CONFERE'18
5 ET 6 JUILLET 2018, BUDAPEST, HONGRIE.

PROPOSITION OF A DESIGN RULES FRAMEWORK

Simon Debord[1], Frederic Segonds[1], Romain Pinquié[2], Philippe Veron[2], Nicolas Croué[3]
[1] Arts et Métiers ParisTech, LCPI
151 boulevard de l'Hôpital
75013 Paris, France
Simon.debord@ensam.eu

[2] Arts et Métiers ParisTech, LISPEN
2 Cours des Arts et Métiers,
13617 Aix-en-Provence

[3] Sogeti High-Tech
3 Chemin de Laporte,
31300 Toulouse

Keywords: Design Rules, Computer Aided Design, Knowledge Management, Ontology.

ABSTRACT.

[Context] The design activity relies on rules preventing errors. Design errors are choices that make certain designs "not allowed" or inappropriate for their intended use. Design rules arise from all stages of a product life cycle. For instance, when a maintenance operator observes a systematic failure, his observation should lead to a new design rule. Such a design strategy is known as Design for X (DFX or DfX). Nevertheless, the ever-increasing complexity of engineered systems and projects lead to a bewildering array of rules among which a subset is only of interest for a specific designer. The subset of design rules that is of interest for a given designer depends on the design context. For instance, a mechanical designer designing the ribs in aluminum of an aircraft wing using the part design workbench of the CATIA software is interested by the rules that belong to the design context: DC = {domain = mechanics; role = designer; part = rib; material = aluminum; system = aircraft; sub-system = wing; software = CATIA; workbench = Part design}. **[Problem]** So far, design rules are stored in documents or in the head of subject-matter experts. Design rules are therefore difficult to collect, validate, store, retrieve, use, verify, and maintain. **[Proposition]** We propose a design rules framework that enables a subject-matter expert to formally encode design rules in a shared and structured knowledge base. Embedded in a Computer Aided Design (CAD) environment, our ontology will extend traditional CAD capabilities as it will, for instance, ease: 1) the retrieval of rules according to a particular design context, and 2) the recommendation of design rules while a design activity is being performed **[Future work]** Our Design Rules Framework will be improved before integrating it in a CAD environment to provide designers with an extended CAD environment.

1 INTRODUCTION

[Context] Designing a product is a knowledge-intensive activity. Indeed, for numerous reasons – e.g. employee turnover, product complexity, extended enterprise, etc. – subject-matter experts must capitalize and share design rules to prevent designers to propose undesirable solutions. From our experience, we know that large companies usually store an extensive number of design rules in various documents. Each document is several hundred pages containing an unstructured blend of texts, equations, tables, sketches, and charts. Although companies tend to add new rules for every design error, there is no debate about the necessity to adopt a “lean engineering” strategy by using such a knowledge base with parsimony during the design process.

[Problem] Formerly, when companies used to store a small number of design rules and share them among tens of designers in a unique design office, documents were adequate. However, today, this approach is not practicable anymore. First, the large number of experts and the geographically dispersed teams make the collection of requirements in documents cumbersome. That is all the more true at a time when design rules are stored in multiple repositories: documents, databases, expert’s head, etc. Once stored in a repository, design rules must be validated “Are we defining the right design rule?”, verified “Are we defining the design rule right?”, and managed for decades. Documents are not easy to manage in configuration. Finally, when a designer wants to make sure that its solution satisfies all design rules of interest, it has no other alternative than to spend a large amount of time to retrieve the subset of design rules for its own design context. For instance, a mechanical designer designing a rib in aluminum for an aircraft wing using the CATIA CAD software wants to easily retrieve the design rules belonging to the design context: DC = {domain = mechanics; role = designer; part = rib; manufacturing process = milling; material = aluminum; system = aircraft; sub-system = wing; software = CATIA; activity = CAD; workbench = Part design}. For all these reasons, there is a need for a collaborative software framework that facilitates the collection, storing, use, and maintenance of design rules. Based on an ontology of design knowledge, such a tool – standalone or integrated in a CAD system – will pave the way to provide designers with and extended design environment that could provide new services, such as:

- Collect design rules in a Design for X (DfX) socio-technical environments,
- Automate design routines (e.g. basic CAD modelling operations, semantic annotations, etc.),
- Recommend design rules to a designer while a design activity is being performed,
- Verify that a solution satisfies a set of design rules and provide coverage metrics,
- Notify new rules for continuous learning.

[Proposal] In this paper, we introduce our design rules framework. First, we will describe our design rules framework from an operational perspective, that is, the stakeholders and the services the framework provide to them. Second, we propose various properties that will enable a knowledge engineer to tag design rules so as to facilitate their exploitation as the volume of rules in the knowledge base is significant. To conclude, we will give the gist of the ontology that supports our design rules framework and illustrate our proposition with a use case inspired from current design practices in the aerospace industry.

2 RELATED WORK

The literature review starts with a clarification of what we consider as a “design rule” before to sum up the different propositions for checking designs rules on CAD models. Finally, we review the use of ontologies in engineering design.

2.1 Design rules

The term “design rule” is a suitcase-like word that people use to conceal the complexity of different things. In the design rule checker proposed by Garcia et al. [1] and Radhakrishnan et al. [2], the design rules are limited to basic inequalities between primitive geometric entities for sheet-metal engineering. Huang et al. [3] focus on the automatic detection of 3D CAD model errors for NC programming. To detect the design errors, they propose 7 heuristic rules design rules under the form of algorithms that make use of the geometry and the topology of the CAD model. As mentioned by Sharka et al. [4], parametric CAD modelers such as CATIA and its knowledgeware modules (advisor, expert, …) offers numerous way to model, automate and check design rules using knowledge modelling capabilities (parameters, formulas, design tables, rules and checks, power copy, template, reactions, scripts, etc.). In the context of knowledge-based engineering, Calkins et al. [5] distinguish four categories of design rules: (1) Heuristics – experimental rules-of-thumb and best practices; (2) Empirical rules – rules developed from experimental data, that is, based on curve-fitted expressions; (3) Legislated constraints – engineering standards or laws; and (4) Laws of physics – analytical or numerical model derived from scientific principles.

Although most research studies focus on the modelling and checking of design rules limited to geometrical and topological properties, when we read design guidelines, we observe that a significant number of rules do not deal with such properties – e.g.:

- *The junction must be designed to minimize weight.*
- *Fasteners to be used at the longitudinal joint shall be lockbolt.*
- *Circumferential ply drop off under the junction are forbidden.*
- *When a sandwich structure is terminated, the panel edge must be sealed.*
- *Fasteners in a joint shall have the same pitch, diameter, bolt type, hole tolerance and material.*
- *CPC shall be applied onto the surface of aluminum alloy.*

Consequently, we claim that the originality of our design rule frawemork is to deal with any kind (geometric or not) of design rule represented in any form (text, sketch, table, equation, etc.).

2.2 Checking design rules

Design rules arise from the consideration of a product potential issues during its whole life cycle. It includes how a product is manufactured, assembled, installed, used, maintained, and recycled. By capitalizing feedbacks for each phase of a product's life cycle, companies maintain a set of design rules that prevent invalid design. To this end, industry and research have developed many tools and methods [2, 3] to help designers to control the impact of their design decisions on the product's life cycle. These tools are referred to as Design for X or DfX (sometimes Design for excellence), where X stands for one or many design considerations of a product. Among these considerations, manufacturing (DfM) and assembly (DfA) prevail, but quality, cost, environment, maintenance, reliability and end-of-life are well-documented too. Each technique provides the designer with guidelines including design rules. For instance, a DfM rule for a sheet metal part could be *"Inner bend radius should be equal to material thickness"*.

The prominent place of Computer Aided Design tools in the design process leads to the development of DfX software that embed design rules and ensure the validity of CAD models. Several commercial tools are available either as standalone applications or as a plug-in. Among these tools, DFM Pro from GeometricPLM [8] can be integrated in Solidworks, PTC CREO Parametric and Siemens NX. Checkmate is directly included in Siemens NX [9]. DFMA from Boothroyd Dewhurst [10] and aPriori [11] are standalone applications. The CAD model is analyzed by the tool, the geometric features are recognized, and the software assesses the model with regard to a set of rules embedded in the software. These tools include a design rules database and use automated features recognition techniques (AFR). For example, for the sheet metal design rule of the previous paragraph, the software recognizes the feature *"Inner bend radius"* and the parameters *"radius value"* and *"material thickness"* to check the rule. Many methods exist for AFR and will not be more detailed in this article. Babic et al. [12] propose an extensive review of the different techniques. In such systems, the design rules are checked *a-posteriori*. Such a strategy prevents the designers to deliver invalid design and to continuously learn design rules. The capabilities of such tool largely depends on the number and the quality of the set of rules stored in the knowledge base which is not expandable. Indeed, these systems do not offer the possibility to modify – i.e. add, modify or remove rules – the knowledge base. Efforts have been made to develop tools that enable designers to author and check design rules in DFM systems, but too many studies focus on the development of the umpteenth AFR technique and/or design rules checking technique. For example, Radhakrishnan et al. [2] propose an algorithm that reasons on the geometry of sheet metal parts to recognize basic features (holes-slots-bends) and their associated parameters (e.g. distances) and check the design rules by constraining these parameters. In this proposition, the design rules are directly encoded in the algorithm and require advanced programming skills to update them. Furthermore, the design rules are limited to distances between basic geometric entities. Similarly, Hariya et al. [13] describe a technique to check design rules related to geometric features of injection molded parts. The input of their system is a CAD model and a defined thickness. Each specific feature (rib-boss-hole-fillet) are extracted at a time. Then the parameters of each feature are compared to the design rules stored in a database.

As an alternative to AFR techniques and logic programming, more recently, Rangarajan et al. [14] created a DfM platform that enables the authoring and the verification of design rules on CAD models. A semantic model of CAD sheet metal features and attributes is created using an English-like semantic language named Semantic Application Design Language or SADL. The design rules are also authored in SADL. Then, the system creates a semantic instance by extracting the sheet metal features from the CAD model using a Java program and directly maps the semantic instance in SADL. The rules can finally be executed against the geometric features and the result is displayed to the user. The interesting contribution of this paper is the capability to specify new rules in a natural language that can be directly executed on a semantic representation of a CAD model independently from any CAD software.

As mentioned in the previous section, these approaches are limited to the verification of DfM rules based on geometric and topological features and do not deal with less formal design rules. Additionally, they focus on the verification of the quality of CAD features a-posteriori without supporting other capacities, such as the recommendation of design rules in real time while designing, the automation of design routines and the continuous learning of design rules.

2.3 Engineering ontologies

An ontology is a formal representation of the knowledge (concepts and relationships) of a given domain of interest. In [16] Tecuci defines an ontology as:

> *"A hierarchical representation of the objects from the application domain. It includes both descriptions of the different types of objects and descriptions of individual objects together with the properties of each object and the relationships between objects."*

Liu and Lim, [16] review the recent developments in using ontologies to manage design knowledge. Three major application fields stand out. First, an ontology can be used as a knowledge base to store concepts and relationships. Second, by coupling an ontology with a reasoner, it is possible to infer new knowledge from the initial knowledge base. The inference of new knowledge serves different purposes. For example, Bock et al. [17] combined modeling languages and ontologies to model complex products. Catalano et al. [18] created a Product Design ontology that facilitates information exchange and collaboration in the product design process. Li and Yoo [19] created a web-service based on semantic web technologies to provide design participants with common engineering data, such as engineering ontologies for product description and design rules. Designers encode the design rules in a understandable semantic query language that enable a user to reason on the product's features stored in the ontology. Research works have shown that ontologies is also effective to ensure the interoperability of design information. For instance, Dartigues et al. [20] developed a CAD feature ontology to link data between CAD and CAPP (computer-aided process planning) software. Several studies have demonstrated the advantages of using ontologies to manage the product diversity (Pinquié *et al.* [21]). For example, Yang et al. [22] proposed a product configuration system based on a meta-ontology for modelling the configuration of products. The refinement of this ontology gives a customized product model (car, computer, cellphone, etc.). The input of customer requirements as inference rules to this latter ontology has the effect of populating the ontology with instances that correspond to the product configuration matching the customer requirements.

The literature shows that using ontologies is common place in the engineering design domain, however they are not used for reasoning on design context and recommending design rules. Similarly, to context-aware application, we assume that the coupling of a design context ontology with the semantic web technologies reasoning capabilities facilitates the exploitation of a very large set of design rules.

3 A SEMANTIC DESIGN RULE FRAMEWORK

In this section, we present our design rule framework from three perspectives. First, we detail the operational view, that is, the stakeholders and the services the framework shall provide to the stakeholders. Second, as the set of design rules in a company is very large, we propose a list of properties to navigate in the knowledge base. Finally, we give the gist of our design rule framework and the underlying ontology.

3.1 Design rules framework: a user-centered description

The product development process involves various stakeholders. Further in this article, we propose a *'design rules framework'* that aims at extending the current capabilities of a CAD environment that includes various tools for geometric modelling, FEA analysis, CAM, etc. Figure 1 shows that our design rules framework has two stakeholders.

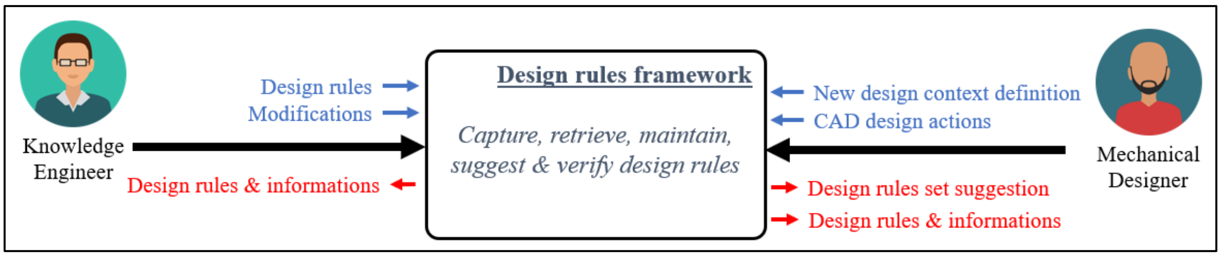

Knowledge Engineer
Design rules
Modifications
Design rules & informations
Design rules framework
Capture, retrieve, maintain, suggest & verify design rules
New design context definition
CAD design actions
Design rules set suggestion
Design rules & informations
Mechanical Designer

Figure 1 - Context diagram showing the interaction between the Design Rules Framework and the users

These two profiles work for a large industrial company that designs complex products. The first stakeholder is a knowledge engineer who is in charge of the implementation and the management of the engineering design knowledge, especially the design rules. As a knowledge expert, we suppose he is an experienced engineer who acquired multidisciplinary knowledge that ease the communication with his colleagues who suggest new design rules. He knows most of the design rules, best practices and standards of his company. He participates to the capitalization of design knowledge in collaboration with the subject-matter experts. His main missions are to develop and manage the design rules in the framework. The second user profile is a designer who must provide design solutions that complies with numerous design rules. Our design rules framework shall provide services to both user profiles as detailed hereafter.

Knowledge Engineer –

- **[RULES CAPTURE]** The first capability is the capture and storage of the design rules. Design rules are stored in several sources (handbooks, documents, databases, human memory…). After identifying the design rules in a traditional source (e.g. a company design guide), the knowledge engineer adds them in the *'design rules framework'*. The UI mock-up in Figure 2 shows that the knowledge engineer identifies the concepts in the design rule and links them with existing engineering concepts and instances of the ontology that underlies our framework. For example, for the rule: *"the minimum pocket thickness T should not be less than 1.3 mm for aluminum and 0.8 mm for hard alloys"*, the knowledge engineer links the rule with the concepts: milling, pocket, aluminum and hard alloys. In addition, one can also identify the rules according to its place in the taxonomy.
- **[RULES RETRIEVAL]** In operation, the knowledge base will store thousands of design rules. It is therefore necessary to provide retrieval capabilities to navigate in the knowledge base. Since the rule is related to several concepts, we can find rules by filtering the concepts. In Figure 3, to find the latter rule, the user queries the concepts *pocket* and/or *aluminum* to obtain a list of rule instances including these two concepts.

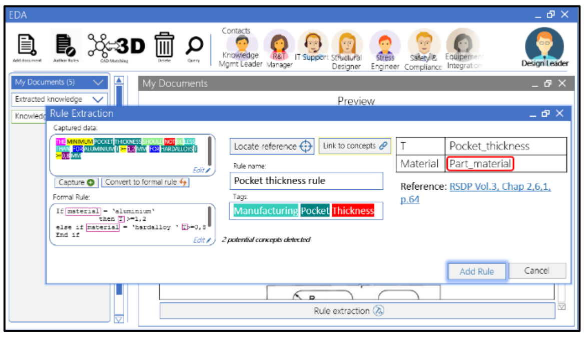


Figure 2 - User interface for design rules capture

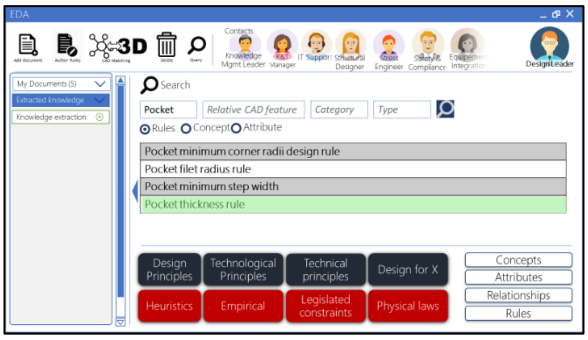


Figure 3 - User interface for design rules query

- **[MAINTENANCE]** At any time, the knowledge engineer can maintain – i.e. create, update, delete – a rule, release a new version or an alternative rule. The changes made are reported so as to provide a follow-up through time. The Figure 4 shows the evolution of the ontological representation of a design rule. In this example Rule 1.1 initially relates to three concepts: *Pocket* (respectively sub-concept of *Milling*), *Hard-alloys* and *Aluminum* (both sub-concepts of *Material*). A revision could be to state that this rule is also applicable for *Titanium* concept. The revised rule is now called *Rule 1.2*. Finally, alternative design rules can be created like *Rule 1.21*. It is a variant of *Rule 1.2* but does not replace it.

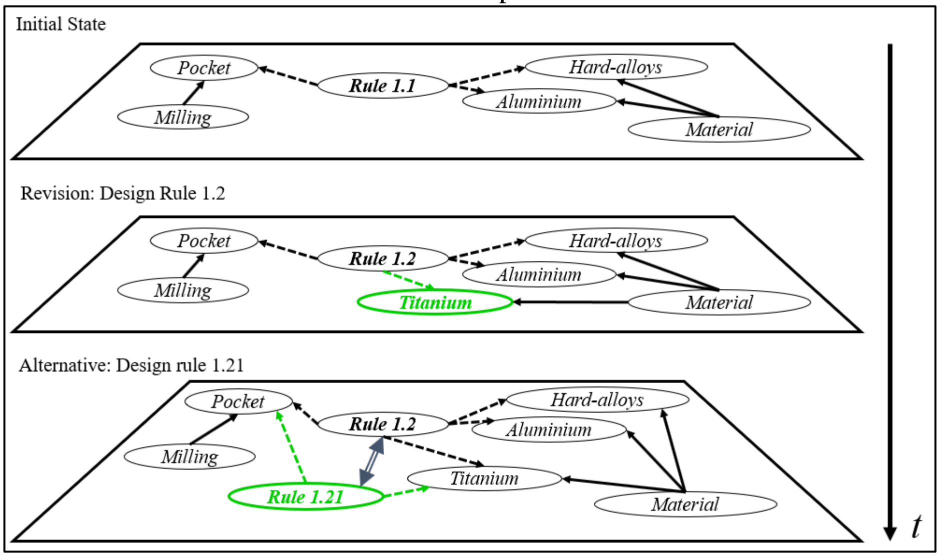


Figure 4 - Evolution of design rules through time

Mechanical Designer –

- **[DESIGN RULES SUGGESTION]** When designing a mechanical part, the mechanical designer specifies his design context. He selects the type of part he is designing, the components connected to it, the material, the manufacturing process and his collaborators. By selecting each one of the previous items, the *design rules framework* automatically suggests and adjusts the possibility for the other items. Once all the fields complete, Figure 5 shows that the framework provides him with an adequate design rules subset, and information related to his design context.
- **[DESIGN RULES VERIFICATION]** While modelling a part in the CAD environment, design rules are suggested dynamically to the designer. Figure 6 shows that, in a given design situation, several rules may be applicable. It is the designer's choice whether to apply or not the proposed rules. Such "*real-time*" design assistance helps the designer to continuously learn and acquire design knowledge efficiently avoiding rework.

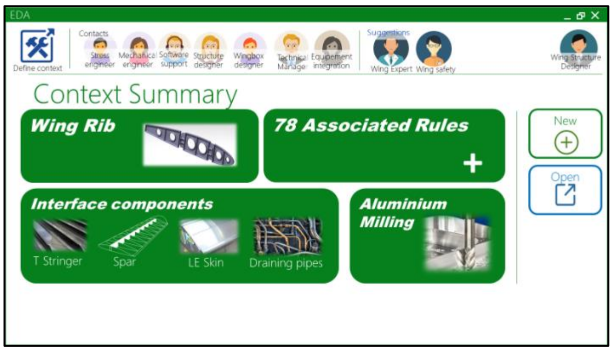
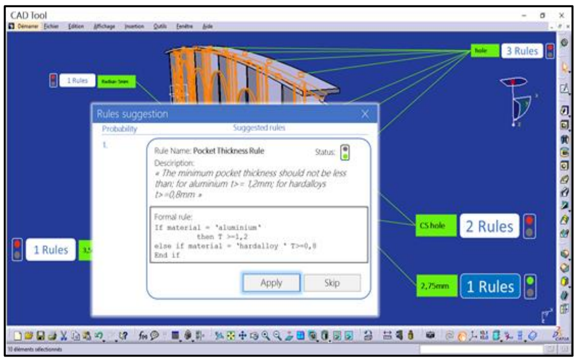


Figure 5 - Design context summary



1.

Rule Name: Pocket Thickness Rule Status:

Description:

« The minimum pocket thickness should not be less than: for aluminium t>= 1,2mm; for hardalloys t>=0,8mm »

Formal rule:

```
If material = 'aluminium'
          then T >=1,2
else if material = 'hardalloy ' T>=0,8
End if
```

Figure 6 - Design rules suggestion in CAD environment

3.2 A taxonomy of design rules

To ease the exploitation – i.e. retrieval, management, etc. – of design rules, we propose a set of properties that will stand as entry points to navigate in a very large set of rules. The following enumeration results from users' experience.

- **The DfX origin:** it is the life cycle phase from which the design rule is required – e.g. needs analysis, design, manufacturing, assembly, transport, use, maintenance, recycling.
- **Topics: a** designer wants to focus on the rules belonging to its domain of expertise – e.g. mechanics, electronics, quality, safety, etc.
- **The application domain:** a designer wants to easily distinguish the rules that concern a tool-based modelling activity – e.g. CAD modelling rules – from the ones related to an engineering activity – e.g. the type of bearing according to loads conditions.
- **Objective:** a designer originally wants to realize a function and needs the related design rules – e.g. the design rules to fulfill the function "*to bond two composite plates*".
- **Authority:** designer will concentrate on unnegotiable rules before looking at nice-to-have recommendations – e.g. a design rule that must be satisfied to certify a system.
- **Granularity**: a designer wants to filter design rules that constrain the design of parts from the ones that constrain the design of assemblies.
- **Feature granularity:** for a rule concerning a single part (i.e. a rule with a granularity "Part"), a designer can filter the "Intra-feature" rules – e.g. the rules limited to a hole – from "Inter-features" rules – e.g. rules that constrain the position of hole from the edges of a plate.

This initial list of properties that serves to tag design rules will be gradually completed in the future while studying new design manuals.

3.3 Design rules framework

[PURPOSE] The aim of our *Design Rules Framework* is to lay the first stone for a broader research work that intends to bridge the gap between design rules that originates from various sources and their use in computer-aided design environments. Currently, the various sources of design rules (documents, databases or as tacit knowledge) and the exponential growth of design data in large companies make difficult for designers to retrieve the right information at the right time. As a first step, we propose a semantic framework to not only capture and classify the design rules, but also facilitate their retrieval, recommend them to a designer according to a given design context, and check design solutions. Ontology-based knowledge representation allows to capture, model, share, reuse, and maintain design knowledge. To create our ontology we used Protégé [23] which relies on the language OWL.

Our *Design Rules Framework* offers the possibility to capture design rules, therefore a part of the ontological representation is dedicated to the storage of design rules defined by the knowledge engineer. In the part (1) *Design rules base* of Figure 7, the rules are stored according to the taxonomy presented in Section 3.2. One of the main advantages of our approach is the retrieval and the recommendation according to a given design context DC, such as DC = {role; organization; part; manufacturing process; material; system; sub-system; software; activity}, where each item is likely to bring design rules or act on their application. Consequently, in our framework, Figure 7 shows that the design rules are linked to design context elements by using properties. For instance, a design rule that specifies a minimum radius on a machined component should refer to the appropriate manufacturing process. In the *design context knowledge* modelling of part (2) Figure 7, the design context items are classes containing instances, and properties define the relationships between them. When designing a new component, a designer defines his current design context by selecting the elements that correspond to his situation. Technically, he instantiates a separate class, called *User Current Design Context* (3) or CDC, with the elements that match his so-called context. Then, coupling the reasoning capabilities of OWL with inference rules enables the attribution of the design rules of interest (4) to a design rules set (5) that matches the designer's need.

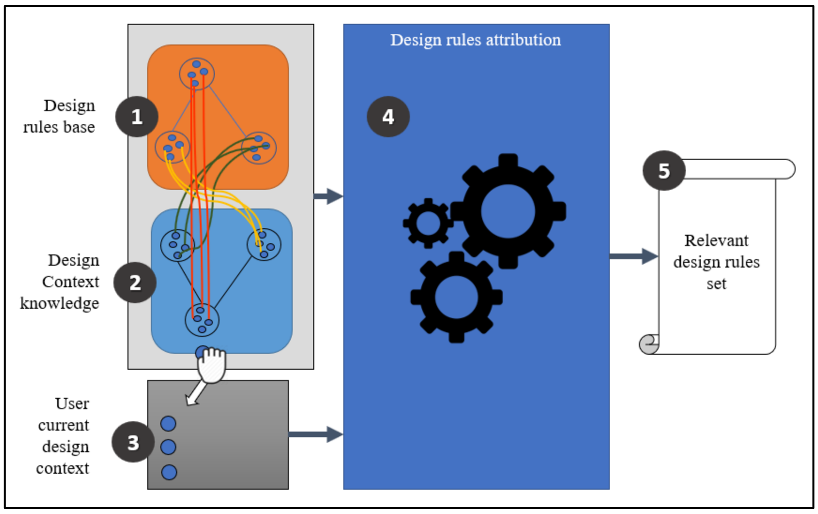


Figure 7 - Architecture of the Design Rules Framework

Note: In this part we assume that any design rule related to elements of the CDC should be part of the design rules set.

The so-called inference rules (not to be confused with *design rules*) are written with the Semantic Web Rule Language (SWRL). In our study, the role of inference rules is to:
1. Structure the intrinsic knowledge and constraints of the design context;
2. Organize the taxonomy. If a design rule has a specific attribute, then it should belong to a particular category;
3. Prevent incoherent CDC definition;
4. Instantiate design rules in the DR set.

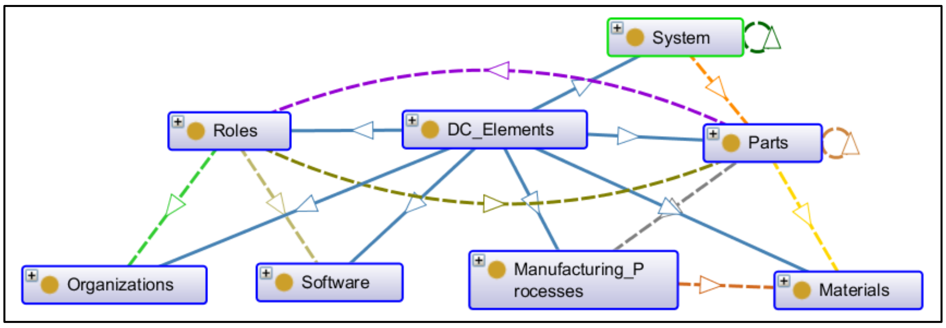


Figure 8 - Classes hierarchy of the design context knowledge modelling. Solid lines in blue indicate the sub-classes and dashed lines correspond to the properties. (PROTÉGÉ screenshot)

Figure 8 shows that the *design context knowledge* modelling includes classes and subclasses representing the elements to be considered during design. The blue arrows represent the class-subclass relationships (called "*is_a*" properties). For example, Materials is a sub-class of DC_Elements (for design context elements). The doted arrows describe the relationships between classes. For example, the orange arrow between Manufactuting_processes and Materials corresponds to the property: Consumes. This representation is a generic ontology or meta-ontology. It has to be enriched with domain knowledge to operates the *Design Rules Framework*. The seven classes constitute the backbone of the framework. Domain experts will collaborate with a knowledge engineer to add subclasses to each one of the current classes to represents its own design context knowledge.

The Design Rules Base is organized according to the properties presented Section 3.2. We consider Design Rules Base as a single class. Instances of design rules compose this class. In Section 3.1, the attributes are *data properties* [1] that can take several values as illustrated Figure 9. Hence, a design rule is an instance (or individual) of the class Design Rules Base and this rule attributes characterizes it. This facilitates the retrieval of design rules to specific properties.

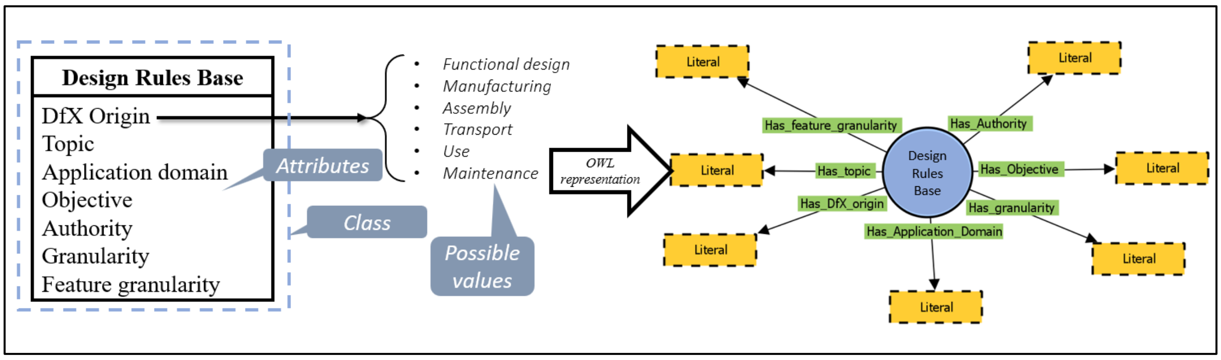


Figure 9 - Design Rules Base class description (left) and an illustration of its OWL language modelisation.

The relationships that link *design rules* stored under the Design Rules Base class with design context elements (i.e. instances of DC_elements subclasses) is modelled with *object properties*[2]. Figure 10 shows an example of the relationship of a design rule DR2 (assembly rule between a rib and a spar in an aircraft wing) with two elements of the design context: Spar and Rib_numero_1. This relationship is modelled with the property Rule_for. Such property allows the attribution of *design rules* to design context elements. Figure 10 also represents the different levels of detail of the design rules framework. The first layer corresponds to the generic model of the design context knowledge that needs to be completed with domain specific classes and instances as show layer 2 and 3. To get a set of design rules matching its need, the user instantiates a class called Current Design Context with instances from DC_elements. Then, thanks to SWRL rules, the inference engine integrated in *Protégé* executes SWRL rules to instantiate *design rules* related to DC_elements to the

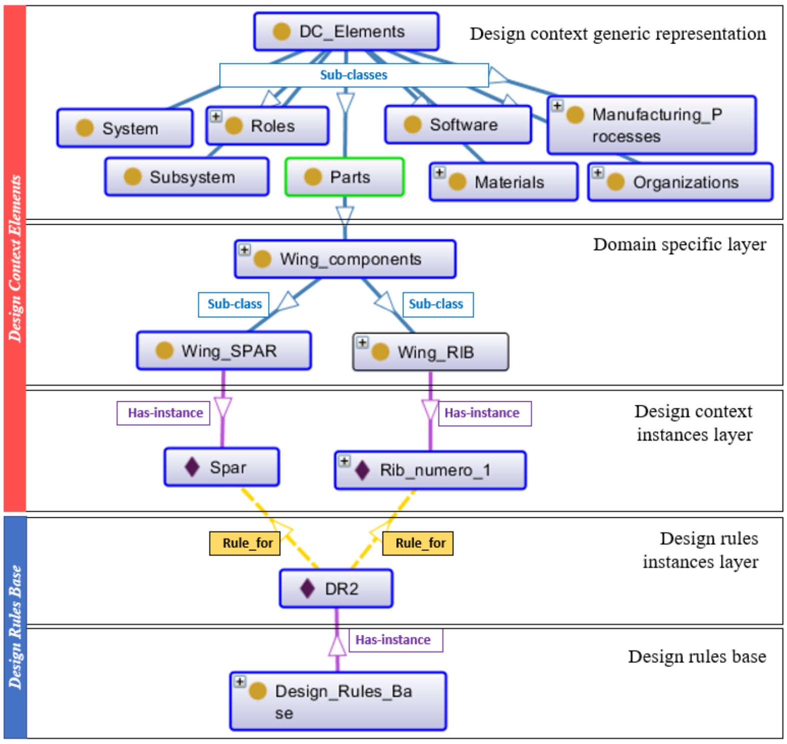


Figure 10 – Relationship between the design rule base and design context elements. Purple diamonds represent instances and yellow circles represents classes. (PROTÉGÉ screenshot)

[1] Data properties describe relationships between an individual (i.e. an instance) and data value (like an integer, a string, a Boolean, etc.)
[2] Object properties are relationships between two individuals.

Current_Design_Rules_Set class. The SWRL rule figure 11 states that any *design rule* related to an instance of the Current Design Context belongs to the Current_Design_Rules_Set class.

```
CURRENT_DESIGN_CONTEXT(?z) ∧ Rule_for(?r,?z) ∧
Design_Rules_Base(?r)  →  CURRENT_DESIGN_RULES_SET(?r)
```

Figure 11 - SWRL Rule for *Design Rules Set* populating.

Any change in the Current Design Context results in an adjustment of the Current_Design_Rules_Set. Therefore, when a designer modifies his design context, the *Design Rules Framework* automatically suggests new *design rules*. Besides, SWRL rules serves a knowledge engineer to express constraints specific to the *Current Design Context*. For example, if the designer chooses a specific manufacturing process that uses only a restricted range of materials, he shouldn't be allowed to select unauthorized material instances.

4 CONCLUSION & FUTURE WORK

With the increasing amount of design rules, it is more and more difficult for designers to find which rule to apply at the right time. In this paper, we propose a Design *Rules Framework* that aims to ease the capture, retrieval and suggestion of design rules according to a given design context. This work paves the way for a broader research work that intends to develop a tool extending the current CAD capabilities. We use the OWL ontology language to model design context knowledge in the form of an ontology including design concepts, relationships and instances. We introduce design rules properties that facilitates the retrieval of design rules. SWRL rules are used to distil a relevant set of design rules that matches the context of a designer at a given moment.
As a future work, we intend to assess our proposition with a use case based on industrial rules. Another goal will be to verify design rules in a CAD environment including various kinds of models. This implies the need to formally represent design rules so that a machine can interpret them.

REFERENCES

[1] L. E. R. García, A. Garcia, and J. Bateman, "An ontology-based feature recognition and design rule checker for engineering," *CEUR Workshop Proc.*, vol. 809, pp. 48–59, 2011.
[2] R. Radhakrishnan, A. Amsalu, M. Kamran, and B. O. Nnaji, "Design Rule Checker for Sheet Metal Components Using Medial Axis Transformation and Geometric Reasoning," no. 3, pp. 179–189, 1996.
[3] B. Huang, C. Xu, R. Huang, and S. Zhang, "An automatic 3D CAD model errors detection method of aircraft structural part for NC machining," *J. Comput. Des. Eng.*, vol. 2, no. 4, pp. 253–260, 2015.
[4] W. Skarka, "Application of MOKA methodology in generative model creation using CATIA," *Eng. Appl. Artif. Intell.*, vol. 20, no. 5, pp. 677–690, 2007.
[5] D. E. . Calkins Egging N.b Scholz C.b, "Knowledge-Based Engineering (KBE) Design Methodology at the Undergraduate and Graduate Levels," *Int. J. Eng. Educ.*, vol. 16, no. 1, pp. 21–38, 2000.
[6] J. Bralla, "Design for Manufacturability Handbook," *McGraw-Hill Prof.*, p. 1368, 1998.
[7] J. G. Bralla, *Design for Excellence*. 1996.
[8] "Dfmpro." [Online]. Available: https://dfmpro.geometricglobal.com/.
[9] "Siemens NX Checkmate." [Online]. Available: https://www.plm.automation.siemens.com/en_us/Images/2504_tcm1023-11882.pdf.
[10] B. Dewhurst, "DFMA." [Online]. Available: https://www.dfma.com/.
[11] "aPriori." [Online]. Available: https://www.apriori.com/.
[12] B. Bojan, N. Nenad, and M. Zoran, "A review of automated feature recognition with rule-based pattern recognition," no. April, 2008.
[13] M. Hariya, N. Nonaka, Y. Shimizu, K. Konishi, and T. Iwasaka, "Technique for checking design rules for three-dimensional CAD data," *Proc. - 2010 3rd IEEE Int. Conf. Comput. Sci. Inf. Technol. ICCSIT 2010*, vol. 1, pp. 296–300, 2010.
[14] A. . Rangarajan, P. . Radhakrishnan, A. . Moitra, A. . Crapo, and D. . Robinson, "Manufacturability analysis and design feedback system developed using semantic framework," in *Proceedings of the ASME Design Engineering Technical Conference*, 2013, vol. 4.
[15] G. Tecuci, *Building Cognitive Assistants*. 2016.
[16] Y. Liu and S. C. J. Lim, "Using Ontology for Design Information and Knowledge Management : A Critical Review," pp. 427–433, 2011.

[17] C. Bock, X. Zha, H. W. Suh, and J. H. Lee, “Ontological product modeling for collaborative design,” *Adv. Eng. Informatics*, vol. 24, no. 4, pp. 510–524, 2010.
[18] C. E. Catalano, E. Camossi, R. Ferrandes, V. Cheutet, and N. Sevilmis, “A product design ontology for enhancing shape processing in design workflows,” *J. Intell. Manuf.*, vol. 20, no. 5, pp. 553–567, 2009.
[19] X. Li and S. B. Yoo, “Integrity validation in semantic engineering design environment,” *Comput. Ind.*, vol. 62, no. 3, pp. 281–291, 2011.
[20] C. Dartigues, P. Ghodous, M. Gruninger, D. Pallez, and R. Sriram, “CAD/CAPP Integration using Feature Ontology,” 2007.
[21] R. Pinquié, “Synthèse collaborative d’exigences en ingénierie de produits.,” 2016.
[22] D. Yang, M. Dong, and R. Miao, “Development of a product configuration system with an ontology-based approach,” *CAD Comput. Aided Des.*, vol. 40, no. 8, pp. 863–878, 2008.
[23] Stanford University, “Protégé.” 2007.

Contact principal : Simon DEBORD

Coordonnées : simon.debord@ensam.eu