Automated business rules and requirements to enrich product-centric information

Virginie Fortineau, Thomas Paviot, Samir Lamouri

Computers in Industry 104 (2019) 22–33

Automated business rules and requirements to enrich product-centric information

Virginie Fortineau[a,*], Thomas Paviot[b,c], Samir Lamouri[b,c]

[a] Khtema, Paris, France
[b] LAMIH-UMR CNRS 8201, UVHC, Le Mont-Houy, 59313,Valenciennes, Cedex 9, France
[c] Arts et Métiers ParisTech, 151 boulevard de l'Hôpital, 75013, Paris, France

Keywords:
Business rule
Requirements
Ontology
BIM
PLM

A B S T R A C T

Current PLM or BIM based information systems suffer from a lack of checking components for business rules. One reason is the misunderstanding of the role and nature of business rules, and how they should be treated in a product-centric information system. This paper intends to provide both a process and a related model to build such a component and enrich future systems. Rules and requirements process management enables the unambiguous formalization of implicit knowledge contained in business rules, generally expressed in easily understandable language, and leads to the formal expression of requirements. In this paper, the requirements are considered a consequence of the application of a business rule. A conceptual model is then introduced, called DALTON (DAta Linked Through Occurrences Network), which supports this process. In this ontology, concepts and product data, coming for instance from an existing product database, are represented using instances and occurrences, connected together with triples built from business rules and requirements according to previous management processes. An experiment involving a set of SWRL rules is conducted in the Protégé environment that validates the model and the process.

1. Introduction

The research presented in this paper has been accomplished in collaboration with a French company in the field of nuclear power plant engineering. Designing complex systems such as nuclear power plants involves hundreds of people working together for many years. From the very start of construction to the time when the power plant is operational, engineers use a vital enterprise asset: business rules. Business rules ensure that the product will fit all of the requirements, including those regarding performance or conformance with safety regulations.

Knowledge that is covered by business rules results from decades of successes, failures, optimizations, test and simulation results, maintenance operations, and so on. In the context of the digitalization of industrial design methodologies and tools (known, for example, as PLM or BIM), business rules must become a part of an efficient information system. However, many issues related to business rules management have been found in the literature, such as: knowledge elicitation, rule formalization, completeness and consistency of the business rules set, automated or assisted rule application, and automated rule checking on a set of actual data.

In this study, we do not cover the questions related to knowledge elicitation, as we treat rules that were already formalized. The problem addressed in this paper is limited to how to integrate formalized rules into a product-centric information system and enable them to be applied in order to verify product data. Moreover, we discuss these issues on a conceptual level, interrogating the nature of business rules, and the right processes to manage them, providing a management process and a rule/requirement/product meta-model as our main contributions. Implementation and technical issues are only provided as prospects as they depend on applicative (specific) constrains.

The two contributions of this work - the rule management process and the meta-model - are validated using a simplified real-case scenario and an ontology-based technology since ontologies are human readable and include an inference mechanism, which makes the proof of concept quick.

The paper is structured as follows. Section 2 is a literature review that targets both the next generation of PLM/BIM systems and the concept of business rule. Section 3 introduces a rule and requirement engineering process that could be supported by those systems. Section 4 presents a meta-model, called DALTON (an acronym for DAta Linked Through Occurrences Network), which is

* Corresponding author.
E-mail address: vfortineau@gmail.com (V. Fortineau).

DOI : 10.1016/j.compind.2018.10.001

a conceptualization of the proposed process to enable its implementation. Then, an ontological implementation to validate the model/process is described in Section 5 and results are discussed in Section 6. Finally, Section 7 concludes the paper.

2. Towards the next generation of industrial enterprise systems

2.1. PLM, BIM, closed-loop

PLM is the acronym for Product Lifecycle Management. Several definitions exist in the literature, merging two points of view: the information system point of view, which considers PLM as an integrated enterprise IT solution to manage product-related data [19,27,37], and the methodological point of view, which considers PLM as an approach aimed to manage information about a product among all stakeholders during the entire product lifecycle. A definition from Stark [39] provides a consensual point of view about what PLM actually is:

"Product Lifecycle Management (PLM) is the business activity of managing, in the most effective way, a company's products all the way across their lifecycles; from the very first idea for a product all the way through until it is retired and disposed of".

Terzi et al. [40] provided an overview of the history of PLM: the first PLM systems that were developed were mostly document management systems, enabling the centralization of different product views generated along a product lifecycle (design sketches, Bill of Materials, 2D and 3D files, etc.). Then, PLM systems moved toward data management systems, starting with design data as an extension of PDM systems (product data management systems, repositories of all design data related to the product). The first issue for a PLM was to include all of the phases in the product lifecycle. Since this historical review, many works have notably discussed the design/manufacturing interface by targeting, for instance, PLM/ERP interoperability [34] or integrating design and assembly stages [28].

Going step by step to the middle of life (MOL), and especially the usage/maintenance phase, PLM studies have faced an important issue: at this point of the product lifecycle, the information that was centralized in the manufacturing enterprise's IS is then spread out. The concept of Internet of Things (IoT) has emerged, which states that inform1ation is supported by the product (or part) itself and not by the enterprise information system [43]. In a way, IoT is a complementary approach to PLM, solving the hole of the Middle of Life phase.

Complementarily to the IoT concept, Kiritsis [25] proposes the closed loop. Whereas the aim of PLM is to master and exchange information along the whole product life-cycle, closed loop PLM aims to integrate knowledge from the other stages of the lifecycle to improve decision-making, notably during the design phase. Thus, closed-loop PLM is the continuous learning version of PLM.

From the building industry perspective, information systems and methodologies related to digital data management are known to belong to the BIM area, which is the acronym for Building Information Modeling [42]. Recently, the terms "closed-loop" and "BIM" have appeared in the literature: Chang et al [8] run simulation loops, integrated with the building data, in order to optimize the thermal structure of the façades.

In the use case of our study, a nuclear power plant could be considered the object of a BIM study, since it is a building, as well as from the PLM point of view, because the items that operate in a nuclear power plant (valves, tanks, motors, pumps etc.) are produced using PLM tools. Regardless, despite the fact that the businesses of AEC (targeted by BIM) and classical industrial sectors (aerospace, automotive, steel industry) are slightly different, there are similarities and connection points that will hopefully make those systems more cooperative in the future [25].

Finally, a study by Panetto et al. [33] provides a consistent view of PLM, BIM and their possible future. They consider the future of Enterprise Information System and reveal two major trends:

- moving from system integration to system interoperability;
- and from dynamic (exchanged) and real-time data to inferred data, which means automated information generation and checking.

They provide more detail on this second point:

"it is also important to note that recent advances in information and communication technologies (ICT) have allowed enterprises to move from highly data-driven environments to a more cooperative information/knowledge-driven environment".

Therefore, the issue discussed in the paper is how to enrich (or extend) current systems (PLM/BIM) with intelligent information produced during the system engineering process (for instance, in the form of business rules).

2.2. Problem statement: implementing business rules in a product-centric information system

The research area of this study is the integration of knowledge in an information system. Applied to industrial Information Systems (IS), the correlated problem is the integration and management of business rules in a PLM / BIM system. Business rules are indeed the IS formalization of knowledge. For instance, as Yao and Kumar [44] explain, while designing a decision-making tool for medical purposes based on ontologies, "procedural knowledge is encapsulated in rules", and the knowledge manager provided is an OWL API, combined with a JESS engine to execute SWRL rules. Also in the domain of mobile devices, Nalepa and Bobek [30] state that "a rule-based system consists of three main elements: knowledge base, fact base and inference engine. Knowledge base is considered as a set of rules". For BIM applications, Sacks et al. [38] consider that rules "encapsulate expert knowledge of the domain". Finally, Bajec and Krisper [2] conclude that "the roots of business rules come from the Artificial Intelligence community, where they have been successfully applied as a way of representing knowledge".

Moreover, the Business Rules Group (BRG) defines a business rule as follows [23]:

"A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behaviour of the business. [. . .] From the information system perspective, it pertains to the facts that are recorded as data and constraints on changes to the values of those facts".

The double-nature of this definition confirms that a business rule is at once an element in the information system as well as an expression of business-related knowledge. Thus, implementing business rules raises the issue of integrating knowledge into an information system, as described in Fig. 1.

The research question in this study is therefore how to integrate and manage business rules in a product-centric information system (PLM or BIM). Thus, we consider that the elicitation of rules is already done, because knowledge elicitation is an(other) issue in of itself, and we focus on the computer-aided management of the existing rule set. The industrial application domain (the nuclear industry) adds some additional requirements to this study:

- given the amount of data and the number of rules, rule management should include automated verification of the rule;
- given the amount of data and the number of rules, the checking algorithms should be generic;

- given the lifespan of the power plant, the provided system should be based on open-source technologies, and shall not rely on commercial tools;
- the IS expression of rules should be easy to understand by a business user;
- full traceability of information must be ensured, from expression to verification;
- the model provided should anticipate the fact that, given the amount of data, a distributed architecture might be preferred, and that data can be exchanged with stakeholders separately from the corresponding concept/knowledge base. For this reason (and others that will be developed later), meta-modelling is required.

2.3. Difficulties of checking rules

The literature on rule checking and business rules management is important. The literature reveals that the checking of single rules isn't the main challenge. Rule languages are in fact well suited to do so (SWRL, OCL, Prolog, etc.); rather, the issue is automatically checking a set of rules with minimum effort on the part of the end user or from IT support.

The literature on business rule processing can be divided into two types: the business side, which provides methodologies or languages to manage business rules that are mainly not computable. Like Bajec and Krisper [2], some authors indeed consider that "business rules are set and owned by the business and have to be therefore managed by the business". The process model BPMN (Business Process Model and Notation), for instance, enables end users to write business rules in a text format. The SBVR language (Semantics of Business Vocabulary And Rules Specification) provides a structure for business rules based on business concepts, but is not implemented such that it is executed on product data. On the other hand, IT experts have many languages to execute rules such as Prolog, Java, SWRL, etc. However, they are mostly captives of modelling languages, demand a computable expression for each rule, and are not readable by business users. Business rule management must be considered as a continuous process of reconciling both sides [16], such as the initiative by Knuplesch and Reichert [26] in the healthcare domain.

2.4. The lack of a generic, time-based approach

In Table 1, only the works that provide automated (computer-assisted) verification of the rules are collected. As this table reveals, most of the contributions to business rules management tend to provide a specific computational expression for each rule. Therefore, the involvement of an IT expert is required. Some tools limit the user's commitment (especially IT) in implementing a rule, such as Drools[1], which provides patterns for the rule's computational expression. Also, Fortineau et al. [16], Njonko et al. [31], as well as the SBVR initiative [3] try to express rules in a controlled, natural language, to automatically transform the natural expression of the rule into a formal one "in order to be machine-processed" [31]. The output of this approach, as explained by Beach et al. [4], "leads to regulatory compliance systems that are often closed and can only be maintained by dedicated software developers. This [. . .] is simply not viable in the complex and continuously changing regulatory landscape". Most of the time, they try to develop rule engines, where the rules are stored and can be executed at one time on a set of data.

To conclude, the literature demonstrates that existing contributions on business rules management in engineering mainly rely on two hypotheses that we believe to be false:

- a set of business rules can be checked at a given moment for a set of data. We think that checking a set of computational rules at one time during the SE process is a mistake. There is a "right moment" for each rule, i.e. when the corresponding data (the data that activates the rule, and the data that enables the verification) actually exist. For each rule, those moments are different, so each rule should follow its own workflow. Hence, checking a rule is not an event but a full workflow, consisting of different stages that may not be simultaneous (the data that activates the rule are potentially produced before the data required for the rule verification, so both stages may not be simultaneous);
- each rule must be checked individually and a computational expression for each rule is required. When the amount of data grows, this second assumption is a scientific lock.

We intend to provide a generic framework for rule processing, enabling the verification of all rules with a generic (and unique) computational expression. This framework relies on a continuous rule engineering process consisting of different stages with various temporalities.

3. Contribution: a rule and requirement engineering process

3.1. Why rules should not be checked

In fields other than engineering, the literature review on "rule checking" is surprisingly poor. In fact, the vocabulary associated with a "rule" is not "checking" but "applying". In the legal domain, notably, decrees are delivered in the application of laws [20]. In addition, Yao and Kumar [44], in designing a rule engine for medical purposes, state that rules are related "to the stage at which they are applied in the clinical process". Moreover, Zeichner et al. [46], when using crowdsourcing for rule evaluation state that "As mentioned, in instance-based evaluation individual rule applications are judged rather than rules in isolation, and the quality of a rule-resource is then evaluated by the validity of a sample of applications of its rules." This semantic issue is not just a detail, because considering that a rule should be applied and not checked is an important paradigm shift. This raises two research questions:

- "When should the rule be applied?"
- "What is the generated output of the application of a rule?"

The first question is discussed in Section 3.3 and the second question is in the next Section 3.2.

3.2. The relationship between business rules and requirements

3.2.1. Definition of requirements regarding the notion of business rules

Managing business rules calls for the use of related notions such as specifications, requirements or constraints. Contrary to the definition of business rules that was discussed in multi-disciplinary groups like the BRG, the definition of related notions in the literature are in movement; no consensual definition has emerged from the literature review. Moreover, this is a semantic issue, and a semantic issue will always face limits in that different languages will have various definitions for terms. In addition, English is the common language in the research field, but global researchers tend to define concepts given an instinctive interpretation from their native language. As the literature cannot provide a consensus on

those terms, we have chosen to provide our own definition of all concepts to avoid any misunderstanding on the part of the reader.

Requirements are frequently defined by the process that manages them, called requirement engineering. More precisely, they are defined by the aim of each phase of the requirement engineering process. For instance, Greenspan et al. [21], who 24 years ago expressed their "debt" to requirement definition, state, "Requirements definition is a careful assessment of the needs that a system is to fulfil", usually in the form of functional specifications. In their view, a requirement is a global statement that, when processed, produces detailed specifications on the expected size, behaviour, or material of a system or a part of the system. In 2007, when making a state-of-the-art on requirement engineering in software development, Cheng and Atlee [9] still did not define the requirement in itself, but give a definition of each step of the requirement engineering process. Starting with requirements elicitation: "Requirements elicitation comprises activities that enable the understanding of the goals, objectives, and motives for building a proposed software system. Elicitation also identifies the requirements that the resulting system must satisfy to be considered acceptable." We can understand from this definition that a requirement expresses an "expected result" of the system that can be of different natures. Finally, in the second edition of their book published in 2017 [12], gives the following definition of a requirement: "Requirements are the basis for every project, defining what the stakeholders – users, customers, suppliers, developers, businesses – in a potential new system need from it and also what the system must do in order to satisfy that need."

The abstraction level of a requirement also cannot find a consensus in the literature: it could be a high level expectation (notably in the primary phases of design) or detailed specifications, usually divided into functional and non-functional specifications. In a large-scale study in 2017 on the "*pain in requirements engineering*", Fernandez et al. [14] proved that having overly abstract or not well-defined requirements is the main cause of requirements engineering failures. A reason why the notion "requirement derivation" exists in the literature [12] is that it refines the requirement to a sufficiently detailed format to allow it to be checked. We reached the same conclusion during our applicative project, and the derivation of business rules is discussed is possible by adding a taxonomic link between business rules. In the remainder of the study, however, we consider only the sufficiently detailed and complete business rules for implementation and treatment in the information system.

Moreover, the requirement engineering process has about the same stages as the business rule engineering process (elicitation, modelling, verification/checking, management) [9]. The main difference between a requirement and a business rule seems to be the "always true in a given condition" nature of a rule: while a requirement, in our opinion, ***is a given specification constraining the system, usually expressed by a specific stakeholder, in a specific project***, a business rule is ***a specification that should always be applied, for any equivalent system, when its condition is fulfilled***. For example: "machine A should have an emergency button to stop it when pressed" is a requirement specific to machine A, expressed by a project stakeholder. However, "when a machine is used by a human being in production, it shall always have an emergency button to stop it when it is pressed for security reasons" is a business rule, covering the knowledge of the company engineers or regulatory constraints, and that will apply on any machine that fits the condition (for instance, machine A).

With this example and the given definitions, it is clear why, in the remainder of this paper, we consider the requirement is (but not exclusively) the output of the application of a rule. This is consistent with the literature on business rule formalization.

3.2.2. Formalization of business rules

Usually, rule languages divide a rule into two parts: the condition (the IF part) and the implication (the THEN part). Thus, the IF/THEN format is the most widespread representation of formal business rules. Even Nalepa and Bobek [30] propose "rules, usually of a form of production rules: IF < conditions > THEN < action>". The Rule Interchange Format (RIF, [47],), which is a W3C recommendation, has a similar structure.

However, while the nature of the content of the IF part is consensual (it is the set of conditions required to apply the rule), the content of the THEN part is not clearly defined. For Nalepa and Bobek [30], it is an action. For the RIF language, it is an implication. According to Yao and Kumar [44], "rules are triggered by contextual data and produce actions, which are generally in the form of reminders, alerts, and recommendations". The term recommendation is interesting, because it is not in the lexicon of acting, but it refers to "an expected state or result"; in other words, a requirement (mandatory or optional).

Thus, the proposed structure of a business rule is the following: when the < condition(s) to fulfilled > is true, then the < expected result> (a requirement) must be checked regarding the product data.

3.3. When to apply rules

Usually, the management of business rules is divided into two phases: the rule expression and checking. However, we state that because rules are the application of user knowledge, they should be processed all along the product lifecycle. Even a single rule may be applied at different stages of the lifecycle. For example, a rule related to the evacuation time during a fire is applied during the design phase (for a simulation), and during the usage phase (for a real-life exercise). This rule has two occurrences for the application. In fact, rule processing is re-iterated at each phase of the product lifecycle during an engineering process, which aims to apply knowledge: the system engineering process.

System Engineering (SE) is an engineering process during which the engineers perform a system analysis to translate initial requirements into technical requirements and, finally, produce the product view at each step of the product lifecycle [29]. It is a recursive and iterative process when engineering knowledge is applied to conceive the best product representation. In the ISO 15,288 standard [24], the product view is considered the decomposition of the logical view following requirements engineering during the SE process. Consequently, we consider that the SE process is the process in which business rules are applied and that the SE process is orthogonal to the product lifecycle.

Fig. 1 synthetizes those primary thoughts: knowledge management is achieved among the time a business expert's knowledge is elicited in the form of business rules. Business rules are processed during the SE process in every moment of the product lifecycle. The SE process produces requirements constraining the product views and should be checked with regards to the product data generated along the product lifecycle and stored in a product-centric system (PLM or BIM, for instance).

3.4. Processing time-based rules and requirements

As previously presented, the verification of a rule is made in several stages: condition fulfilment and checking related requirements. Moreover, rule verification can occur at a different time in

Table 1
Bibliography for automated rule checking and business rules management.

Paper	Domain	Modelling language	Rule language	Rule engine, tool	Business rules	Comput. expression
Yao and Kumar [44]	Healthcare	OWL	SWRL	Jess	yes	Specific
Nalepa and Bobek [30]	Mobile devices	—	Prolog/HMR	HeaRT	yes	Specific
Zayaraz et al. [46]	Language extraction	Ontology	Nave Bayes		no	generic
Bernardi et al. [5]	Process management	—	Declare		yes	generic
Aichernig and Schumi [1]	Process management	XML	C#	FScheck	yes	specific
Fortineau et al. [17]	PLM	OWL	SWRL	Protégé	yes	Specific
Njonko et al. [31]	All	RuleCNL Vocabulary	RuleCNL	RuleCNL	yes	specific
Pham and Le [36]	Automated processes in bank and accounting	PetriNets	ECAE	ECAE reasoner	yes	specific
Beach et al. [4]	AEC	OWL, IFC-OWL	SWRL (with RASE rule converter)	Protégé	yes	Specific
Choi et al. [10]	AEC	IFCs		InSightBIM-Evacuation	yes	specific
Study target	PLM/BIM	any	any	To define	yes	specific

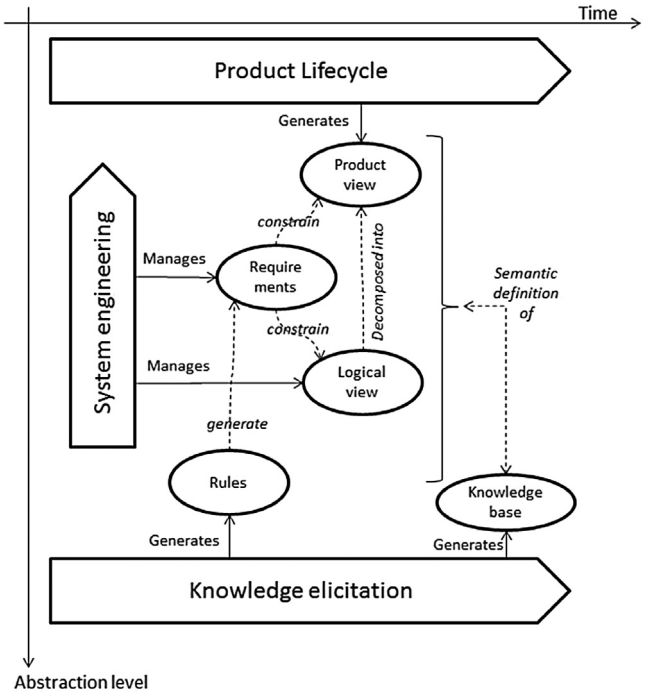

Fig. 1. Relations between System Engineering, Knowledge Management and Product Life- cycle.

the product lifecycle. Finally, business rules must be formalized in order to be processed. For these reasons, a rule engineering process is proposed that covers the various steps of rule engineering. Moreover, because all steps are not simultaneous, it has been considered that each step must produce a resulting piece of data to be stored in order to complete the traceable information. The proposed rule engineering process, which consists of 4 stages, is synthetized in Fig. 2.

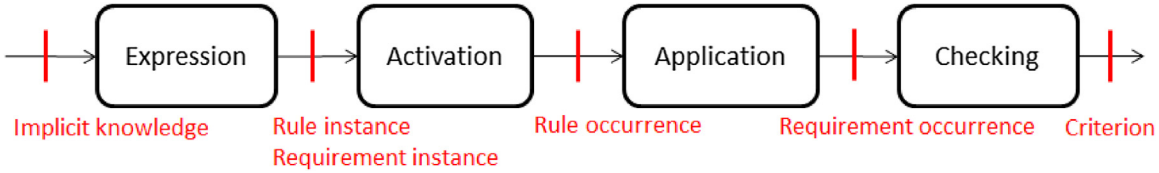

Fig. 2. Rules and requirements process.

3.4.1. Stage 1: expressing the rule (what the rule means)

Processing the business rules starts with the expression of the rule. This formalization is based on abstract concepts and not on product data, because a rule is always true and can be written before any technical data is actually produced. Therefore, rule management supposes that a related concept base is built. The formalization is made of two parts: the condition of the rule and the expected result of its application, which is modelled as a requirement.

3.4.2. Stage 2: activating the rule (when the rule applies)

Stage 2 is the activation of the rule. This stage represents the time when the rule should be applied. A rule is indeed formalized before any actual data is produced. Later, during the SE process, some data fits the condition of the rule and requires the application of the rule. In the proposed process, the output of the rule activation (stage 2) is an occurrence of the rule. There might be several occurrences for the same rule (each time that the rule should be applied during a product lifecycle).

The concept of rule activation exists in the literature. Pham and Le Than [36] embed the notion Event-Condition-Action type rules (ECA) of activation in the form of an event. The approach of Bernardi et al. [4] uses DECLARE as declarative process modelling language [35] and makes use of the notion of rule activation:

> "An activation of a business rule in a process instance is an event whose occurrence imposes certain obligations on the occurrence of other events in the same process instance. For example, for the business rule "every request is eventually acknowledged", each request is an activation, since the occurrence of a request forces an acknowledgement that it will eventually occur. In this case, the activation request becomes a fulfilment or a violation depending on whether it is followed by an acknowledgement or not. In our context, the notion of rule activation is crucial because we try to identify characteristics of the lifecycle of an activation that allow us to discriminate by whether that activation is a fulfilment or not."

However, they mix the activation and the application of the rule in this definition when we consider that they must be separated, because they are not always simultaneous. Moreover, they consider whether there is a fulfilment of the obligation (requirement) related to the application of the rule, whereas later we show that there could be several.

3.4.3. Stage 3: applying the rule (what the rule implies)

Stage 3 is the application of each activation (occurrence) of a rule. Applying the rule assumes an expression of the expected result of the rule in the specific context that activates the rule: the output of stage 3 is an occurrence of the requirements, specifically related to the occurrence of the rule that was applied.

3.4.4. Stage 4: checking the corresponding requirements (consistency with the product data)

Stage 4 checks the occurrence of the requirement; this means that the set of data must be consistent with the requirement. There can be different existing values for the same attribute constrained by the requirement. For instance, a pressure has two values: one calculated during the design phase, and the second measured during the usage phase. The first might fit the requirement, but the second might not. Each one is stored in a different criterion with a different result. This concept of criterion corresponds to the "multiple fulfilments and violations of a rule" expressed in [4]:

> "In process instance? a; b; a; c? for example, the response rule is activated twice, but the first activation leads to a fulfilment (eventually b occurs) and the second activation leads to a violation (b does not occur subsequently) [. . .] An algorithm to discriminate between fulfilments and violations of a rule in a process instance has been presented by [7]."

The output of stage 4 is one or several criterion, each one reflecting the fulfilment (or not) of the occurrence of requirement regarding a piece of data. Hence, a rule might be verified and not verified at the same time.

4. The DALTON model supporting processing rules and requirements

4.1. A generic meta-model based on triples

Rules and requirements are formalized based on engineering concepts. Therefore, a model of concepts is required to support the expression of rules. To support a product lifecycle, this model suffers from several constraints, the first being to manage non-canonic data. In order to understand this constraint, let us consider the following two rules:

- Rule 1: "the materials of seismic systems must be painted blue";
- Rule 2: "the valves must have the code BR."

During the product lifecycle, the technical data valve123 is created. It is a valve in the primary system. This valve is concerned with both rules, but not in the same way: it is concerned with the first because it is part of a seismic system, and by the second because it is a valve. Applying both rules on valve123 supposes it is categorized as a valve and as a material of a seismic system; i.e., taxonomies are needed. Moreover, we need non-canonic taxonomies (this means that an individual may belong to two or more concepts that are not strictly dependent on one another. Indeed, in the example below, some seismic systems might not have valves and not all valves are part of seismic systems).

The DALTON model is a conceptualization that supports rule processing in order to implement it in any technology. DALTON stands for "DAta Linked Through Occurrence Network" because the model relies on a network of data labelled using a network of concepts, the data being the actual occurrences of concepts. This conceptualization, presented in Fig. 3, is independent of any technology or modelling paradigm. However, for better comprehension, an UML representation of the DALTON model is presented in Fig. 4.

There are two kinds of objects in the DALTON model:

- instances are objects representing the concepts;
- occurrences are objects representing the applicative data related to concepts. For example, the instance valve can have different occurrences: valve3453, valve38, etc.

The DALTON model is generic; i.e., it has neutral classes (the classes are not business-oriented) to fit the issue of genericity and automation. A model supporting business rules must be semantically rich. The richness of information lies in the relations between concepts, and not in the concepts themselves. Then, the network of instances and the network of occurrences are modelled as lists of triples. A triple is made of three elements: a subject, a predicate and an object. The proposed model is therefore based on triples of instances, related to (populated with) triples of occurrences. The triple-like representation is classical in ontology-based approaches. The Resource Description Format (RDF) is, for

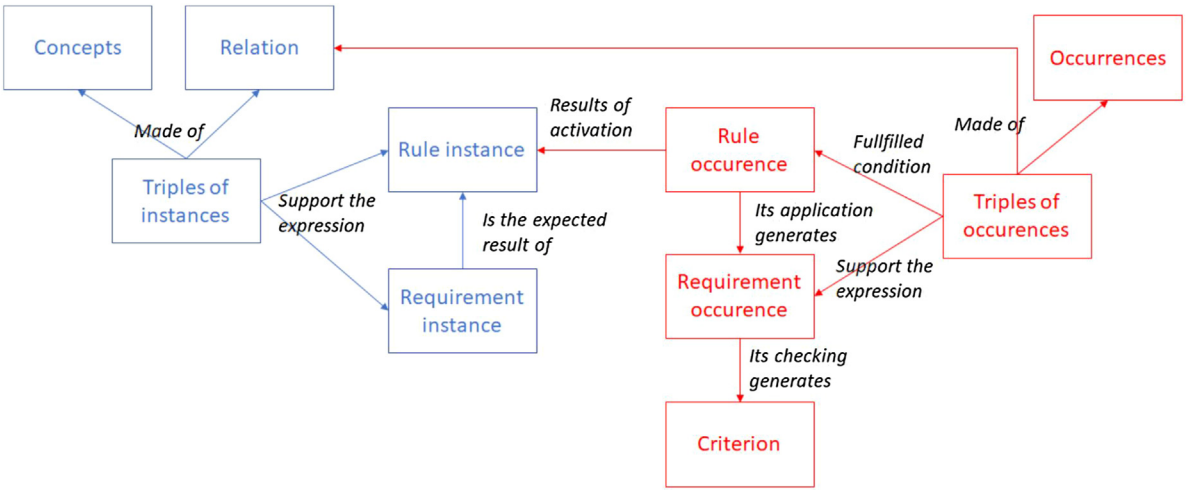

Fig. 3. Conceptual representation of the DALTON model.

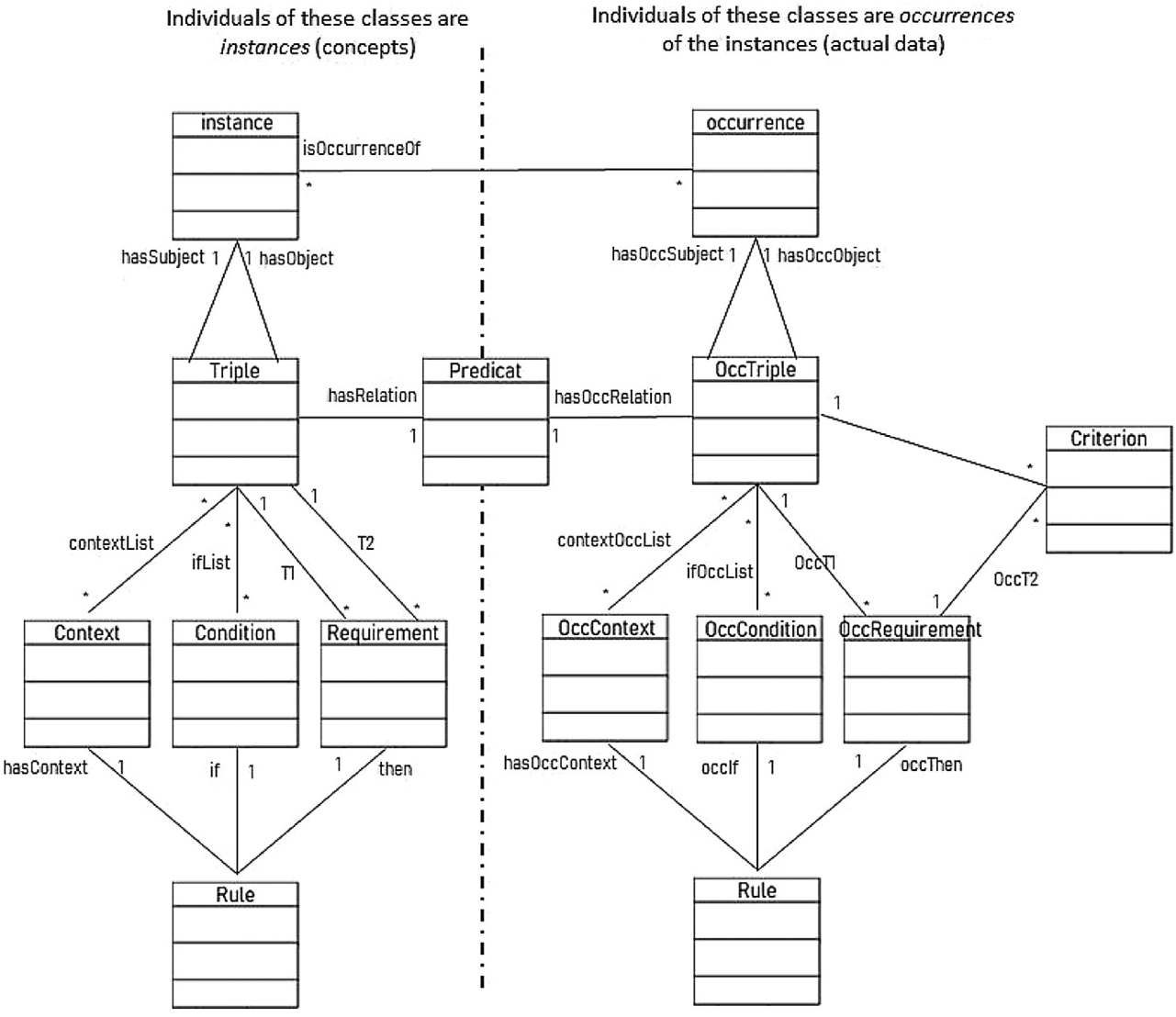

Fig. 4. The DALTON conceptualization formalized as an UML-like class diagram.

instance, a triple-based modelling language. It is also the right way to formalize natural language. For instance, Zayaraz et al. [45] extract sentences in the form of triple, using a Binary Decision Tree (BDT)-based engine.

The main elements of DALTON are presented in Fig. 3. The fundamental classes of instances are:

- triples of concepts;
- instances of rules and requirement (formalization of the rule).

Built as a mirror of the instances, the fundamental classes of occurrences are:

- triples of occurrences, representing the data network;
- occurrences of rules, resulting from the activation of rule instances;
- occurrences of requirements, resulting from the application of rule occurrences;
- occurrences of criterion, resulting from the comparison of requirements;
- occurrences of each existing data value.

4.2. A generic rule and requirement model

The DALTON model helps to formalize rules in the form of three elements. As with most rule languages, a rule is made from a condition and an implication. The condition is a list of triples reflecting the situation that activates the rule, and the implication is a requirement. Moreover, the use case rules raise the point that the condition may not be directly related to the implication. For example, in the following rule: “the materials of seismic systems must be painted blue”, the condition is “systems are type seismic” (and only that), and the implication is “the material must be blue”. However, how can one know that the materials of the concerned system are covered by this rule? This link between the condition and the implication is stored in the DALTON model as the *context* of the rule, which is a list of triples. This is completed by a requirement model based on 4 elements and built according to the literature [11]:

(1) the constrained element, to which the requirement is directly related. It can be a function, a material, a system, etc.;
(2) the constrained attribute;
(3) the expected value of the attribute. Here, the value must be considered in the wider sense; it can be a quantitative or a qualitative element;
(4) the context to find the right constrained element.

In the DALTON model, these four elements are represented using two triples. The first triple represents the constrained element and its attributes. The second represents the attribute and its link to the expected value. The context is not expressed in the two triples, because a requirement is generated by a rule, in which the context is already stored.

5. Ontological validation of the model

5.1. A SWRL ontology for model validation

To prove the concept of the DALTON model, a DALTON-like ontology was implemented in Protégé software using SWRL rules to validate the rule and requirement treatment mechanisms. Despite the fact that it is an OWL file, this ontology has RDF expressivity that allows it to stay consistent with the UML model proposed in Section 4.1. Indeed, we did not use OWL axioms to define the ontology, because it demands to define specific / business-oriented classes [15]. As a matter of fact, the T-Box reflects the generic classes of the DALTON classes, and the inference is operated on the A-Box, using SWRL rules.

The properties of the ontology define the meta-relations between DALTON classes. They are listed in Table 2.

The use case consists of a single rule, LSN2 that states: “if the fluid that passes through a system is borated water, then the physical materials of this system must have characteristic MaterialCode worth BA”. The rule is instantiated with a condition, a context, and a requirement ex2 as follows[2]:

- if: {system - isPassedThrough - fluid ; fluid - has - fluidType ; fluidType - isWorth - Borated Water};
- context: {system - isMadeUpOf - material};
- then ex2: {tl = material - has - materialCode ; t2=materia1Code - isWorth - BA};

Then, occurrences are added (meaning actual systems and materials). The system considered is called "ASG", which is part of the secondary circuit of the power plant. ASG is an individual of the *Occurrence* class, and is *aKindOf* system. The fluid passing through the ASG is the SecondaryFluid, which is borated water. Materials of the ASG system have two ASGMaterialCode values: BA and BB. Three outputs are expected as a result for the reasoning:

- because the fluid passing through the ASG system is borated water, the rule LSN2 is activated;
- this rule occurrence generates a occurrence of requirement, called Occex2, with its first occurrence triple being {ASGMaterial - has - ASGMaterialCode};
- ASGMaterialCode has two values: BA that fulfils Occe x 2 t, and BB that violates Occex2.

Ten SWRL rules (see Table 3) are used to get this result. Rules SWRL1 and SWRL2 perform a preliminary reasoning: they map the triple of occurrences to the corresponding triple of instances. For example, they detect that the triple {ASG - isPassedThrough – SecondFluid} *isType* {system - isPassedThrough – fluid}. Rules SWRL3 to 5 “activate” LSN2; that is to say, it has an occurrence (OccLSN2) that exists because the following list of occ triples exists and fulfils the condition of LSN2:

- If: OccLSN2 = {ASG - isPassedThrough - SecondFluid ; Second-Fluid - has - SecondFluiTyp ; SecondFluidTyp - isWorth - BoratedWater}

SWRL 6 finds the context of OccLSN2: it is the triple of occurrence {ASG - is made up of - ASG Material} which means that OccLSN2 can be applied. The application of OccLSN2 is achieved by using the following algorithm:

- link OccLSN2 to the occurrence of requirement generated (Occex2) with SWRL7;
- find the occurrence constrained by Occex2 thanks to ex2 also with SWRL7;
- find the first triple of occurrence (*occT1*) of Occex2 with SWRL 8: *occTl* = {ASG Material - has - ASG Material Code}

[2] The list of triples depends on the concept base that is specific to each application. Other concepts and relations could have been used, without impacting the process of the rule.

Table 2
List of owl properties.

owl property	meaning	UML corresponding relation
aKindOf	maps an occurrence to its corresponding instance	isOccurrenceOf
isType	maps a triple of occurrences to its corresponding triple of instances	Is not in the UML model, it is an inferred relation
condα	links a condition triple (number α) to the corresponding rule	Mutualisation of “if” and “ifList”
contα	links a context triple (number α) to the corresponding rule	Mutualisation of “hasContext” and “contextList”
imply	links the rule to its related requirement	Then
co_Obj	defines the occurrence object of a triple of occurrences	hasOccObject
co_Sub	defines the occurrence subject of a triple of occurrences	hasOccSubject
co_Pred	defines the relation predicate of a triple of occurrences	hasOccRelation
tm_Obj	defines the instance object of a triple of instances	hasObject
tm_Sub	defines the instances subject of a triple of instances	hasSubject
tm_Pred	defines the relation predicate of a triple of instances	hasRelation
constElt	defines the constrained. element of the requirement (subject of the first triple defining the requirement)	Mutualisation of “occT1” and “hasOccSubject”
exT1	links the instance of requirement to the first triple of instances that defines it.	T1
exT2	links the instance of requirement to the second triple of in stances that defines it.	T2
occT1	links the occurrence of requirement to the first triple of occurrences that defines it.	occT1
crit	Links the occurrence of requirement to the criterion (the second triple of occurrences)	occT2
fulfil	When the value in the criterion fulfils the occurrence of requirement	Attribute of class “Criterion”

The checking of the requirement Occex2 is achieved in two steps:

- detect the existing values of ASG Material Code with SWRL9 and store them as a criterion;
- compare them to the required value with SWRL10, thus detecting whether it fulfils the requirement.

5.2. Limits due to SWRL expressivity

The ontological implementation of the rule LNS2's use case with the help of ten generic SWRL rules validates the fact that the proposed DALTON model enables reasoning mechanisms to express, activate, apply rules and check requirements. However, this ontology faces some issues that lead to adjustments in the reasoning, but are only due to the use of SWRL and its lack of expressivity [17,18]. Indeed,

- SWRL is not a query language: the detection of condition and context length cannot be automated, because it requires a query;
- SWRL cannot generate individuals: all individuals are created manually, and the SWRL rules infer only the relations between individuals;
- SWRL reasoning is based on the Open World Assumption (OWA): rule SWRL10 checks that the triple of occurrences {ASG Material Code - is worth - BA} fulfils the requirement Occex2 but it cannot infer that the triple of occurrences {ASG Material Code - is worth - BB} violates the requirement Occex2. The OWA indeed prevents any negative reasoning.

6. Discussion and perspectives

The model presented in this paper targets highly large and complex systems, composed of thousands or million parts closely interconnected through many semantic relationships. The development and operation of such systems require engineers to apply many business rules coming from local, national or international regulations (aerospace industry, nuclear power plants) or from the enterprise knowledge. In this context, it order to achieve a “closed loop PLM” as presented in the introduction, the model acts as a basis for a systematic way to address the workflow from the business rule to the requirement checking over the system lifecycle (a Product or a Building). The purpose if clearly to partially automate something that is manually achieved or, in the worst case, impossible to achieve if it appears that the business rules applied to millions of occurrences lead to inconsistencies. In order

Table 3
List of reasoning (SWRL) rules.

SWRL1	Pred(?a),co_Obj(?e,?f),co_Sub(?e,?h),co_Pred(?e,?a),isKindOf(?f,?d),aKindOf(?h,?g),tm_Obj(?c,?d),tm_Sub(?c,?g),tm_pred(?c,?a)→isType(?e,?c)
meaning	find the OccTriples corresponding to the Concept Triples
SWRL2	Value(?f),co_Obj(?e,?f),co_Sub(?e,?h),co_Pred(?e,?$is),aKindOf(?h,?g),tm_Obj(?c,?f),tm_Sub(?c,?g),tm_Pred(?c,?$is)→isType(?e,?c)
meaning	like rule SWRLl but for values
SWRL3	condRulel(?a,?b),aKindOf(?d,?a),isType(?c,?b)→condl(?d,?c)
meaning	find the occ triples that satisfies the first concept triple of the rule condition
SWRL4	co_Obj(?e,?f),co_Sub(?c,?f),condl(?d,?e),condRule2(?a,?b),isKindOf(?d,?a),isType(?c,?b)→cond2(?d,?c)
meaning	find the occ triple that satisfies the second concept triple of the condition and is connected to the occ triple found previously.
SWRL5	co_Obj(?e,?f),co_Sub(?c,?f),cond2(?d,?e),condRule3(?a,?b),aKindOf(?d,?a),isType(?c,?b)→cond3(?d,?c)
meaning	find the occ triple that satisfies the third concept triple of the condition and is connected to the occ triples found previously.
SWRL6	co_Sub(?c,?f),co_Sub(?e,?f),cond1(?d,?e),cont1(?a,?b),isType(?c,?b),aKindOf(?d,?a)→cont1(?d,?c)
meaning	find the occTriple corresponding to the context
SWRL7	co_Obj(?c,?e),contl(?a,?c),exTl(?f,?g),imply(?b,?f),aKindOf(?a,?b),aKindOf(?e,?h),aKindOf(?j,?f),tm_Sub(?g,?h)→constElt(?j,?e),imply(?a,?j)
meaning	apply the rule, by creating the requirement occurrence, and finding the constrained element (occurrence)
SWRL8	co_Sub(?c,?b),constElt(?a,?b),exTl(?d,?e),aKindOf(?a,?d),isType(?c,?e)→occTl(?a,?c)
meaning	find the first occurrence triple of the requirement thanks to the constrained element
SWRL9	co_Obj(?b,?c),co_Sub(?h,?c),co_Pred(?h,?$Worth),aKindof(?c,?f),exT2(?d,?e),occT1(?a,?b),tm_Sub(?e,?f)→crit(?a,?h)
meaning	detect the existing values of the constrained attribute (object of occTl) and store them as “criterion”
SWRL10	crit(?a,?c),isType(?c,?d),aKindOf(?a,?b),exT2(?b,?d) →fulfil(?c,?a)
meaning	compare the existing values of the constrained attribute with the expected one (stored in exT2) and detect if it fulfils exT2

to reach this stage of a partial automation, following issues are identified to be overcome: a tool to assist business and requirement processing (Subsection 6.1), consistent data models (Subsection 6.2), a system architecture (Subsection 6.3) and its related implementation (6.4).

6.1. A graphical vision of rules and requirement processing

With the contributions of virtual reality, system engineers should be provided a virtual knowledge environment to evolve in, which will grow dynamically following the product lifecycle. Thus, a graphic representation of the proposed conceptualization provides real added value. The huge amount of data leads to an investigation of at least 3 dimensions and even four dimensions, as in BIM, where the time dimension is considered in the digital model. With the time dimension, the graphic representation could support the data workflow that is part of PLM/BIM systems.

A graphic representation of a rule (see Fig. 5) - taken from the use case - is proposed in order to check how the proposed conceptualization changes the engineering view of business rules and suggests a new vision for "checking rules". This graphic representation is dynamic and enriched time after time with the data produced from the rules and requirements process. It changes the paradigm of the map verification currently made in the design process of power plants and shows that the vision of "rule checking," based on the binary result of a single processing of a computable version of the rule with regards to a set of data, is obsolete.

6.2. The interoperability of applicable models

In this contribution, the product data is supposed to be represented in a single model, which is not true in a PLM/BIM context, where different applicable views can co-exist. As a matter of fact, the rule and requirement process climbs over the various applicable product models, which is why it is important to create and store data at each step in the process.

For example, the elements in the condition of the rule provided in this study are generated by hydraulic engineers when producing the 2D design file. However, the material code is chosen a few steps later by industrial engineers while sourcing the right supplier. They used to work with different tools based on different product models.

Therefore, one issue remains that the DALTON model currently does not cover: how can the rule-related data be transmitted from one model to another and the different modelling paradigms be managed? We are working on the possibility of designing a unified representation as a pivotal model. A first approach is to use a standard for the pivotal model, but this would lead to a semantic issue, as explained by Beach et al. [3]:

"Choi et al. [8] have developed an approach to regulation checking for high-rise and complex buildings. In their work the authors have specified the regulations in a way that tightly couples them to the industry standard representation of data, the IFCs (Industry Foundation Classes, 2005). This means that implicitly the IFCs have been utilised as the underlying semantics of the model. The key problem with this approach arises if the semantics of the regulations do not match these semantics".

On the contrary, we intend to design a pivotal model that will become the system engineering view, because system engineers play a pivotal role in the product development, since they gather information from every specific business.

6.3. Towards a DALTON-based architecture

In order to design and develop a prototype for an IT component that is able to manage business rules according to the DALTON representation, the following results have to be considered (presented in an unordered list):

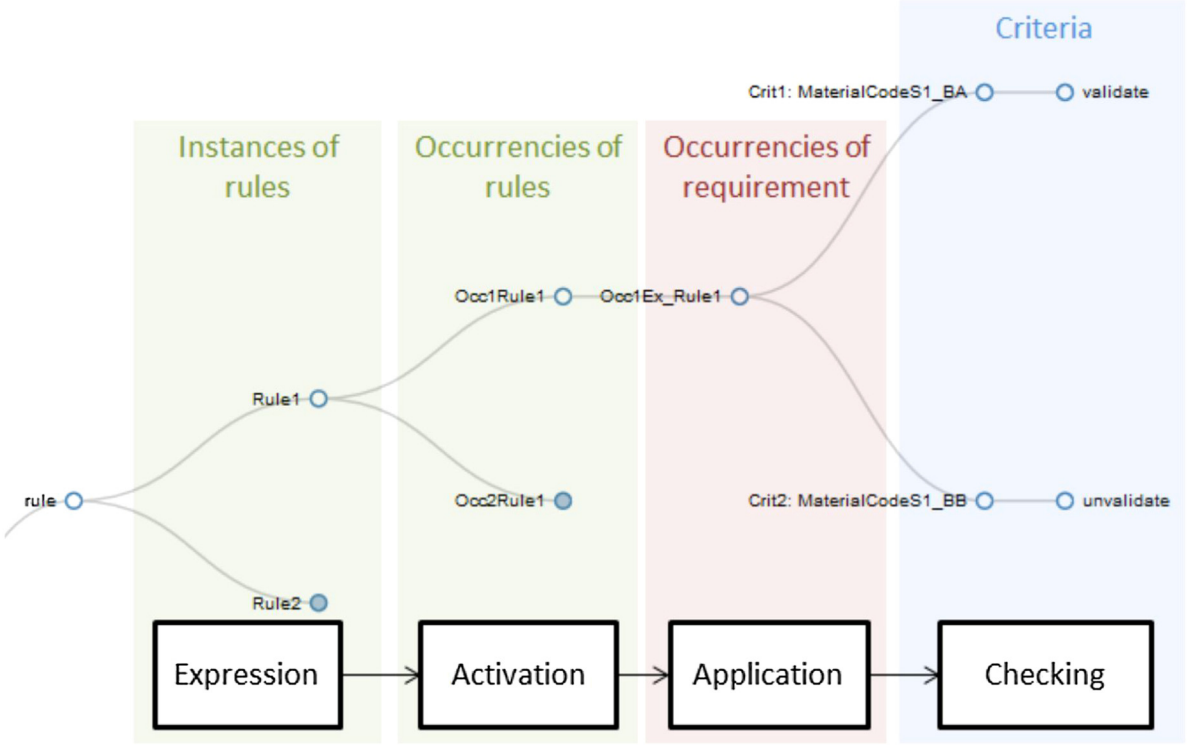

Fig. 5. Rule and requirement graphic presentation.

- genericity: any kind of business rule should be handled by the component;
- data and concept databases have to be separated. Their conceptual consistency may, however, be ensured through semantic mappings or a metamodel;
- a business-oriented formal expression of the rule is mandatory, but all rules must have a unique computational expression;
- a huge amount of data is to be supported (in an order of magnitude of 10^6, considering the usual number of materials operated in a nuclear power plant (pipes, valves, motorpumps, sensors etc.) ;
- a large number of relations are needed to represent the intricacy of the overall data/business rules (the order of magnitude is 10^9 if we consider the usual number of components of nuclear power plants). This intricacy is important because the knowledge is not captured only in the concepts, but rather in the relationships between concepts. For example, Tsuchiya [41] and Nonaka and Takeuchi [32], experts in knowledge management, explain that the information is the structured representation of concepts/data, and knowledge (tacit or explicit) is the result of the interpretation of the information by a semantic framework, for example a human). Moreover, the C-K theory, as explained by Hatchuel et al. [22], formalizes two worlds: the "C" world, the world of concepts, and the "K" world, the world of knowledge. Concepts "become" (are mapped to) knowledge when they are related by properties: "the structures of C and K. Space K contains all established (true) propositions (the available knowledge). Space C contains 'concepts' which are undecidable propositions in K (neither true nor false in K) about partially unknown objects x. Concepts all take the following form: 'There exists some object x, for which a group of properties P1,P2, . . . ,Pk are true in K'." Indeed, a sentence is a network of concepts that DALTON formalizes as a list of triples. A unique identifier (like the IRI) should be affected by any list.
- reasoning must be closed.

6.4. Implementation challenges

Given the "L" and "D" letters of the DALTON acronym ("D" for Data, "L" for Linked), and the billions of objects to manage, concepts and tools from the Linked Data domain [6] are adopted: distributed architectures, URI/IRI for objects identification and location. From a technological point of view, the first decisions lead to the adoption of a high-level programming language to perform closed world reasoning (Python or Java for instance), NoSql database/json/URI for objects management, graphical and dynamic representation of rules and requirements to provide a graphical knowledge modelling and representation of user interface.

The main issue is related to the high number of occurrences (millions of occurrences, as stated above). The storage in itself is not really an issue, because state-of-the-art databases are able to handle such a data size [13]. However, this high number of occurrences, as well as a potential strong intricacy (or conflict) between business rules might have a strong impact on the process time to apply/check business rules over the while system.

This is still a work in progress, and will be further detailed in a future publication.

7. Conclusion

The next generation of enterprise information systems requires richer information, knowledge integration, and automated processes to assist designers and manufacturers with their work. This raises important issues, like how to reconcile the SE process with product-centric information systems available in PLM or BIM. The question of business rules is an illustration of this issue because business rules are the expression of business knowledge; they are applied during the SE process, but they must be verified with regards to the product data. In this paper, we propose a business rules management process made of different stages, and show the importance of generic meta models to support the implementation of automated rule processing. We show that ontological representations are adapted to that, but current tools face some limitations. To provide perspective, we suggest an architecture to implement the proposed conceptualization.

References

[1] B.K. Aichernig, R. Schumi, Property-based testing with FsCheck by deriving properties from business rule model, Software Testing, Verification and Validation Workshops (ICSTW) (2016), doi:http://dx.doi.org/10.1109/ICSTW.2016.24.

[2] M. Bajec, M. Krisper, A methodology and tool support for managing business rules in organisations, Inf. Syst. 30 (6) (2005) 423–443.

[3] I. Bajwa, B. Bordbar, M. Lee, SBVR vs OCL: a comparative analysis of standards, 14th IEEE International Multitopic Conference (INMIC 2011) (2011) 261–266, doi:http://dx.doi.org/10.1109/INMIC.2011.6151485.

[4] T.H. Beach, Y. Rezgui, H. Li, T. Kasim, A rule-based semantic approach for automated regulatory compliance in the construction sector, Expert Syst. Appl. 42 (12) (2015) 5219–5231.

[5] M.L. Bernardi, M. Cimitile, C. Di Francescomarino, F.M. Maggi, Do activity lifecycles affect the validity of a business rule in a business process? Inf. Syst. 62 (2016) 42–59, doi:http://dx.doi.org/10.1016/j.is.2016.06.002.

[6] C.H.T. Bizer, T. Berners-Lee, Linked data-the story so far, Int. J. Semantic Web Inf. Syst. 5 (3) (2009) 1–22, doi:http://dx.doi.org/10.4018/jswis.2009081901.

[7] A. Burattin, F. Maggi, W. Van der Aalst, A. Sperduti, Techniques for a posteriori analysis of declarative processes, 16th IEEE International Enterprise Distributed Object Computing Conference (EDOC2012) (2012), doi:http://dx.doi.org/10.1109/EDOC.2012.15.

[8] S. Chang, D. Castro-Lacouture, F. Dutt, P.P.J. Yang, Framework for evaluating and optimizing algae façades using closed-loop simulation analysis integrated with BIM, Energy Procedia 143 (2017) 237–244, doi:http://dx.doi.org/10.1016/j.egypro.2017.12.677.

[9] B.H. Cheng, J.M. Atlee, Research directions in requirements engineering, 2007 Future of Software Engineering, IEEE Computer Society, 2007, pp. 285–303.

[10] J. Choi, J. Choi, I. Kim, Development of BIM-based evacuation regulation checking system for high-rise and complex buildings, Autom. Constr. 46 (2014) 38–49.

[11] A. Cornière, V. Fortineau, T. Paviot, S. Lamouri, J.L. Goblet, A. Platon, C. Dutertre, Modelling requirements in service to PLM for long lived products in the nuclear field, IFIP International Conference on Advances in Production Management Systems (2014) 650–657, doi:http://dx.doi.org/10.1007/978-3-662-44736-9_79.

[12] J. Dick, E. Hull, K. Jackson, Requirements Engineering, Springer, 2017.

[13] N. Elgendy, A. Elragal, Big data analytics: a literature review paper, in: P. Perner (Ed.), Advances in Data Mining. Applications and Theoretical Aspects. ICDM 2014. Lecture Notes in Computer Science, vol. 8557, Springer, 2014, doi:http://dx.doi.org/10.1007/978-3-319-08976-8_16.

[14] D.M. Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, et al., Naming the pain in requirements engineering, Empirical Softw. Eng. 22 (5) (2017) 2298–2338.

[15] V. Fortineau, A. Cornière, T. Paviot, S. Lamouri, Modeling domain knowledge using inference ontologies: an application to business rules management, IFAC Pap. Online 48 (3) (2015) 573–578, doi:http://dx.doi.org/10.1016/j.ifacol.2015.06.142.

[16] V. Fortineau, T. Paviot, A. Guissé, S. Lamouri, A transformation model to express business rules from natural language to formal execution: an application to nuclear power plant, IFAC Proc. 46 (9) (2013) 1096–1101, doi:http://dx.doi.org/10.3182/20130619-3-RU-3018.00186.

[17] V. Fortineau, T. Paviot, S. Lamouri, Improving the interoperability of industrial information systems with description logic-based models-the state of the art, Comput. Ind. 64 (4) (2013) 363–375, doi:http://dx.doi.org/10.1016/j.compind.2013.01.001.

[18] V. Fortineau, T. Paviot, X. Fiorentini, L. Louis-Sidney, S. Lamouri, Expressing formal rules within ontology-based models using SWRL: an application to the nuclear industry, Int. J. Prod. Lifecycle Manag. 7 (1) (2014) 75–93, doi:http://dx.doi.org/10.1504/IJPLM.2014.065458.

[19] M. Garetti, S. Terzi, N. Bertacci, M. Brianza, Organisational change and knowledge management in PLM implementation, Int. J. Prod. Lifecycle Manag. 1 (1) (2005) 43–51, doi:http://dx.doi.org/10.1504/IJPLM.2005.007344.

[20] T. Gordon, G. Governatori, A. Rotolo, Rules and norms: requirements for rule interchange languages in the legal domain, Lect. Notes Comput. Sci. 5858 (2009) 282–296, doi:http://dx.doi.org/10.1007/978-3-642-04985-9_26.

[21] S. Greenspan, J. Mylopoulos, A. Borgida, On formal requirements modeling languages: RML revisited, Proceedings of the 16th International Conference Onsoftware Engineering (1994) 135–147.
[22] A. Hatchuel, B. Weil, CK design theory: an advanced formulation, Res. Eng. Des. 19 (4) (2009) 181.
[23] D. Hay, K.A. Healy, J. Hall, C. Bachman, J. Breal, J. Funk, J. Healy, et al., Defining Business Rules. What are They Really? Tuch. rep., The Business Rules Group, 2000.
[24] IS0-15288, Systems and Software Engineering -System Life Cycle Processes, International Standard Organization, 2015.
[25] D. Kiritsis, Closed-loop PLM for intelligent products in the era of the internet of things, Comput.-Aided Des. 43 (5) (2011) 479–501, doi:http://dx.doi.org/10.1016/j.cad.2010.03.002.
[26] D. Knuplesch, M. Reichert, A visual language for modeling multiple perspectives of business process compliance rules, Softw. Syst. Model. 16 (3) (2017) 715–736.
[27] K.H. Kung, C.F. Ho, W.H. Hung, C.C. Wu, Organizational adaptation for using PLM systems: group dynamism and management involvement, Ind. Mark. Manag. 44 (2015) 83–97, doi:http://dx.doi.org/10.1016/j.indmarman.2014.04.018.
[28] B. Marconnet, F. Demoly, D. Monticolo, S. Gomes, An assembly oriented design and optimization approach for mechatronic system engineering, Int. J. Simul. Multidiscip. Des. Optim. 8 (A7) (2017), doi:http://dx.doi.org/10.1051/smdo/2016016.
[29] P. Mauborgne, S. Deniaud, E. Levrat, E. Bonjour, J.P. Micaëlli, D. Loise, Operational and system hazard analysis in a safe systems requirement engineering process- application to automotive industry, Saf. Sci. 87 (2016) 256–268, doi:http://dx.doi.org/10.1016/j.ssci.2016.04.011.
[30] G.J. Nalepa, S. Bobek, Rule-based solution for context-aware reasoning on mobile devices, Comput. Sci. Inf. Syst. 11 (1) (2014) 171–193, doi:http://dx.doi.org/10.2298/CSIS130209002N.
[31] P.B.F. Njonko, S. Cardey, P. Greenfield, W. El Abed, RuleCNL: A controlled natural language for business rule specifications, International Workshop on Controlled Natural Language (2014) 66–77, doi:http://dx.doi.org/10.1007/978-3-319-10223-8_7.
[32] I. Nonaka, H. Takeuchi, The Knowledge-Creating Company, Oxford University Press, 1995.
[33] H. Panetto, M. Zdravkovic, R. Jardim-Goncalves, D. Romero, J. Cecil, I. Mezgár, New perspectives for the future interoperable enterprise systems, Comput. Ind. 79 (2016) 47–63, doi:http://dx.doi.org/10.1016/j.compind.2015.08.001.
[34] T. Paviot, V. Cheutet, S. Lamouri, A PLCS framework for PDM/ERP interoperabilty, Int. J. Prod. Lifecycle Manag. 5 (2–4) (2011) 295–313, doi:http://dx.doi.org/10.1504/IJPLM.2011.043182.
[35] M. Pesic, H. Schonenberg, W. vVn der Aalst, DECLARE: full support for looselystructured. Processes, 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC2007) (2007), doi:http://dx.doi.org/10.1109/EDOC.2007.25.
[36] T.A. Pham, N. Le Thanh, A rule-based language for integrating business processes and business rules, The 9th International Web Rule Symposium (RuleML) 1417 (2015) 8.
[37] D. Romero, F. Vernadat, Enterprise information systems state of the art: past, present and future trends, Comput. Ind. 79 (2016) 3–13, doi:http://dx.doi.org/10.1016/j.compind.2016.03.001.
[38] R. Sacks, L. Ma, R. Yosef, A. Borrmann, S. Daum, U. Kattel, Semantic enrichment for building information modeling: procedure for compiling inference rules and operators for complex geometry, J. Comput. Civil Eng. 31 (6) (2017) 04017062.
[39] J. Stark, Product lifecycle management, Product Lifecycle Management 1, Springer, Cham, 2015, pp. 1–29, doi:http://dx.doi.org/10.1007/978-3-319-17440-2_1.
[40] S. Terzi, A. Bouras, D. Dutta, M. Garetti, D. Kiritsis, Product lifecycle management : from its history to its new role, Int. J. Prod. Lifecycle Manag. 4 (4) (2010) 360–389, doi:http://dx.doi.org/10.1504/IJPLM.2010.036489.
[41] S. Tsuchiya, Improving knowledge creation ability trough organizational learning, Proceedings of International Symposium on the Management of Industrial and Corporate Knowledge, ISMICK, Compiègne (1993).
[42] R. Volk, J. Stengel, F. Schultmann, Building information modeling (BIM) for existing building - literature review and future needs, Autom. Constr. 38 (2014) 109–127, doi:http://dx.doi.org/10.1016/j.autcon.2013.10.023.
[43] F. Wortmann, K. Fliichter, Internet of things, Bus. Inf. Syst. Eng. 57 (3) (2015) 221–224.
[44] W. Yao, A. Kumar, ConFLexFlow: integrating flexible clinical pathways into clinical decision support systems using context and rules, Decis. Support Syst. 55 (2) (2013) 499–515, doi:http://dx.doi.org/10.1016/j.dss.2012.10.008.
[45] G. Zayaraz, et al., Concept relation extraction using naive bayes classifier for ontology based. Question answering systems, J. King Saud Univ.-Comput. Inf. Sci. 27 (1) (2015) 13–24, doi:http://dx.doi.org/10.1016/j.jksuci.2014.03.001.
[46] N. Zeichner, J. Berant, I. Dagan, Crowdsourcing inference-rule evaluation July, Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers vol. 2 (2012) 156–160.
[47] J. Zhao, H. Boley, J. Dong, A fuzzy logic-based approach to uncertainty treatment in the rule interchange format: from encoding to extension, Uncertainty Reasoning for the Semantic Web II, Springer, 2013, pp. 197–216.