsides of CEP 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 meta model for EPL and an automatic model-to-code solution have been designed to implement the rules in commercial solutions. The EPL metamodel proposed by [7] is detailed in Fig.3.

This EPL metamodel is composed of four main types of components: the "SearchConditions","Pattern", "Output" and "Link" elements.

To illustrate these types of components, the following simple examples are used: if x > y then z and if Mean(k,l,m) > y then z.

The "Link" elements are designed to connect the elements of the three other components. It is divided into operands (as x, k, l, m, y and also Mean(k,l,m)) and operators (as >). An operator is an operation performed on one or several operands. Operators can either be Unary, Binary or N-ary depending on the number of operands they can be applied on.

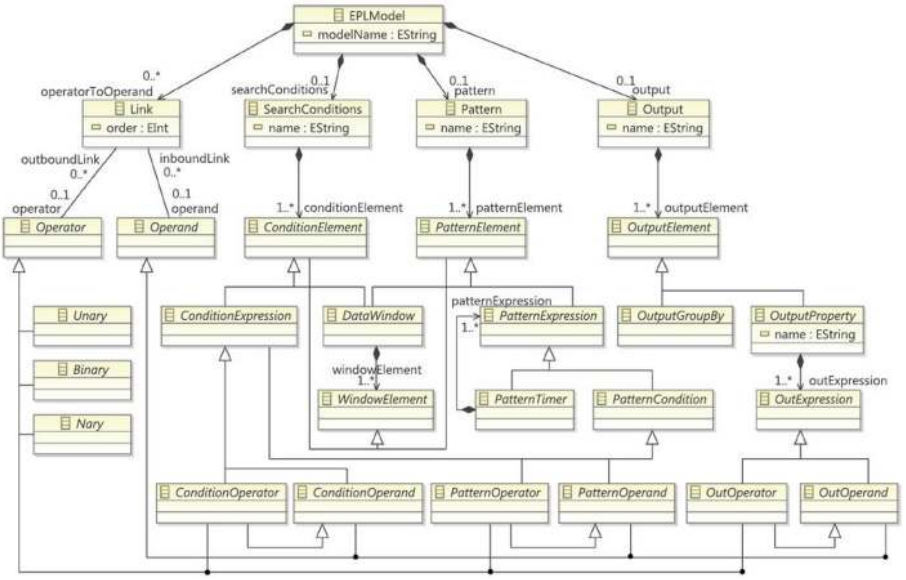

Fig. 3. EPL metamodel [7].

The "SearchConditions" component is a collection of "ConditionsElement". These elements are rules composed of "ConditionExpression" elements (as x > y and Mean(k,l,m) > y). These "ConditionExpression" elements are built from"DataWindow" (as x, y, Mean(k,l,m)), acting as operands, compared with operators. "DataWindow" are transformations of "WindowElement" (as k, l, m and x) which are input data.

The "Pattern" component is defined as "a template specifying conditions which can match sets of related events". It is used to manage sets of events occurring in a timeframe. For instance, a "PatternCondition" can count the occurrence of a specific type of event while the "PatternTimer" specifies the length of a monitoring timeframe. All these elements can be considered in the transformation process and thus leads to extend the work of Sarazin et al. [25].

Finally, the "Output" (as z) component specifies 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. [7].

3 Proposed Model Driven Architecture

The previous sections aim to argue that the use of CEP combined to ES is relevant to provide a PHM architecture. The combination of the two technologies is based on conditions used to detect relevant anomalies. Indeed, CEP are used

to reduce the incoming flow of data to the ES. Therefore, the configuration of the CEP essentially depends on the ES content. This paper's proposal is to automatically configure the CEP from the ES using a model driven architecture (MDA) [26]. This section presents first the concepts as Model, Metamodel and Model Transformation, before proposing a model transformation from CBM to CEP rules based on an MDA methodology.

3.1 Model, Metamodel, Model Transformation

According to [2], a model is a "formal specification of the function, structure and/or behavior of an application or system". It should also be noticed that a single system can be represented by many different models depending on the point of view adopted [4]. The rules used to create and structure a model are defined using a metamodel, which is an "explicit specification of an abstraction" [3]. It defines the concepts manipulated in a model and the relations between them. A model can thus be considered as an instance of a metamodel and all models must conform to their own metamodels, conforming themselves to a metametamodel [5].

In order to differentiate the business concepts manipulated in a system from the technological platform used to implement them, a model-based methodology named Model Driven Architecture (MDA) has been defined [26]. This methodology can be used in software development to separate the design steps based on the formalisation of business logic from the the technical implementation steps. According to the MDA guide, this methodology improves the "portability, interoperability and reusability" of the final result. The MDA methodology relies on four main types of models [2, 26]:

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

The CIM is designed from the formalisation of the business logic by a domain expert with its own vocabulary. According to the MDA methodology, once the CIM is defined, the PIM can be designed to include a first level of specification. The PIM purpose is to make it possible to adapt the CIM to different platforms of the same kind. The PM describes the platform used to implement the model. It describes the concepts manipulated in the platform and the structure binding these concepts together. Finally, the PSM is defined as a "view of a system from the platform specific viewpoint". It is the implementation of the PIM on the platform modeled by the PM. [2].

MDA methodology consists in transforming a CIM to a PIM and a PIM to a PSM. This process is called model transformation [2]. It 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" [5]. In a model transformation, all concepts may not be commonly shared by the source and target models.

Consequently, the first step of a model transformation is to identify the shared concepts in each model, which correspond to the transformation domain, and the specific concepts which are not shared (Fig. 4). In the transformation domain, the concepts are then converted from the source to the target model using transformation/mapping rules. The specific part of the source model can be considered as capitalized knowledge while the specific concepts of the target model are additional knowledge that should be implemented from external sources [27].

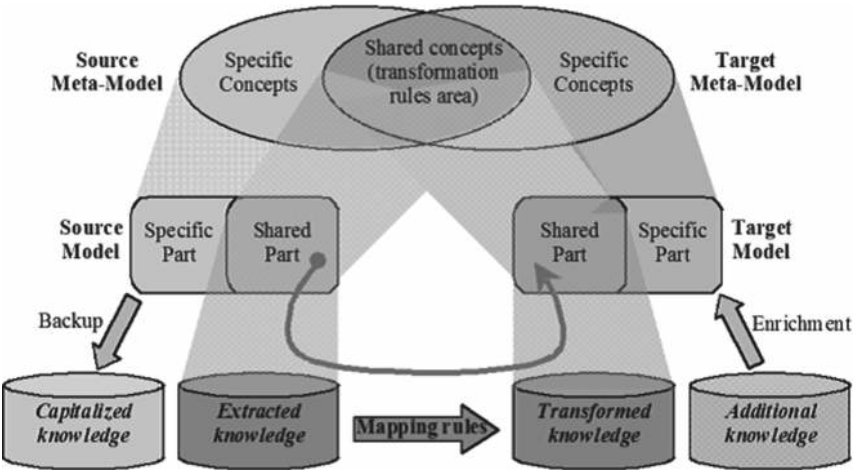

Fig. 4. Model transformation principle [27].

3.2 Model-Driven Architecture from CBM to EPL Rules

To transform CBM rules into CEP rules, an MDA can be implemented. According to the MDA philosophy, the transformation should be performed in two steps: the first step is to transform the CBM model into generic rules and the second step should convert these generic rules into CEP rules. Should alternatives to modern CEP emerge, the generic rules are designed to improve the transformation's adaptability towards new solutions. Consequently, CBM and generic rules refer to CIMs because the rules contain no platform specification even though their respective structures are different. However, EPL rules can be assimilated as a PIM. In [7], a model to code transformation has been presented to convert EPL rules into more specific languages like EQL, CQL, SteamSQL or CCL. These languages could be considered as PMs and the rules implementations in these languages would be PSMs.

In this paper, a metamodel for generic rules will be defined and the transformation rules from CBM to generic rules and from generic rules to EPL will be detailed (Fig. 5).

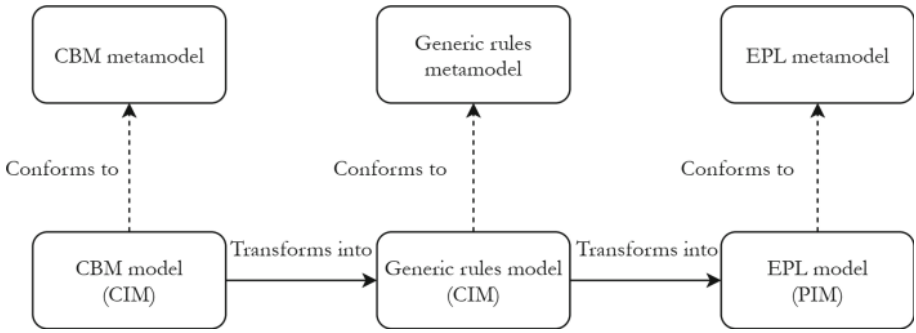

Fig. 5. Model transformation from a CBM to EPL rules [25].

The first transformation should convert CBM rules into generic rules. As such, a metamodel should be used to define and structure the concepts manipulated in a CBM rule. The metamodel chosen has been designed by [13] with concepts defined from ISO standards. This metamodel describes the different parts of a CBM solution (Fig. 6). According to 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 is composed of all information related to the systems structure. 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" [15].

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

The "Information Sources" block collects all the elements related to information gathering. This block contains monitoring variables generated from sensor data, variables, and measurement techniques. These monitoring variables are meant to be used by the "Symptom Analysis" block for anomaly detection purpose.

The "Symptom Analysis" block defines the descriptors, symptoms and information rules. A descriptor is a "feature, data item derived from raw or processed parameters or external observation" [14]. A descriptor is produced by processing 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" [14]. 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" [13].

Finally, the "Maintenance Decision-Making" block, regroups several CBM processes divided into three types: anomaly detection, diagnosis and prognosis activities. The detection element is used to calculate the descriptors values and spot abnormal behaviors using interpretation rules. It is a "conclusion or group of conclusions drawn about a system or unit under test" [14]. When an anomaly is detected, it can trigger a diagnosis process in order to identify the related failure and the responsible maintainable item. When an anomaly or a diagnostic process is performed, it can also trigger a prognostics process to estimate the Remaining Useful Life of a maintainable item. The results of these processes are collected by a maintenance decision process meant to help the end-user.

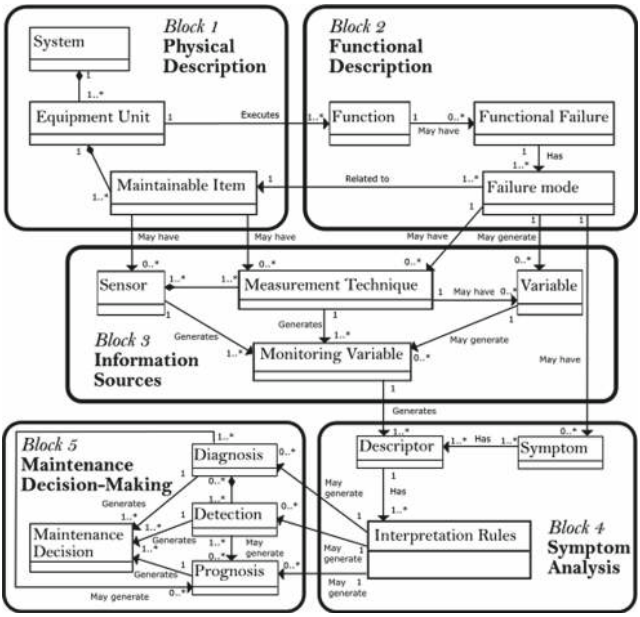

Fig. 6. Basic structure for the CBM solution [13].

4 Model Transformation

Once the motivations to combine ES and CEP in a PHM architecture have been explained, the model transformation from CBM to EPL should be detailed. This transformation is performed in two steps: the CBM rules are first converted into generic rules before being transformed into EPL. In this chapter, the metamodel for generic rules will first be detailed, then the mapping for the first and second transformations will be described.

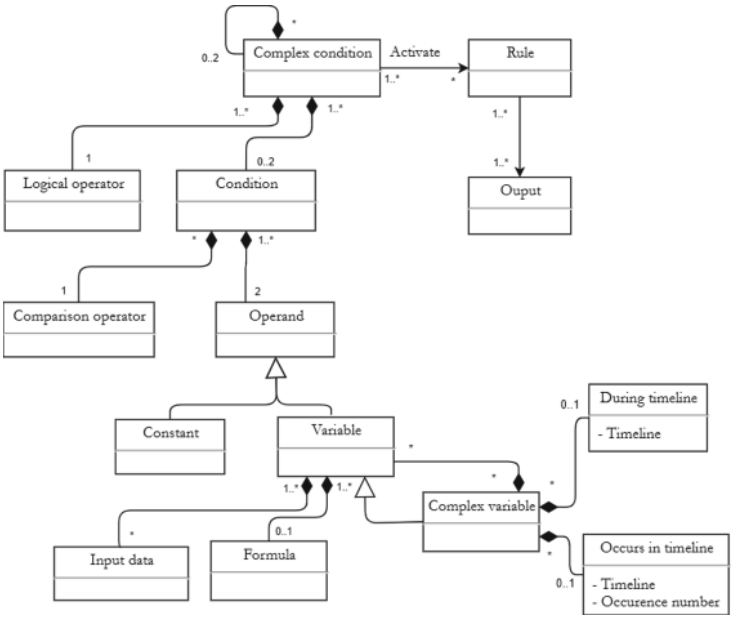

Fig. 7. Generic rules metamodel.

4.1 Generic Rules Metamodel

The first model transformation converts CBM rules into generic rules. As the metamodel for CBM has been previously presented, the generic rules metamodel should now be detailed (Fig. 7). The purpose of designing these generic rules is to improve the transformation adaptability should an alternative to EPL emerge. This metamodel should define a rule general structure while staying at a conceptual level.

According to this metamodel, input data can be transformed into a variable using a formula. This variable can then generate a complex variable which can be created from several variables using a formula and/or by applying timeframe operators. Two timeframe operators are presented in this metamodel: "during timeline" which counts the duration of an event are situation in a predefined timeline and "occurs in timeline" which counts the number of occurrences of an event in a given timeline. These operators and their aggregation in a complex variable can be considered a major modification of the version presented in [25] because it allows the event's chronology to be considered when defining rules activation conditions. The generated complex variable can then be used as a variable. Indeed, a complex variable is a specific kind of variable, which is usually more expensive to create in terms of duration or calculation.

A condition is a comparison operator (e.g. $\leq, \geq, =$) applied to two operands, which can be either a constant value or a variable. A complex condition is an aggregation of two simple or complex conditions related by a logical operator (e.g. AND,

OR). For instance $(a \geq b)OR(c \leq d)$ is a complex condition composed of two conditions $(a \geq b)$ and $(c \leq d)$ related by the logical operator "OR". The highest level complex condition activates a rule which generates an output. This output can potentially be used as an input for an another rule and inserted as a variable. Consequently, a business rule could be converted into several generic rules.

Once the generic rules metamodel has been defined, the mapping rules with the CBM metamodel presented previously can be detailed.

4.2 From CBM Knowledge Base to Generic Rules

As the source and target models for the first transformation, respectively the CBM and generic rules metamodels, have been defined, the mapping rules can be detailed. The first step in performing model transformation is to identify the shared concepts. In the source model, these concepts are:

- the sensor and variables which can be considered as inputs
- measurement techniques as a first input processing
- the monitoring variables which result from input data transformation
- 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 target model concepts, mapping rules should be applied according to Table 1.

Table 1. Transformation rules from a CBM model into generic rules.

CBM model concept	Generic rule model concept
Sensor	Input data
variable	Input data
Measurement technique	Formula
Descriptor	During dimeline
	Occurs in timeline
	Formula
	Variable
	Complex variable
Monitoring variable	Variable
Interpretation rules	Constant
	Comparison operator
	Logical operator
	Condition
	Complex condition
	Rule
	Output
Symptom	Rule
Diagnosis, detection, prognosis	Output

According to this mapping, the source model's sensor and variable concepts can be matched as input data in the target model. These input data are then transformed into a target model variable using a formula. The process is similar to the generation of the source model's monitoring variable from measurement techniques applied to a sensor or variable. As such measurement techniques are matched with a formula and monitoring variable as a target model's variable.

A descriptor is processed from the transformation of one or several monitoring variables using a formula and/or timeline operators. Consequently, a descriptor is mapped to several target model's concepts. Depending on the descriptor's content, formula and variable instances can be generated or even complex variable with "during timeline" and/or "occurs in timeline" instances. Interpretation rules define the conditions applied to a descriptor to trigger a maintenance action.

Similar to descriptors, interpretation rules can be mapped to several target model's concepts depending on their content. Indeed, interpretation rules can be matched with complex conditions and generate the conditions, logical and comparison operators they are composed of. Should an interpretation rule be too complex, an output can be generated to be used as variable in a new rule. One such example will be presented in chapter 5. The symptom component provides business logic on the state of the system depending on the interpretation rule's activation. It can thus be mapped to the target model's rule. The maintenance actions can be triggered by the rule activation and can thus be matched as a rule output.

Once the generic rules are designed, the second transformation into EPL rules can be performed.

4.3 From Generic Rules to EPL

This section aims at presenting the mapping rules, available in Table 2, from generic rules to EPL. In this table, the generic rule model concepts of Table 1 have been factorised to simplify the connection with the EPL model concepts. As a reminder, CEP is designed to monitor large amounts of data in real time and detect patterns thanks to rules which respect the EPL Metamodel. To achieve this purpose, a CEP receives incoming data, named "WindowElement", which has to be combined with others to create a "DataElement". The combination of incoming data could be based on a "PatternExpression" such as the number of occurrences during a timeframe or others aggregation operators as well as the identity function. Once a "DataElement" is generated, a CEP aims to identify a desired pattern referred to as a "SearchCondition" that is a combination of "ConditionElement". More extensive details have been presented in Sect. 2.4.

According to these mapping rules, the source model's input data corresponds to the target model's WindowElement. A WindowElement can be transformed by a PatternExpression, which refers to a formula, in order to provide a DataElement that refers to a variable. Regarding the complex variables, it is obvious that they refer to DataWindow because they are generated based on a combination of variables thanks to aggregation operators. This means that a variable could

Table 2. Transformation rules from generic rules to EPL.

Generic rule model concept	EPL model concept
Input data	WindowElement
Variable	DataWindow or WindowElement
Formula	PatternExpression
Occurs in timeline	PatternCondition
During timeline	PatternTimer
Complex Variable	DataWindow
Constant	Operand
Comparison operator	Operator
Logical operator	Operator
Condition	ConditionElement
Complex condition	SearchConditions
Rule	EPLModel
Output	OutputElement

be a WindowElement. Thus depending on the rule, variables could refers to DataWindow or WindowElement.

The "Occurs in timeline" operator is a pattern condition while the "During timeline" operator is a PatternTimer. A constant is matched as an operand, while the logical and comparison operators are matched as operators. A condition, composed of operands and operators, corresponds to a ConditionElement. A complex condition, composed of several conditions can be assimilated to a SearchCondition element. Finally, the rule, activated by a complex condition and generating a output can be translated as an EPLModel and the source model's output as an OutputElement. In addition, the OutputElement could be used as a new incoming event by CEP as explained in Sect. 2.3. Thus, the generated output element may then be integrated as a WindowElement of another rule.

These rules allow generic rules to be transformed into EPL rules. In the next chapter, examples of such transformations will be presented.

5 Case Study/Illustration

Previously, the motivations behind combining CEP and ES in a PHM solution have been explained. The metamodels for CBM, generic and EPL rules have been presented and the mapping rules have been specified according to the MDA methodology. In this section, a realistic case study with two CBM rules will illustrate how these transformations should be applied. Events chronology management will be displayed to demonstrate the value of these transformations extended from the work of [25].

5.1 System Description

The proposed case study consists in detecting abnormal situations on a system in charge of regulating the airflow in a room. To ensure this functionality the considered system is composed of two actuators A1 and A2 which have to open or close a "panel". A2 can be considered as A1 backup. As such they should be opened and closed at the same time. An abnormal situation is referring to (1) an abnormal opening or closing of the panel and or (2) to an overpressure in the room. In order to detect these abnormal situations, data are gathered by sensors and sent to the PHM architecture. This data is:

- two Boolean variables are used to indicate the actuators position:
  - FO: equals 0 if the actuator is not opened and 1 if opened
  - FC: equals 0 if the actuator is not closed and 1 if closed
- P: the pressure inside the room.

Therefore, the following input data are available to detect abnormal situations: A1_FO, A1_FC, A2_FO, A2_FC and P. Based on these data, the following rules could be used to detect the two kind of abnormal situations:

1. **One of the Two Actuators Has Failed:** This situation occurs when A1 and A2 are in different positions or it can be due to a loss of signal which implies that the Boolean value has not been updated. These abnormal situations could be identified thanks to the following logical expression:

$$\text{If } (A1\_FO \neq A2\_FO) \text{ AND } (A1\_FC \neq A2\_FC)$$

2. **Risk of Overpressure inside the Room Increases:** When the pressure inside the room increases by 0.14psid in a 500 ms timeframe during at least 1s or if this same increase occurs 3 times in 10 s. Figure 8 illustrates the detection of over-pressure in the room. This figure simulates incoming P values each

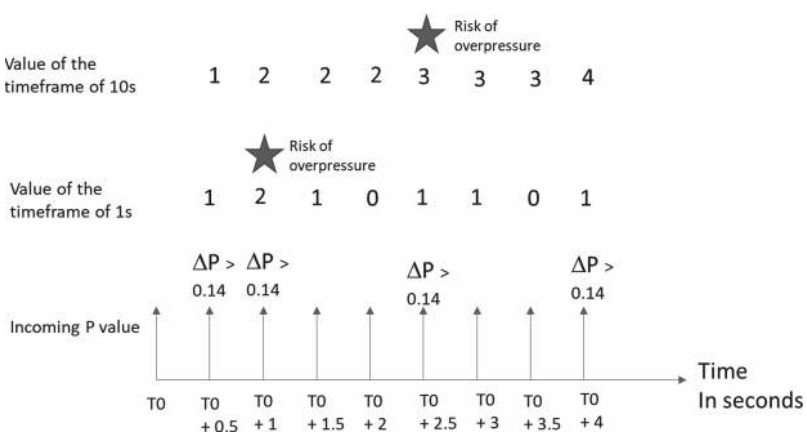

Fig. 8. Illustration of the detection of overpressure in the room.

500ms and the value of the two timeframes (1s and 10s) based on the value of the pressure difference ΔP. If the value of ΔP is over 0.14psi, then each timeframe increases by 1 and if the value of the timeframe is over the threshold, then an alert of risk of over-pressure has to be identified. This figure illustrates also that the timeframe value could decrease if $\Delta P \leq 0.14psi$.

This case study is relevant because the first rule illustrates the transposition of a rule with several monitored variables in the generic rule metamodel and the EPL metamodel whereas the second rule illustrates how to interpret the time windows in the generic and EPL metmamodel.

5.2 Examples of CBM Models

The models of the two previously presented rules are respectively detailed in Fig. 9.

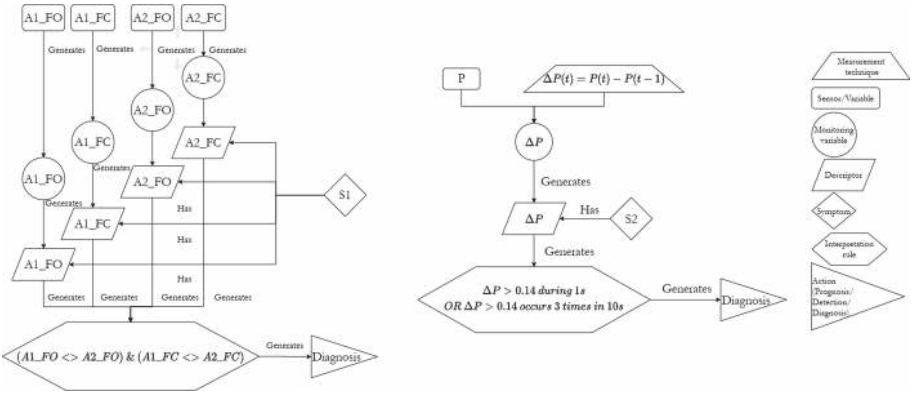

Fig. 9. CBM model for the two rules.

In the first example, A1_FO, A2_FO, A1_FC and A2_FC are being monitored and compared in the interpretation rule $(A1\_FO <> A2\_FO)\, AND\, (A1\_FC <> A2\_FC)$. The related symptom is called "S1" and triggers a diagnosis action when the interpretation rule is activated. This example is meant to detail how to manage several inputs in the transformation process.

In the second example, only the pressure is being monitored. The data generated form the sensor is transformed by the measurement technique Δ into a monitoring variable ΔP. This monitoring variable generates an identical ΔP descriptor which is used in the interpretation rule $(\Delta P > 0.14\ during\ 1s)\ OR\ (\Delta P > 0.14\ occurs\ 3\ times\ in\ 10s)$. The symptom related to this rule is "S2" and triggers a diagnosis action. This example is meant to explain how the "during" and "occurs in timeline" operations are managed in the transformation process.

5.3 Generic Rules Examples

This section focuses on the transformation from the CBM level to the generic rules level of the two rules presented in Sect. 5.1. For each example, the mapping rules will be applied to the source model before presenting the resulting target model.

Regarding the first rule, used to detect that one of the two actuators has failed, the mapping between the CBM model elements and the generic rule model elements is detailed in Table 3. The resulting model is represented in Fig. 10. In this example, it should be noticed that the source model's interpretation rules are transformed into several elements in the target Model. Indeed, the interpretation rule corresponds to a Complex Condition composed of two conditions linked by

Table 3. Application of transformation rules from CBM to generic rules for the first rule.

Source model concept	Source element	Target element	Target model concept
Monitoring variable	A1_FO	A1_FO	Variable
Monitoring variable	A1_FC	A1_FC	Variable
Monitoring variable	A2_FO	A2_FO	Variable
Monitoring variable	A2_FC	A2_FC	Variable
Interpretation rules	$A1\_FO \neq A2\_FO$ and $A1\_FC \neq A2\_FC$	&	Logical operator
		$\neq$	Comparison operator
		$A1\_FO \neq A2\_FO$	Condition (C1)
		$A1\_FC \neq A2\_FC$	Condition (C2)
		C1 & C2	Complex condition
Prognosis	One of the two actuators has failed	One of the two actuators has failed	Output

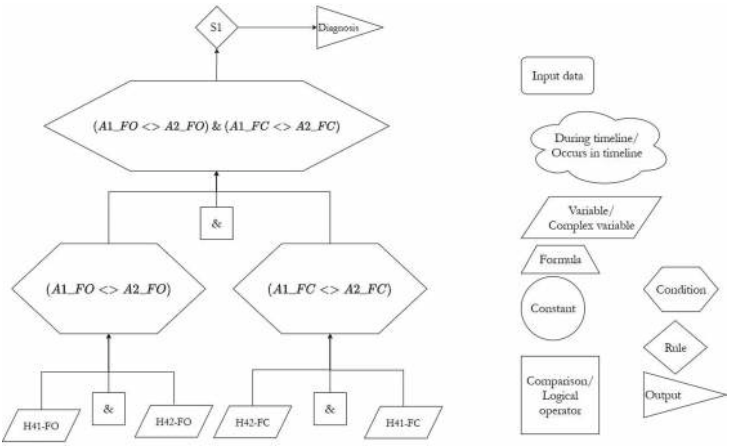

Fig. 10. Generic rule model of the first example.

Table 4. Application of Transformation Rules from CBM to Generic Rules for the second rule (model a).

Source model concept	Source element	Target element	Target model concept
Variable	P	P	Input data
Measurement technique	Δ	Δ	Formula
Monitoring variable	ΔP	ΔP	Variable
Interpretation rules	$\Delta P > 0.14$ during 1 s or $\Delta P > 0.14$ during 3 times in 10 s	0.14	Constant
		$>$	Comparison operator
		$\Delta P > 0.14$	Condition
		$\Delta P > 0.14$	Complex variable

Table 5. Application of transformation rules from CBM to generic rules for the second rule (model b).

Source model concept	Source element	Target element	Target model concept
Interpretation rules	$\Delta P > 0.14$ during 1 s or $\Delta P > 0.14$ occurs 3 times in 10 s	$\Delta P > 0.14$	Complex Variable
		or	Logical operator
		1 s	During timeline
		3 times in 10 s	Occurs in timeline
		$\Delta P > 0.14$ during 1 s	Condition (C1)
		$\Delta P > 0.14$ occurs 3 times in 10 s	Condition (C2)
		C1 or C2	Complex condition
Diagnosis	Risk of over-pressure	Risk of over-pressure	Output

logical operator. Each condition is composed of two Variables and a Comparison Operator.

Regarding the second rule, which aims to detect a risk of overpressure inside the room, this rule is based on occurrences number of pressure variation above a threshold. If the number of occurrences is greater or equal to 2 in one second or 3 in ten seconds then the risk has to be detected. In this kind of rules, it is necessary to define a complex variable and split the CBM rule into two generic rules at the CIM level in two rules at the PIM level. The first generic rule should generate a complex variable when the pressure variation is above 0.14 psi. The second generic rule should monitor the number of these occurrences over a timeframe to detect the overpressure risk. Table 4 details the mapping from the source model's elements to the target model's element whereas the left part of Fig. 11 corresponds to the part of the target model generating a complex variable.

Once the complex variable $\Delta P > 0.14$ psi is generated, the next part of the second rule could be transformed. Table 5 shows the mapping from the source model elements to the target model elements illustrated by the right part of Fig. 11.

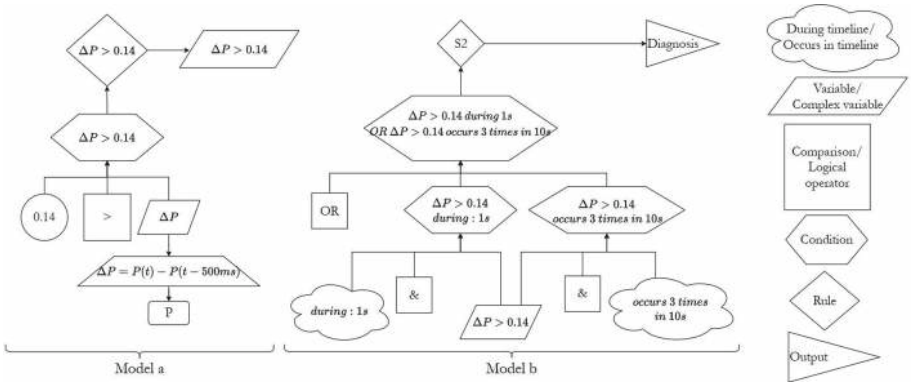

Fig. 11. Generic rule models describing the second example.

5.4 Examples of EPL Models

This section focuses on the transformation from generic to EPL rules of the models presented in Sect. 5.3. These transformations are based on the mapping rules detailed in Sect. 4.3.

A generic rule model of the first rule, referring to an abnormal situation caused by the failure of one of the two actuators, has been described in the previous section. It can now be transformed into an EPL model based on the mapping rules presented in Table 6. The resulting model is detailed in Fig. 12.

Regarding the second Rule, referring to a risk of overpressure in the room, two generic rule models have to be transformed. The generic model which generates the complex variable ΔP >0.14 psi can be converted in the EPL Model presented in the left part of Fig 13 based on the mapping rules presented in Table 7.

Finally, the model of the second rule detecting a risk of over-pressure in the room is transformed in the EPL model represented in the right part of Fig. 13 according to the mapping rules listed in Table 8.

Table 6. Application of transformation rules from generic rule to EPL for the first rule.

Source model concept	Source element	Target element	Target model concept
Variable	A1_FO	A1_FO	WindowElement
Variable	A1_FC	A1_FC	WindowElement
Variable	A2_FO	A2_FO	WindowElement
Variable	A2_FC	A2_FC	WindowElement
Logical operator	&	&	Operator
Condition (C1)	$A1\_FO \neq A2\_FO$	SearchCondition	
Condition (C2)	$A1\_FC \neq A2\_FC$	SearchCondition	
Complex condition	C1 & C2	C1 & C2	SearchElement
Output	One of the two actuators has failed	One of the two actuators has failed	Output

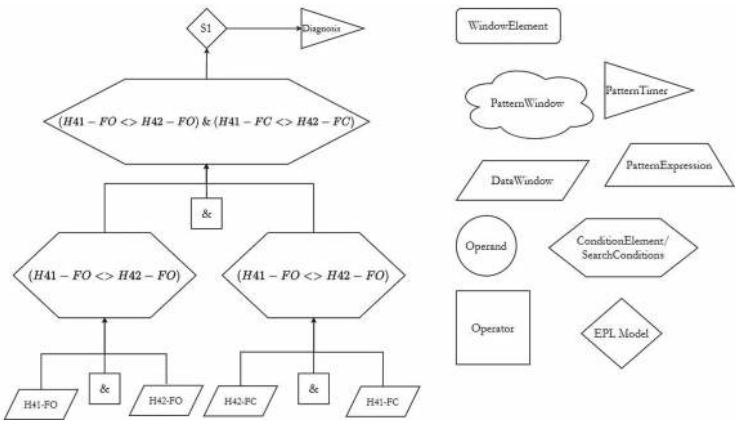

Fig. 12. EPL rule model for the first example.

Table 7. Application of transformation rules from generic rule to EPL for the second rule.

Source model concept	Source element	Target element	Target model concept
Input data	P	P	WindowElement
Formula	Δ	Δ	PatternExpression
Variable	ΔP	ΔP	DataWindow
Condition	$\Delta P > 0.14$	$\Delta P > 0.14$	SearchCondition
ComplexVariable	$\Delta P > 0.14$	$\Delta P > 0.14$	OutputElement

Table 8. Application of transformation rules from generic rule to EPL for the second rule.

Source model concept	Source element	Target element	Target model concept
Complex variable	$\Delta P > 0.14$	$\Delta P > 0.14$	WindowElement
Logical operator	OR	OR	Operator
During timeline	1 s	during: 1 s	PatternTimer
Occurs in timeline	3 times in 10 s	occurs in 10 s	PatternCondition
Condition (C1)	$\Delta P > 0.14$ during 1 s	$\Delta P > 0.14$ during 1 s	ConditionElement
Condition (C2)	$\Delta P > 0.14$ occurs 3 times in 10 s	$\Delta P > 0.14$ occurs 3 times	ConditionElement
Complex condition	C1 or C2	C1 or C2	SearchConditions
Output	Risk of over-pressure	Risk of over-pressure	Output

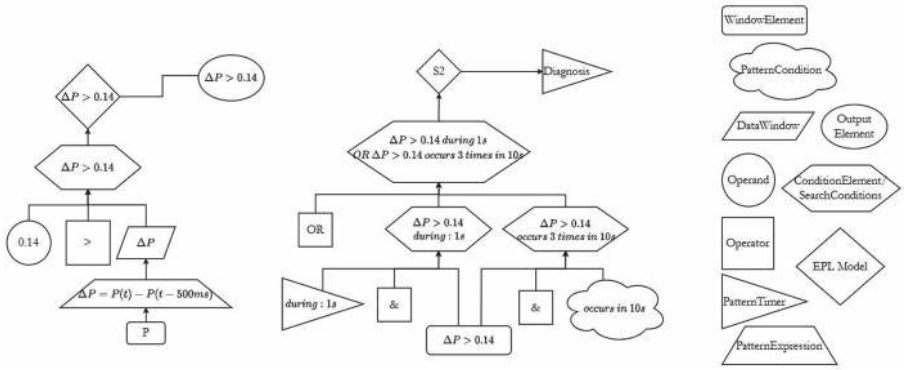

Fig. 13. EPL rule models describing the second example.

6 Summary and Future Work

In the maintenance domain, the multiplication of data sources have boosted the development of condition-based maintenance (CBM) strategies and given birth to new approaches such as Prognostics and Health Management (PHM).

Nowadays, two main approaches are explored to detect failures in PHM solutions: one based on machine learning, the other based on expertise and general domain knowledge. This work focuses on the solutions based on expertise and capitalised knowledge and thus focuses on systems with few records related to failures. In such context, the use of Expert Systems (ES), in charge of exploiting the capitalized knowledge, is relevant. Moreover, unlike machine learning approaches, an ES is able to explain the need for maintenance actions. This aspect is a key stone for decision in the maintenance domain. However, current ES have scalability limits regarding the multiplication of data sources and especially the volumetry and velocity of the incoming data. Therefore, this paper

proposes to combine the ES with Complex Event Processing (CEP) to tackle these limits. Indeed, a CEP aims at processing data efficiently to immediately recognise patterns when they occur. Therefore, a CEP can be used in order to filter the incoming data and only requests the ES when it is relevant, in other words a CEP aims to reduce the volumetry and the velocity of the incoming data for the ES.

Even if the idea to combine ES with CEP is promising, it requires that the configuration of the CEP, especially the rules, are always in line with the needs of ES. This requirement implies that the configuration of the CEP has to be automatically generated from the ES. This paper details the proposed Model Driven Architecture (MDA) used to generate the CEP configuration from the ES. This MDA consists in transforming, first, CBM models into generic rules models before transforming these generic rules models into EPL models. Then code, as EQL rules for example, can be generated from these EPL models. This paper focuses on the first two transformations and the need to pass through the generic rules level due to the CBM rule's complexity such as the use of temporal parameters or the need for intermediate results. Due to this complexity, the transformation of descriptors and interpretation rules concepts of the CBM can provide lots of different generic rules concepts.

However, the presented work has limitations which have to be addressed. One of the main challenges in this model transformation is to automatically generate descriptors from documentation. To deal with this issue, exploiting resources in text format may be very helpful. To perform this, using complementary approaches might be necessary to extract and decompose descriptors into generic rules model instances. Consequently, this transformation could be improved by the use of Natural Language Processing (NLP), for example the use of entity named recognition algorithms. In addition, this proposal has, for now, been tested on case studies such as presented here, however the short-term planned perspective is now to implement this architecture on a real complex system.

References

1. AFNOR: NF EN 13306 - Maintenance – Terminologie de la maintenance, January 2018
2. Belaunde, M., et al.: MDA guide version 1.0. 1 (2003)
3. Bezivin, J., Gerbe, O.: Towards a precise definition of the OMG/MDA framework. In: Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001), pp. 273–280, November 2001. https://doi.org/10.1109/ASE.2001.989813
4. Bezivin, J., Briot, J.P.: Sur les principes de base de l'ingénierie des modèles. L'OBJET **10**(4), 145–157 (2004)
5. Bézivin, J., Büttner, F., Gogolla, M., Jouault, F., Kurtev, I., Lindow, A.: Model transformations? Transformation models!. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MODELS 2006. LNCS, vol. 4199, pp. 440–453. Springer, Heidelberg (2006). https://doi.org/10.1007/11880240_31

6. Blanchard, B.S., Verma, D.C., Peterson, E.L.: Maintainability: A Key to Effective Serviceability and Maintenance Management. Wiley, New York (1995). https://trove.nla.gov.au/work/30017742
7. Boubeta-Puig, J., Ortiz, G., Medina-Bulo, I.: A model-driven approach for facilitating user-friendly design of complex event patterns. Expert Syst. Appl. **41**(2), 445–456 (2014). https://doi.org/10.1016/j.eswa.2013.07.070, http://www.sciencedirect.com/science/article/pii/S0957417413005575
8. Cugola, G., Margara, A.: Processing flows of information: from data stream to complex event processing. ACM Comput. Surv. (CSUR) **44**(3), 1–62 (2012)
9. DePold, H.R., Gass, F.D.: The application of expert systems and neural networks to gas turbine prognostics and diagnostics. J. Eng. Gas Turbines Power **121**(4), 607–612 (1999). https://doi.org/10.1115/1.2818515
10. Etzion, O., Niblett, P., Luckham, D.: Event processing in action. Manning Greenwich (2011)
11. Flouris, I., Giatrakos, N., Deligiannakis, A., Garofalakis, M., Kamp, M., Mock, M.: Issues in complex event processing: status and prospects in the big data era. J. Syst. Softw. **127**, 217–236 (2017)
12. Gertler, J.: Fault Detection and Diagnosis in Engineering Systems. CRC Press (1998). Google-Books-ID: fmPyTbbqKFIC
13. Guillen, A.J., Crespo, A., Gómez, J.F., Sanz, M.D.: A framework for effective management of condition based maintenance programs in the context of industrial development of E-Maintenance strategies. Comput. Industry **82**, 170–185 (2016). https://doi.org/10.1016/j.compind.2016.07.003, http://www.sciencedirect.com/science/article/pii/S0166361516301178
14. ISO: ISO 13372, Surveillance et diagnostic des machines – Vocabulaire, June 2012
15. ISO: NF EN ISO 14224 - Petroleum, petrochemical and natural gas industries - Collection and exchange of reliability and maintenance data for equipment, October 2017
16. Jackson, P.: Introduction to Expert Systems, 3rd edn. Addison-Wesley Longman Publishing Co. Inc., Boston (1998)
17. Jardine, A.K.S., Lin, D., Banjevic, D.: A review on machinery diagnostics and prognostics implementing condition-based maintenance. Mech. Syst. Sig. Process. **20**(7), 1483–1510 (2006). https://doi.org/10.1016/j.ymssp.2005.09.012, http://www.sciencedirect.com/science/article/pii/S0888327005001512
18. Jin, X., Wah, B.W., Cheng, X., Wang, Y.: Significance and challenges of big data research. Big Data Res. **2**(2), 59–64 (2015). https://doi.org/10.1016/j.bdr.2015.01.006, http://www.sciencedirect.com/science/article/pii/S2214579615000076
19. Jouin, M., Gouriveau, R., Hissel, D., Péra, M.C., Zerhouni, N.: Prognostics and health management of PEMFC – state of the art and remaining challenges. Int. J. Hydrogen Energy **38**(35), 15307–15317 (2013). https://doi.org/10.1016/j.ijhydene.2013.09.051, http://www.sciencedirect.com/science/article/pii/S036031991302274X
20. Kalogirou, S.A.: Artificial intelligence for the modeling and control of combustion processes: a review. Progress Energy Combustion Sci. **29**(6), 515–566 (2003). https://doi.org/10.1016/S0360-1285(03)00058-3, http://www.sciencedirect.com/science/article/pii/S0360128503000583
21. Lee, J., Jin, C., Liu, Z., Ardakani, H.D.: Introduction to data-driven methodologies for prognostics and health management. In: Ekwaro-Osire, S., Goncalves, A., Alemayehu, F. (eds.) Probabilistic Prognostics and Health Management of Energy Systems, pp. 9–32. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-319-55852-3_2

22. Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., Siegel, D.: Prognostics and health management design for rotary machinery systems—reviews, methodology and applications. Mech. Syst. Sig. Process. **42**(1), 314–334 (2014). https://doi.org/10.1016/j.ymssp.2013.06.004, http://www.sciencedirect.com/science/article/pii/S0888327013002860
23. Liebowitz, J.: Expert systems: a short introduction. Eng. Fracture Mech. **50**(5), 601–607 (1995). https://doi.org/10.1016/0013-7944(94)E0047-K, http://www.sciencedirect.com/science/article/pii/0013794494E0047K
24. Luckham, D.C., Frasca, B.: Complex event processing in distributed systems. Computer Systems Laboratory Technical Report CSL-TR-98-754. Stanford University, Stanford 28 (1998)
25. Sarazin, A., Truptil, S., Montarnal, A., Lamothe, J., Commanay, J., Sagaspe, L.: Towards model transformation from a CBM model to CEP rules to support predictive maintenance. In: MODELSWARS 2020-The 8th International Conference on Model-Driven Engineering and Software Development, vol. 1, pp. 205–215. SciTePress (2020)
26. Siegel, J.: MDA guide, revision 2.0 (2014)
27. Truptil, S., et al.: Mediation information system engineering for interoperability support in crisis management. In: Popplewell, K., Harding, J., Poler, R., Chalmeta, R. (eds.) Enterprise Interoperability IV, pp. 187–197. Springer, London (2010)
28. Vachtsevanos, G., Lewis, F., Roemer, M., Hess, A., Wu, B.: Systems approach to CBM/PHM. In: Intelligent Fault Diagnosis and Prognosis for Engineering Systems, pp. 13–55. Wiley, Hoboken (2006). https://doi.org/10.1002/9780470117842.ch2, http://onlinelibrary.wiley.com/doi/10.1002/9780470117842.ch2/summary
29. Vichare, N.M., Pecht, M.G.: Prognostics and health management of electronics. IEEE Trans. Components Packag. Technol. **29**(1), 222–229 (2006). https://doi.org/10.1109/TCAPT.2006.870387
30. Xiaoxue, L., Xuesong, B., Longhe, W., Bingyuan, R., Shuhan, L., Lin, L.: Review and trend analysis of knowledge graphs for crop pest and diseases. IEEE Access **7**, 62251–62264 (2019). https://doi.org/10.1109/ACCESS.2019.2915987, conference Name: IEEE Access