A Property Graph Data Model for a Context-Aware Design Assistant

Romain Pinquié[1](✉), Philippe Véron[2], Frédéric Segonds[3], and Thomas Zynda[4]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, Grenoble, France
romain.pinquie@grenoble-inp.fr
[2] LISPEN, Arts et Métiers ParisTech, Aix-en-Provence, France
philippe.veron@ensam.eu
[3] LCPI, Arts et Métiers ParisTech, Paris, France
frederic.segonds@ensam.eu
[4] Capgemini, Toulouse, France
thomas.zynda@capgemini.com

Abstract. [Context] The design of a product requires to satisfy a large number of design rules so as to avoid design errors. [Problem] Although there are numerous technological alternatives for managing knowledge, design departments continue to store design rules in nearly unusable documents. Indeed, existing propositions based on basic information retrieval techniques applied to unstructured engineering documents do not provide good results. Conversely, the development and management of structured ontologies are too laborious. [Proposition] We propose a property graph data model that paves the way to a context-aware design assistant. The property graph data model is a graph-oriented data structure that enables us to formally define a design context as a consolidated set of five sub-contexts: social, semantic, engineering, operational IT, and traceability. [Future work] Connected to or embedded in a Computer Aided Design (CAD) environment, our context-aware design assistant will extend traditional CAD capabilities as it could, for instance, ease: (1) the retrieval of rules according to a particular design context, (2) the recommendation of design rules while a design activity is being performed, (3) the verification of design solutions, (4) the automation of design routines, etc.

Keywords: Design rule · Graph modelling · Knowledge management · Context-awareness · Cognitive assistant

1 Introduction

[Context] Designing a product is a knowledge-intensive activity. Thus, to prevent design errors, that is, choices that make certain designs "not allowed" or inappropriate for their intended use, design departments prescribe design rules. A design rule is a prescriptive statement – often an unstructured blend of text and graphical objects (equation, table, chart, sketch, etc.) – aiming at assisting deployed designers for the achievement of a valid design, in compliance with best practices, applicable regulations, and Design for X

C. Fortin et al. (Eds.): PLM 2019, IFIP AICT 565, pp. 181–190, 2019.
https://doi.org/10.1007/978-3-030-42250-9_17

constraints. To store the bewildering array of design rules, companies use unstructured documents – mainly in PDF format – which are over tens or hundreds of pages.

[Problem] Formerly, when companies used to store tens of design rules and share them among tens of designers in a unique design office, documents remained adequate. However, today, for various reasons, a document-based approach is not suitable anymore. First, the large number of experts and the geographically dispersed teams make the collection of design rules in documents cumbersome. That is all the more true at a time when design rules are stored in multiple repositories: documents, databases, models, 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 but change management using documents is laborious. Finally, when a designer must provide a design free of errors, it has no other alternative than to go through the "Big Data" and spend a large amount of time to retrieve the subset of design rules that matches its own design context. Because of the aforementioned reasons and the recent Renaissance of the Model-Based approach encouraging a full model-based engineering, we state that unstructured documents can no longer serve as an efficient solution for storing design rules. There is a need for a context-aware design assistant that aids designers in the collection, organisation, retrieval, use, and modification of design rules.

[Proposal] The main contribution of this paper is twofold. First, we will analyse the operational view of the context-aware design assistant to identify the stakeholders and the services the assistant shall provide to them. Second, based on the identified services, we will derive the graph-property data model that will support the context-aware design assistant.

2 Literature Review

[Information retrieval] The problem of providing the right information to the right person at the right time is one of the fundamental goals that motivated academics and industrialists to work out a new strategic product-centric, lifecycle oriented and information-driven approach – Product Lifecycle Management (PLM). Numerous PLM and knowledge engineering research studies proposed state-of-the-art solutions to improve the access and reuse of information stored in engineering documents [1–5]. Although, the basic information retrieval capabilities – e.g. keyword search, faceted search, etc. – of search engines facilitate the access to textual content, the lack of a structured representation degrades the performance for many reasons (technical terms, ambiguities, etc.) [6]. This is the reason why researchers have reused semantic web techniques including modelling languages (e.g. RDF, RDFS, OWL), query languages (e.g. SQWRL), and software (e.g. Protégé) for modelling domain-specific knowledge, such as geometry and topology [7, 8], feature recognition [9], generative modelling [10], nuclear design rules [11], configuration management [12] and so on. An ontology, in its broadest sense, that is, a description providing a shared understating of a given domain, facilitate the reuse of knowledge, but it is extremely time-consuming to be developed and maintained. It is therefore interesting to use natural language processing and text mining techniques to not only automate the acquisition and processing of knowledge, but also to integrate both rule-based and machine learning-based capabilities to make the assistant "intelligent".

[Cognitive assistant] Cognitive assistants, which are also known as expert systems or knowledge-based agents, are "intelligent" computer programs that learn more or less complex problem-solving expertise from human experts so as to assist human nonexpert in solving similar problems [13]. One key feature of cognitive assistant is its ability to adapt itself to a given context that is not limited to linguistic characteristics like information retrieval systems. It can therefore provide better answers than a search engine. For instance, a cognitive assistant could process multi-factorial information including the user role, its social relationships in the company, the operational CAX environment he is using, etc. to provide personalised answers to questions asked by a designer.

[Context-aware] Many research studies on knowledge management refer to the concept of "context". For instance, Dhuieb et al. [14] propose a framework for managing manufacturing knowledge with a multiscale and context-aware approach. Although the application to manufacturing differs to design, the authors provide us with some details on the definition of the context that includes three viewpoints: operational (activities and task of the worker), organizational (team and role of the worker), and user-centric (expertise and skills). Related to our research goal, Rowson et al. [15] investigate the idea of building reusable expert knowledge using screen monitoring and contextual similarity. Context similarity is defined as the identification in real time of a resemblance between the script under elaboration and schemes in the knowledge base, but it seems to us that too many aspects of the framework are assumed, such as the form and the content of the knowledge base, the way it is fed with information, the query language and patterns, the similarity measure, etc. The concept of "context" remains therefore too fuzzy to be reused for our mission. If we extend our literature review to the theory of information retrieval in context, Ruthven [16] sums up the different way to explore context (related searches, keywords query expansion, query suggestion, etc.) based on various dimensions of the user context: task context, social context, personal context, spatio-temporal context, environmental context, etc. We may wonder, why is it so difficult to define what "context" means? The reason for this is simply that "context" is one of those suitcase-like words that we use to conceal the complexity of very large ranges of different things whose relationships we do not yet comprehend. In this paper, we propose to use a property graph data model to attempt to define the concept of context in product design.

[Property graph data model] A property graph data model is a model where data structures for the schema and/or instances are modeled as graphs for managing graph-like data and the data manipulation is expressed by graph-oriented operations using a graph query language [17]. A graph-oriented data structure facilitates the modelling of entities, relationships and properties that make up the design context. Using NoSQL graph-oriented database systems such as Neo4J is also flexible as we can create, read, update, and delete nodes and relationships without impacting the schema. This is a very important advantage since we will never come up with a complete property graph data model the first attempt.

3 A Property Graph Data Model for a Context-Aware Design Assistant

In this section, first, we detail the operational view, that is, the stakeholders, the services the context-aware design assistant shall provide to the stakeholders, and the inputs/outputs of the assistant. Second, we give the gist of the modelling process that leaded us to the property graph data model underlying the context-aware design assistant.

3.1 Operational Analysis of the Context-Aware Design Assistant

Our context-aware design assistant is a knowledge-based cognitive assistant that shall help designers to provide solutions satisfying applicable design rules.

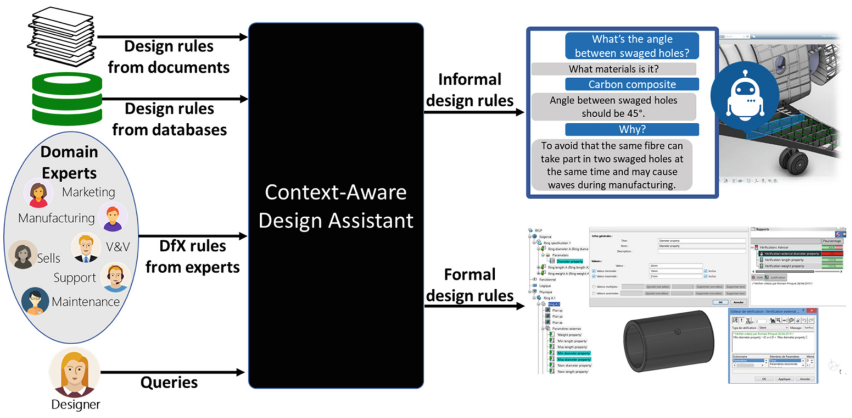

Design rules from documents
Design rules from databases
Domain Experts
Marketing
Manufacturing
V&V
Sells
Support
Maintenance
DfX rules from experts
Designer
Queries
Context-Aware Design Assistant
Informal design rules
Formal design rules
What's the angle between swaged holes?
What materials is it?
Carbon composite
Angle between swaged holes should be 45°.
Why?
To avoid that the same fibre can take part in two swaged holes at the same time and may cause waves during manufacturing.

Fig. 1. Context diagram of the context-aware design assistant

Figure 1 illustrates the stakeholders interacting with the context-aware design assistant. So far, we have identified two types of intended effects (outputs). First, the context-aware design assistant shall recommend informal – i.e. not computable – design rules to the designer. We do not guess how this service will be implemented – e.g. Chatbot, recommendation engine, search engine, (un)supervised learning, etc. Second, the context-aware design assistant shall communicate computable design rules to extend CAX capabilities (e.g. automating modelling routines, enriching models with semantic annotations, verifying design solutions, etc).

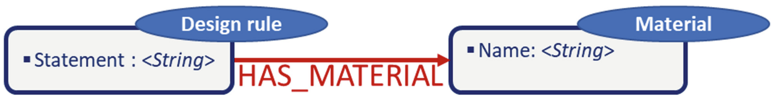
To provide both services, the context-aware design assistant requires inputs. In the one hand, there are the design rules that feed the context-aware design assistant. The design rules, which are recognized and codified knowledge [18], mainly come from unstructured (e.g. PDF, Word, etc.) documents, semi-structured (e.g. Excel, XML, etc.) documents, and databases. The second main source of design rules, which are recognized tacit knowledge (e.g. commonsense) or unrecognized knowledge (e.g. expertise and skill) [18], corresponds to domain experts (e.g. marketing, manufacturing, V&V, sells, support, maintenance, etc.) who shall systematize [18] Design for X rules in the context-aware design environment throughout the product lifecycle. Finally, in addition to the inputs corresponding to design rules, designers shall provide queries to interact with the context-aware design assistant.

3.2 Modelling of the Property Graph Data Model

To derive the property graph data model that supports the mission of the context-aware design assistant, that is, "*As a designer, I want to know which design rules my design shall satisfy, so that I can provide proof design.*", we follow a systematic 4-step modelling process:

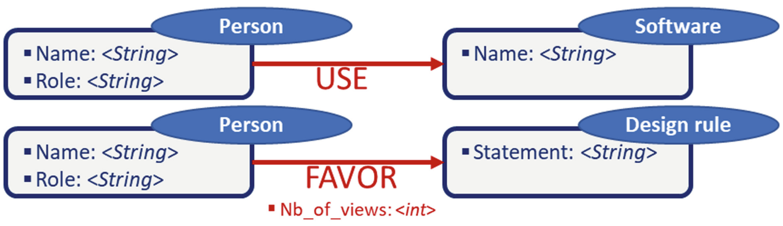
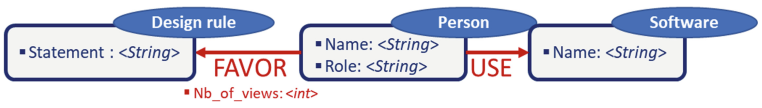
1. Find what questions the context-aware design assistant shall help designers to answer;
2. For each question, identify entities (nodes of the property graph) and relationships (edges of the property graph);
3. Express each question as a graph pattern.
4. Translate the graph pattern into a query path.

The simplest question to answer is a graph pattern corresponding to a predicate, that is, a triple (Subject – Predicate → Object) as follows:

Question	Which (design rules) [has_material] (material *X*)?
Graph Pattern	 Design rule ▪Statement : <String> HAS_MATERIAL Material ▪Name: <String>
Query Path	(:Design_rule) – [:HAS_MATERIAL] → (:Material)

Using such query, we can answer various questions, such as: Which design rule has manufacturing process *X*? Which design rule belongs to the engineering domain *X*? etc.

A graph-oriented data model brings an added-value when queries traverse richly interconnected data. We can therefore answer more sophisticated questions such as the one hereafter.

Question	Which design rules are favored by person who use the same software as me?
Graph Patterns	 Person ▪Name: <String> ▪Role: <String> USE Software ▪Name: <String> Person ▪Name: <String> ▪Role: <String> FAVOR ▪Nb_of_views: <int> Design rule ▪Statement: <String>
Consolidated Graph Pattern	 Design rule ▪Statement : <String> FAVOR ▪Nb_of_views: <int> Person ▪Name: <String> ▪Role: <String> USE Software ▪Name: <String>
Query Path	(:Design_rule) ← [:FAVOR] – (:Person) – [:USE] → (:Software)

It is challenging to enumerate all questions that the context-aware design assistant shall answer. Another complementary reductionist approach consists in defining the parts, which are not questions but pieces of the design context, before reassembling each component to recreate the whole property graph data model (Fig. 2). In general, there is a need for zigzagging between both approaches.

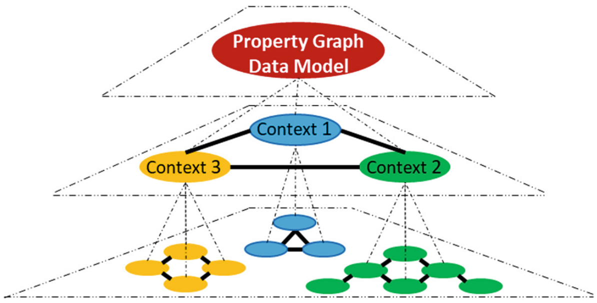


Fig. 2. Multi-level design context modelling

So far, we have identified five sub-contexts that make up the overall design context. Each sub-context is a sub-graph that we illustrate hereafter. To remain synthetic, we do not provide all properties of entities and relationships. For each context, we also give clues on how information can be acquired.

- **Social context:** It is the user profile and its relationships with colleagues. To capture the information, we ask each designer to fill a user profile form except for ":FRIEND_OF" relationships which are extracted from the social platforms deployed within the company (e.g. Slack, Skype, etc.) (Fig. 3).

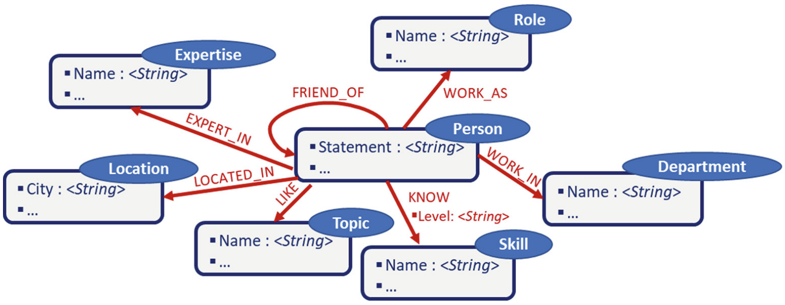


Fig. 3. Social context sub-property graph model

- **Semantic context:** It is mainly the result of natural language processing [19] and machine learning-based text mining [20, 21] techniques applied to a textual design rule. Keywords are words tagged with a part-of-speech corresponding to a noun, a verb, an adjective, or an adverb. In addition to the part-of-speech tagging, natural language processing techniques (sentence splitting, tokenization, lemmatization, and stemming) enable us to derive the stem and the lemma of each keyword. Removing the inflected forms of the keywords – lemmatization – enable us to get linguistically-related terms (synonym, holonym, meronym, hypernym, derived related terms, and the definition) from the Wordnet thesaurus. In addition to extend keywords with linguistic contextonys, we can use the open multilingual knowledge graph ConceptNet to find related concepts. For instance, using the conceptual relationship (:Airplane) – [:USED_FOR] → (:Travel) we can ease the navigation among design rules containing both entities (:Airplane) and (:Travel). The self-relationship [:SIMILAR_TO] on the (:Lemma) entity helps to retrieve similar normalised keywords (lemmas). The similarity score is computed using the Word2vec [22] and GloVe [23] language models (Fig. 4).

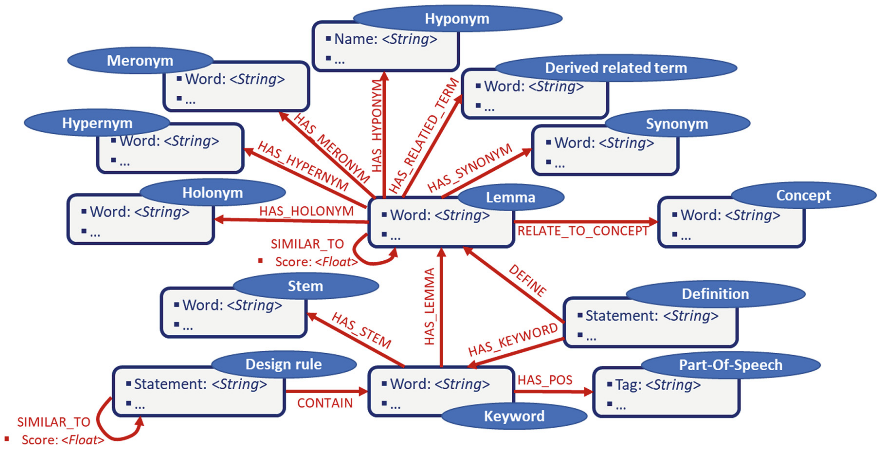


Fig. 4. Semantic context sub-property graph model

- **Engineering context:** It is a set of interrelated engineering information that is also derived by processing the text of the design rule statement. By using a rule-based classifier and taxonomies that enumerate materials, manufacturing process, and bill-of-materials we can identify keywords corresponding to specific domain knowledge. Thus, when a designer is looking for design rules related to a rib made of aluminum, he can explore such graph patterns. The (:Expertise) – e.g. electronics, mechanics, IT, etc. – entity to which the design rule belongs to can be inferred using a supervised machine-learning based classifier [20] (Fig. 5).
- **Operational IT context:** It is the current working IT situation within which the designer operates. The software (e.g. CATIA), the workbench (e.g. Part Design), and the operation (e.g. Extrusion) are software processes running on a machine and human-machine interactions that we can monitor. The data being edited (e.g. Beam.prt) and the PDM project within which the designer is working can be captured using the API of the PDM software. The self-relationship [:LINK_TO] represents link between data in the PDM software (e.g. link between a CAD model, its FEA mesh, and its 2D drawing) (Fig. 6).
- **Traceability context:** Finally, the traceability context enables designers to trace the origin of the design rules and manage their changes. When one or several documents

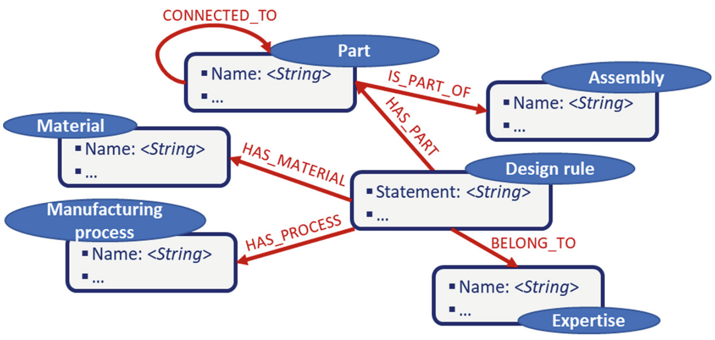


Fig. 5. Engineering context sub-property graph model

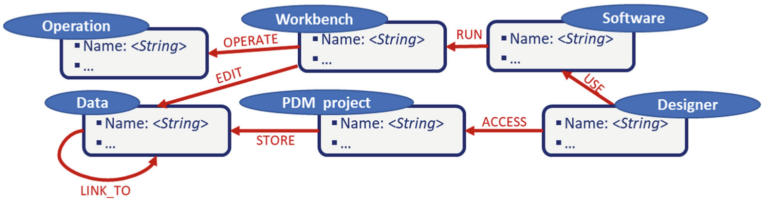


Fig. 6. Operational IT context sub-property graph model

are uploaded to the context-aware design assistant, a new job is created. Job serves to trace uploads. To facilitate the retrieval of design rules within the original documents, we trace the chapter within which it is stated. Most document parsers can retrieve the structure of documents, but there is a limit for some PDF documents encoded in a format that does not provide relevant semi-structured HTML or XML tags. The author of the design rule is either the person that directly prescribes it in the design assistant or the metadata "author" of the document source. Documents parser such as Apache Tika enables us to extract metadata. Finally, basic engineering change management concepts (maturity, revision and iteration) serve to trace the lifecycle of design rules and result from user manual inputs (Fig. 7).

3.3 Consolidation of Sub-property Graph Data Models

All sub-graphs corresponding to sub-contexts must be consolidated to end up with the property graph data model. We do not provide an overview of the whole property graph but we illustrate the concept of consolidation by merging the sub-graph of the social context and the sub-graph of the engineering context using the relationship [:HAS_TOPIC]. This consolidation enables designers to answer new questions such as "Which design rules belong to the expertise ($X = e.g.$ *Mechanics*) that has topic ($Y = e.g.$ *mechanical joints)* is liked by a designer who is a friend of mine?" (Fig. 8)

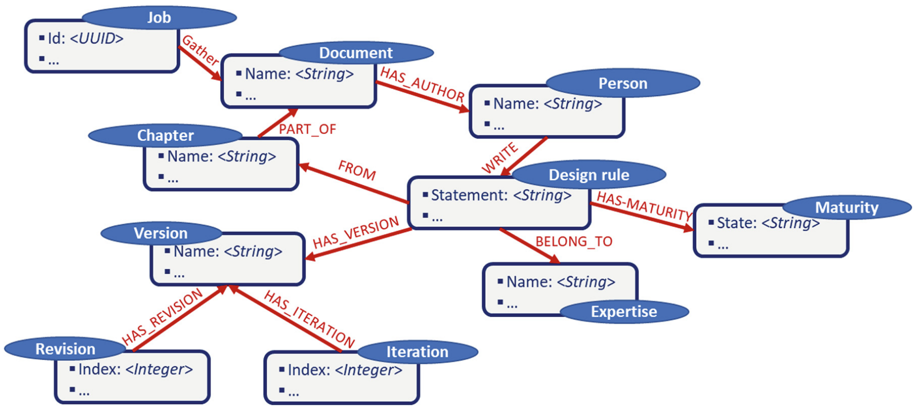


Fig. 7. Traceability context sub-property graph model

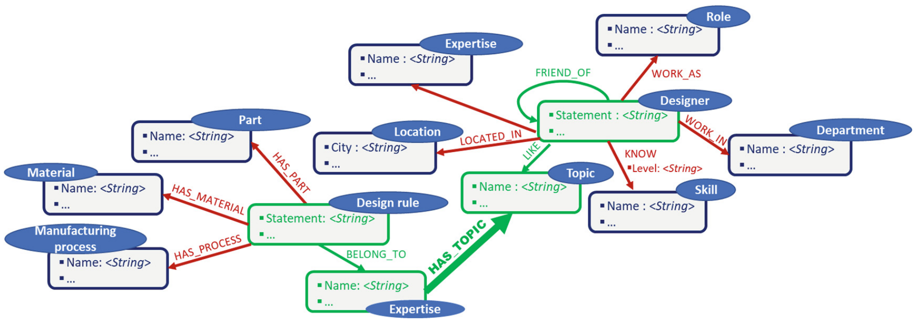


Fig. 8. Example of a consolidation of the social and engineering contexts

4 Conclusion

In this paper, we propose a property graph data model to support a context-aware design assistant. The assistant will use contextual knowledge to retrieve relevant design rules so that designers can create proof designs. The proposed property graph is the consolidation of five sub-contexts that can be queried using graph patterns.

As future work, we intend to continue to enrich our property graph data model and to develop the services that the context-aware design assistant shall provide to designers (design rules recommendation, design verification, design routines automation, etc.).

References

1. Dong, A., Agogino, A.M.: Text analysis for constructing design representations. In: Gero, J.S., Sudweeks, F. (eds.) Artificial Intelligence in Design 1996, pp. 21–38. Springer, Dordrecht (1996). https://doi.org/10.1007/978-94-009-0279-4_2
2. Marsh, J.R.: The capture and utilization of experience in engineering design. Ph.D. thesis, Cambridge University, UK (1997)

3. Song, S., Dong, A., Agogino, A.: Modeling information needs in engineering databases using tacit knowledge. J. Comput. Inf. Sci. Eng. **2**(3), 199–207 (2003)
4. McMahon, C., Lowe, A., Corderoy, M., Crossland, R., Shah, T., Stewart, D.: Waypoint: an integrated search and retrieval system for engineering documents. J. Comput. Inf. Sci. Eng. **4**(4), 329–338 (2004)
5. Yang, M.C., Wood, W.H., Cutkosky, M.R.: Design information retrieval: a thesauri-based approach for reuse of informal design information. Eng. Comput. **21**(2), 177–192 (2005)
6. Li, Z., Raskin, V., Ramani, K.: Developing engineering ontology for information retrieval. ASME J. Comput. Inf. Sci. Eng. **8**(1), 21–33 (2008)
7. Tessier, S., Wang, Y.: Ontology-based feature mapping and verification between CAD systems. Adv. Eng. Inform. **27**(1), 76–92 (2013)
8. Sanya, I.O., Shehab, E.M.: An ontology framework for developing platform-independent knowledge-based engineering systems in the aerospace industry. Int. J. Prod. Res. **52**(20), 6192–6215 (2014)
9. Wang, Q., Yu, X.: Ontology based automatic feature recognition framework. Comput. Ind. **65**(7), 1041–1052 (2014)
10. Sharka, W.: Application of MOKA methodology in generative model creation using CATIA. Eng. Appl. Artif. Intell. **20**(5), 677–690 (2007)
11. Fortineau, V., Fiorentini, X., Paviot, T., Louis-Sidney, L., Lamouri, S.: Expressing formal rules within ontology-based models using SWRL: an application to the nuclear industry. Int. J. Prod. Lifecycle Manag. **7**(1), 75–93 (2014)
12. Yang, D., Miao, R., Wu, H., Zhou, Y.: Product configuration knowledge modelling using ontology web language. Expert Syst. Appl. **36**(3), 4399–4411 (2009)
13. Tecuci, G., Marcu, D., Boicu, M., Schum, D.A.: Knowledge Engineering: Building Cognitive Assistants for Evidence-Based Reasoning. Cambridge University Press, Cambridge (2016)
14. Dhuieb, M.A., Laroche, F., Bernard, A.: Context-awareness: a key enabler for ubiquitous access to manufacturing knowledge. In: 48th CIRP Conference on Manufacturing Systems – CIRPS CMS 2015 (2016). Procedia CIRP **41**, 484–489
15. Rowson, H., Bricogne, M., Roucoules, L., Durupt, A., Eynard, B.: Knowledge capture and reuse through expert's activity monitoring in engineering design. In: Chiabert, P., Bouras, A., Noël, F., Ríos, J. (eds.) PLM 2018. IAICT, vol. 540, pp. 621–630. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01614-2_57
16. Ruthven, I.: Information retrieval in context. In: Melucci, M., Baeza-Yates, R. (eds.) Advanced Topics in Information Retrieval. INRE, vol. 33, pp. 187–207. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20946-8_8
17. Angles, R.: The property graph database model. In: 12th Alberto Mondelzon International Workshop on Foundations of Data Management, Cali, Colombia, 21–25 May (2018)
18. Yoshikawa, H.: Systematization of design knowledge. CIRP Ann. – Manuf. Technol. **42**(1), 131–134 (1993)
19. Pinquié, R., Véron, P., Segonds, F., Croué, N.: Natural language processing of requirements for model-based product design with ENOVIA/CATIA V6. In: Bouras, A., Eynard, B., Foufou, S., Thoben, K.-D. (eds.) PLM 2015. IAICT, vol. 467, pp. 205–215. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33111-9_19
20. Pinquié, R., Véron, P., Segonds, F., Croué, N.: A requirement mining framework to support complex sub-systems suppliers. In: CIRP Design 2018, Nantes, France, 23–25 May (2018)
21. Pinquié, R., Véron, P., Segonds, F., Croué, N.: Requirement mining for model-based product design. Int. J. Prod. Lifecycle Manag. **9**(4), 305–332 (2016)
22. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013)
23. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation (2014)