Methodology for Knowledge-Based Engineering Template Update

Olivier Kuhn, Harald Liese, Josip Stjepandic

HAL Id: hal-01567778
https://inria.hal.science/hal-01567778

Submitted on 24 Jul 2017

Methodology for Knowledge-Based Engineering Template Update

Olivier Kuhn[1], Harald Liese[1], Josip Stjepandic[1]

[1] PROSTEP AG
Dolivostr. 11, 64293 Darmstadt, Germany
{Harald.Liese, Olivier.Kuhn, Josip.Stjepandic}@prostep.com

Abstract. This paper addresses the problem of knowledge templates update in product development. A framework is developed to support template instances update process by providing a decision support system and an update strategy. An Issue-Based Information system is designed to allow the collaboration among various experts in order to solve template related problems. An ontology is used to represent and to reason on knowledge about templates, products and their relations. This allows the construction of an update sequence for the template's instances as it provides an efficient overview of relationships, even implicit ones. The proposed framework fills an actual gap concerning template updates, which slows down templates adoption for large scale industrial use.

Keywords: Knowledge-Based Engineering, Knowledge template, CAA V5, CATIA V5, Update Strategy, Ontology, Issue-Based Information System

1   Introduction

Nowadays, high-tech industries such as the automotive or aerospace are designing products that are more and more complex and that integrate various disciplines. This has lead to the emergence of collaborative platforms that allow several engineers or teams to work efficiently together on a project from distant places. Collaboration is a key to successfully release a product in time and with a high quality [1]. Several collaborative platforms have been developed to support cooperation using Web technologies to improve the integration and interoperability of remote users [2, 3].

However, having several persons with different points of view is a source of conflicts and misunderstandings due to, for instance, differences among domains' vocabulary. One solution to solve this problem is to formalize the technical terms of each domain and to create correspondences among concepts defined within each domain. This can be achieved by defining an ontology of the domains. An ontology allows to represent concepts and relationships between these concepts [4] that can be used to both infer implicit knowledge and to check the consistency of the domain definition by resorting to an inference engine [5].

Beside collaboration, another key factor in modern product design is the ability to reuse existing knowledge in design, products or processes. In this scope, one major change during the last years is the democratisation of Knowledge-Based Engineering

(KBE). KBE is a large field at the crossroads of Computer-Aided Design (CAD), artificial intelligence and programming.

1.1 Knowledge-based engineering templates

Knowledge-based engineering templates are "intelligent" models or features that are able to store design intent and product design knowledge. They can then restitute the knowledge by adapting them to design contexts, i.e. environments where the template is used, such as a car assembly model or a wing model. This paper focuses on the product design where templates can be of several kinds: models, features, etc.

Templates are parametric models designed with KBE elements such as formulas, rules, and automation. KBE elements allow a modification of the template content (e.g., configuration or geometry) according to given inputs. The process of putting a KBE template into a specific context is called "instantiation" and results in the creation of a copy of the template that will be put in the given context. This copy is called an "instance". According to the inputs, the instance will be configured to fit the specific context. Instances have a separate life cycle, which means that any modifications made to the template definition will not modify its existing instances.

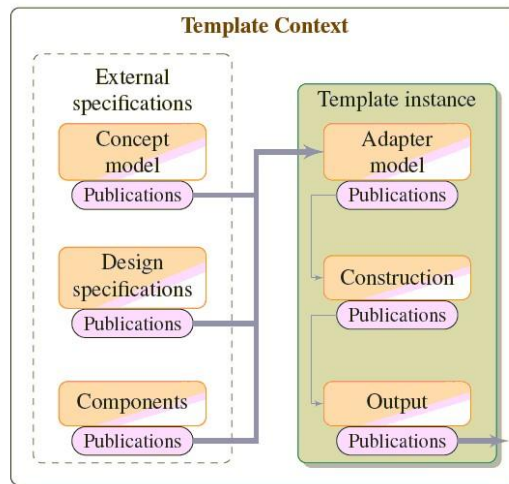**Fig. 1.** Generic structure of a CAD template instance in a context with link flow

Figure 1 presents a generic view of a template instance in a context. The structure of a template definition (and of an instance of this template definition) is composed of three main elements:

1. an adapter, which gathers the inputs and contains basic geometry information that will drive the design of the template,

2. the construction, which is the core of the template and that provides the functionality of the template (geometry generation, calculation, etc.), and

3. the output, which contains elements (geometry, parameters, etc.) that are provided in order to be referenced by external elements located in the context.

The inputs of a template instance can have various sources, which can be the geometry of other models, design specifications or external documents like spreadsheets. The references between elements should be established through output publications, which can be considered as the output interface of an element. The publications provide a named visible reference of an internal element of the document that can be easily identified and referred to. Furthermore, publications preserve the links in case the contents of an element are replaced.

1.2   Template update issues

The template life cycle (figure 2) is an iterative process. Before a template is deployed to end users, it has to be designed based on the requirements, tested, and packaged with the necessary documents (documentation, configuration spreadsheets, related CAD models...). Once deployed, templates are maintained and updated in order to fix possible bugs or to add new functionalities.
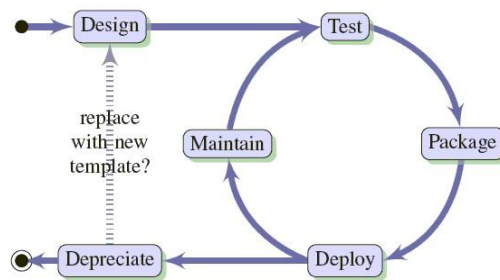

Fig. 2. Knowledge template life cycle

In large and complex assemblies like cars or airplanes the number of templates and template instances can reach several thousands and even more. This implies a huge effort to maintain them as they become more complex by incorporating new potential variants for future design [6]. Because several expertises are involved in the design of a product as well as different points of view, conflicts are very probable. The resolution of these conflicts needs a tool that allows remote stakeholders to propose their ideas. Then the solution coming out from the consensus can be implemented into the template and will give a new version of this template.

When a new template version is available, there is a need to update the corresponding template instances with the new definition, so that it can take advantage of the modifications and the same version is used everywhere in order to ensure the consistency of the instances with the corresponding templates. This synchronisation between the templates and their instances is not fully handled by

current Product Data Management Systems [7] and the impact of the modifications needs to be evaluated by hand. When working with large assemblies containing several thousands of template instances, finding a feasible strategy to update them all is a challenging and time consuming task. The complexity of the relations network in KBE assemblies complicates this task as the order in which instances are updated has a significant impact on the result and might generate time-consuming redundant updates. Relations between documents need an efficient visualisation tool in order to have an overview on all interdependencies [6]. Furthermore two categories of template instances were identified: the direct and the indirect instances. The direct instances are resulting from the instantiation of a template. The indirect instances are the result of the copy of a template instance included in another template.

The application of template modifications to their instances is a difficult and time consuming task. Engineers have to analyse assemblies and relations to estimate which documents have to be updated and then make the updates. Thus this task can lead to errors due to the complexity of the assemblies and relations. To efficiently achieve this task, an update strategy is needed in order to avoid redundant updates or to prevent forgetting one update.

The problem of template management was addressed by only few research activities in the past e.g. the management of templates [7]. The proposed solution was a business process that involves ontologies that are used to represent knowledge about templates and their interconnections. An ontology is created from each template by mapping them to the knowledge model. Each ontology describes the inputs, outputs of the template and interconnections with other models. So the changes made in one template can be propagated to other templates by using the relations described within the ontologies. However his approach focuses on templates to template relations and engineers still have to navigate the ontology to find dependencies.

In order to improve the update process, this paper proposes a framework composed of an ontology-based decision support tool, and an ontology coupled with an algorithm designed to compute a sequence of update tasks for the update of template instances.

2   Update process

The process depicted in figure 3 was elaborated in order to guide the users of the methodology in order to solve template issues and perform template instances updates.

The proposed process covers major aspects of template update, from the cause of the modification, to updating the instances. The process is decomposed into three subprocesses, which are marked (a), (b) and (c) in figure 3:

The aim of subprocess (a) is to allow the decision making process to take place in a collaborative environment, concerning the template issues or new requirements. The modifications applied to a template should then be documented and stored in the system for further use.

The subprocess (b) starts after the template update, with the aim to keep an up-to-date knowledge base about all documents involved in the product design, which are models, catalogues, assemblies, their relations and the templates. This knowledge

base provides a comprehensive overview of the dependencies between documents, which will be used in the following subprocess.
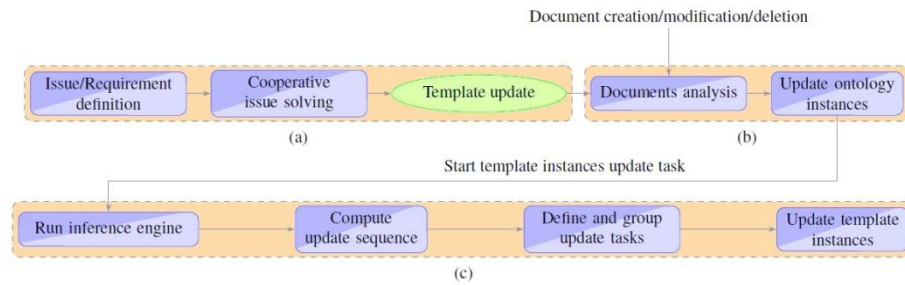

Fig. 3. Proposed global process for template update

The subprocess (c) is triggered according to company criteria (time, manpower, nightly processing, etc.). The objective of this subprocess is to support engineers in the template instances update tasks by providing them a sequence of necessary updates to follow. By doing this, the complex manual task of identifying the elements that require to be updated is avoided.

3 Decision support and design rationale for solving template issues

3.1 Issue-based information system

This section describes subprocess (a) of figure 3 which helps designers with different competences to collaboratively develop suitable design solutions. Decision support systems are already used in several domains such as design processes [8] or process planning [9]. The proposed approach is based on the Issue-Based Information System (IBIS) framework. IBIS is a generic argumentation-based framework for problem solving, which allows several stakeholders to propose their own positions about an issue and then argue about all the positions.

The IBIS framework can also be categorised as a design rationale (DR) tool [10], which represents and stores the decisions, the argumentations and the alternatives behind design choices.

The IBIS framework has been extended in order to solve template issues and to support further tasks such as the template instances updates in the subprocess (c).

3.2 IBIS extension

The IBIS framework provides the basic elements to support the collaborative solving of complex problems like template issues in this case. By storing the process and the choices that have lead to a design solution, it also endorses the role of a design

rationale solution. The IBIS framework has been extended with new concepts from the product design in order to store the knowledge specific to template updates.

An extension of the IBIS framework demonstrated on a CAD template application has been presented in [11]. The IBIS model is specialised in order to integrate the link flow aspects, which are relations between templates and other documents. The aim is to allow the documentation of these relations to provide CAD template designers as well as CAD template users with useful information that will allow speeding up their respective processes.

The structure of this system is implemented with the Web Ontology Language. The Web Ontology Language (OWL) is a knowledge representation language based on open standards and aims at defining ontologies by allowing the formal definition of their concepts and properties. OWL allows using reasoners on knowledge in order to automatically classify the knowledge and inference engines to discover new knowledge. The current work uses OWL-DL, which is a subset of OWL based on description logics [12], which provides the best expressiveness while preserving computational completeness.

4   Knowledge representation for templates and CAD models

By following a right update strategy, engineers will not need to look for template instances in the assemblies and will be informed about the documents that can be impacted by instances updates. The strategy is provided under the form of a sequence of updates to realise. To generate an update sequence, the algorithm requires information about the documents types and the existing relations between documents. To provide this information in a computer understandable and processable format, an application ontology has been developed that is represented in OWL. An ontology has been chosen because it allows the classification of knowledge and the use of inference engines to infer implicit knowledge, i.e., not provided by the documents but known by the designers.

4.1   Case study

In order to define an ontology that can be used as knowledge base to compute update sequences, an analysis of the template update process is needed. For this purpose, some template applications within CATIA V5 (a Dassault Systèmes CAD system) were analysed.

CATIA V5 integrates KBE workbenches that provide mechanisms to create and instantiate KBE templates. It is also possible to use standard CAD models as templates without resorting to a specific workbench. However with this method, there are no explicit template definitions and no support tool for template instantiation. The result of the instantiation with both approaches is the same, it is impossible to distinguish the template instance resulting of a standard model from an instance from the CATIA V5 Knowledge Template workbench.

Regarding relations between documents, 19 different types of links have been identified. Each link involves two documents, one for the source and the other for the

target of the link. Links are used to refer to, or use an element contained in another document. Hence some links have an impact on the update propagation.

For instances update, the users need to locate them, as there is no tracking mechanism. Once they are all located, a rank must be given to each of them in order to establish the order in which they are to be updated. This rank should be estimated under the consideration of the relations between documents. Engineers can then address the update itself by taking care that the elements referencing the instances are updated too.

The analysis on template update shows that several factors impact the instance updates: the document and template types, links between documents and the order in which the instances are updated. We propose to represent the domain with an ontology to obtain an overview of relationships and allow an automatic processing,

4.2 Ontology description

The aim of the ontology is to provide a classification and an overview of the document types and all explicit and implicit dependencies in templates and assemblies. This phase corresponds to the subprocess (b) in figure 3.
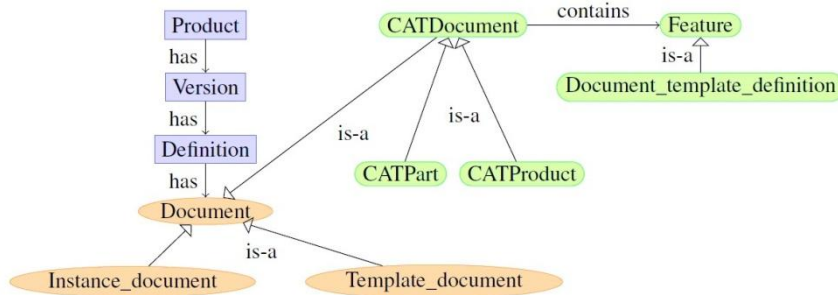

Fig. 4. Extract of the ontology presenting the concepts with the abstraction level (left) and the CAD system concepts, here CATIA V5 (green rounded rectangles)

An application ontology composed of three levels (see figure 4) was created, that features a top-down as well as a bottom-up approach. The top level (rectangles) contains the most general concepts that were taken from the STandard for the Exchange of Product model data (STEP): a product has several versions, each version is composed of several definitions, and each definition owns a list of documents. This level aims at laying a solid basis for the representation of products with their versions and definitions.

The third level (rounded rectangles) contains all CAD system specific concepts. Concepts presented in the figure are defined according to those in CATIA V5. For instance the "CATProduct" concept represents documents defining a product and thus has attributes like name, location on the file system and so on. In CATIA V5, the definition of a document template is a features contained within that document. There

are also feature templates that represent, for example, a predefined hole with its possible diameters listed in an external spreadsheet document. These concepts will be later instantiated with the data coming from the automated analyse of templates, assemblies and their related documents such as drawings.

Finally, the mid-level (ovals) realises the link between the first and the third level. In this level concepts, which are the core of the presented application, are defined: template, instance, document, etc. Their aim is to provide genericity to the presented approach in order to be able to address various CAD systems. With these defined concepts, a more comprehensive and detailed representation of the domain is achieved. The three levels present in the proposed ontology are connected together with different properties like "is-a", "equivalence" or "aggregation" properties.

Link types, represented as a relation in the ontology, have also been classified and problem specific types have been added. For instance, the "propagation link" has been created that group CATIA V5 links that impact the update of instances. This allows enriching the link classification and to formalise the implicit relations like reverse links, which will thereafter be instantiated by the reasoner.

The designed OWL ontology provides a comprehensive overview of elements present in the target domain. The FaCT++ reasoner [13] is used to classify the knowledge and also instances of the third level into upper levels concepts. All this knowledge can now be used to automatically build sequences of updates that will save time to engineers in the template instances update tasks.

5   Update sequence computation

The process of updating of template instances is triggered according to company criteria, such as every week. This is currently a task mainly achieved by hand. The aim of this work is to provide the engineers with a comprehensive sequence they can follow. This step corresponds to the process presented in figure 3 (c). Following this sequence will save time, as the engineers do not have to analyse the status of assemblies with all their complex interdependencies.

5.1   Approach

The documents and links can be represented as a directed graph where each document is a vertex and each link is an edge. The specificity of the obtained graph is that nodes and vertices are typed. Their types depend on the concepts and relations in which they are classified in the ontology. Represented concepts are those from the middle level of the ontology. Hence, the CAD system is abstracted in this representation. One node can have several types as documents can be classified under several concepts. The proposed algorithm works on this graph to extract relevant nodes and to assign them a rank. The objective of the ranking algorithm is to build an ordered sequence.

5.2   Sequence construction

In order to construct the ranked sequence, an approach based on hierarchical structure visualisation and directed graph drawing was designed [14]. These allow an efficient organisation of vertices according to their relations with other vertices. The algorithm they proposed is composed of four phases:

1. Place the graph nodes in discrete ranks.
2. Order nodes within ranks to avoid crossing edges.
3. Compute the coordinates of nodes.
4. Compute edge' splines to avoid nodes.

This paper focuses on the first phase, wherein nodes are ranked. The proposed method builds a hierarchy composed of n levels, from a directed and acyclic graph.

This algorithm has been adapted to the template instance update problem. The graph represents concepts as nodes and links as vertices. The classification realised in the ontology presented in section 4.2 is used to evaluate if the node has to be taken into account or not. The impacts of the link are defined in the ontology through the classification of the various links types.

Specific behaviours are implemented depending on the types of the documents: for example if a document is a template, the behaviour needs to be adapted because templates may be composed of other documents or instances that have to be up-to-date before updating the template's instances. Otherwise, the elements that replace the instance will not be up-to-date and may require additional updates, which is something that should be avoided.

Providing a sequence of updates for template instances in an automatic way avoids the time consuming task of preparing it by hand, as well as it reduces the possibility of making errors, redundant updates, and forgetting some instances. The computation of the update sequence is simplified by the use of the knowledge present in the ontology. Furthermore, the algorithm elaborated from the graph visualisation algorithm provides good results with a low computational complexity. The maintenance of the algorithm is facilitated as the classifications made in the ontology contribute to simplify the algorithm. Using a ranking approach also allows the parallelization of updates, as the documents located on the same rank have no dependencies against each other.

6   Scenario

This section presents the application of the proposed framework in a scenario, showing how it helps experts in template update related tasks.

6.1   Scenario description

The scenario is articulated around a clamp template (figure 5 (a)) that aims at holding objects. It was designed with a CATIA V5 CAD system using the dedicated template workbench. This template is itself composed of four template instances. The

clamp template is used in several assemblies. Figure 5 (b) shows an assembly context where three instances of the clamp template are present. They are used to hold a metal sheet in a manufacturing context. The clamp instances have different configurations, which are driven by individual parameter settings.
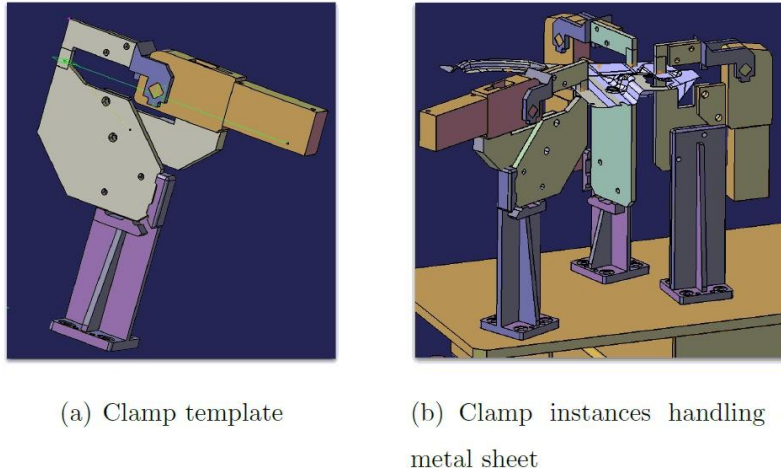

(a) Clamp template   (b) Clamp instances handling a metal sheet

Fig. 5. Three different instances of a clamp template in a metal sheet handling context

In this scenario an engineer is involved in the design of a new tooling station. He resorts to templates from a library to not design it from scratch. The tooling station requires handling metal sheets, so the engineer looks up in the library for a template providing this functionality and finds the clamp template. However, when looking into details, he realizes that the clamp is only meant to fasten horizontal surfaces.

6.2   Issue solving

The template user raises an issue that uses the decision support system (process (a) presented in figure 3). Four stakeholders are involved in the decision process: the template maintainer (A), the template user (B), a person in charge of the costs (C) and an electrical engineer (D). Figure 6 shows the argumentation that takes place to find a solution. Three positions are proposed for this issue, which are possible solutions for the problem. Each position refers to a rough sketch that illustrates the proposed ideas. Then, stakeholders provide their opinions on the positions according to their point of view. At this stage, two solutions fit the problem. To get more information on the initial issue, the template maintainer asks a question about the time constraint to deliver the new template version. In order to provide the new template in time, the chosen solution is to allow the modification of the angle of the clamp. So the desired angle can be set during the instantiation of the template.
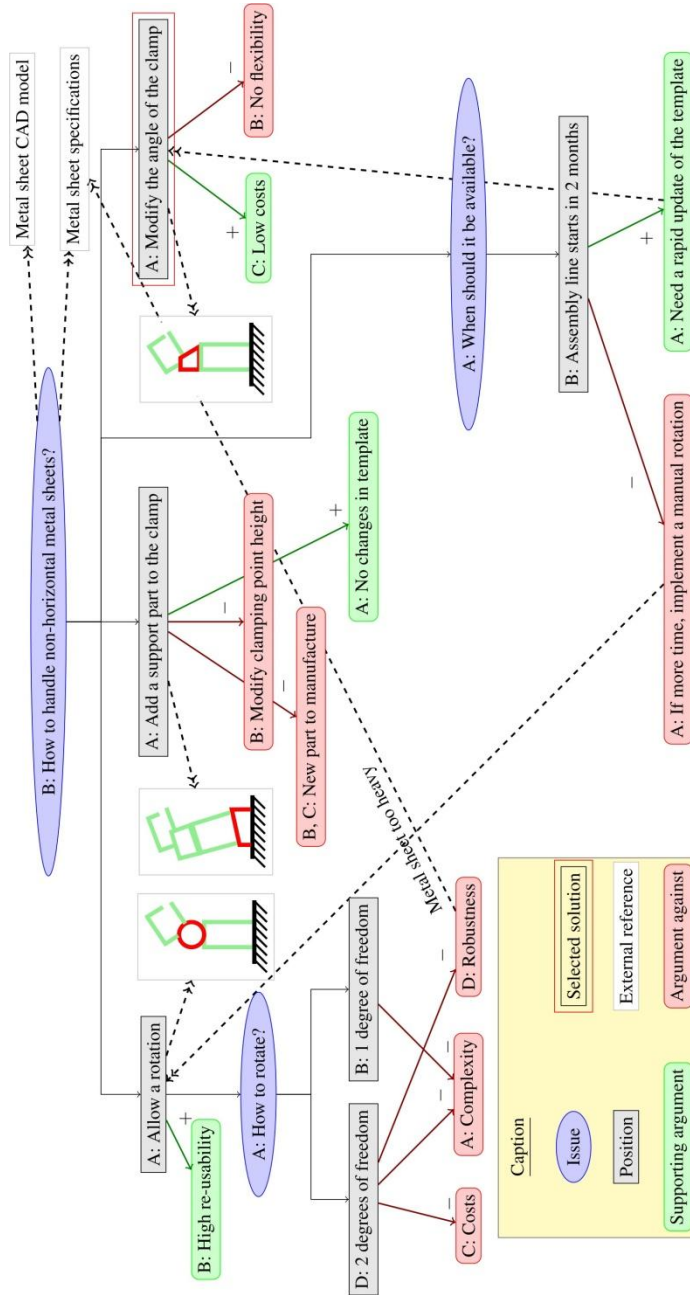
Fig. 6. Issue-based decision process: overview of the result

The adopted solution is to modify the angle of the clamp template with an input parameter in order to be able to generate clamp instances with different angles. The

resulting clamp will then be manufactured with a permanent angle. Now the planned modifications of the template have to be added and documented in the system. Then the template can be updated.

6.3   Template update

The proposed process has been designed in order to interfere as less as possible with the existing company processes. The framework that extends the IBIS provides indications about the template to be modified and the planned updates. Here the planned modifications are to change the angle of the clamp by using an input parameter. The angle has to be provided by the "vertical blade", which is the central part of the clamp. However the "vertical blade" is a template instance, so the best solution would be to update its template definition by adding an angle parameter and corresponding rules that will drive the design of this template. Thus modifications are only applied to the "vertical blade" to point out the benefits of the update sequence on a specific case.

After the update, actual changes to the templates should be documented in the system. This includes the modified documents with a summary of the modifications. This summary should be explicit enough in order to allow persons in charge of the template instances updates to derive the necessary actions to update the existing template instances.

6.4   Models analysis

In order to be able to take advantage of the definition of the domain in the ontology and to compute an update sequence, it is necessary to instantiate the concepts and properties from actual documents. This corresponds to the subprocess (b) of figure 3.

The first step is to extract relevant information from the models, such as the type and the contents of the documents, the relations, the template definition, etc. The analysis of documents is realised by a software developed in C++ and uses the CATIA V5 CAA API from Dassault Systèmes. This application analyses all CAD related documents, i.e., CATParts, CATProducts, catalogues, V4 models, etc. From the retrieved data, the OWL ontology is automatically instantiated, which now constitutes a knowledge base on the CAD related documents.

6.5   Update sequence generation

In the presented scenario, a single template document has been modified so far: the "vertical blade.CATPart" and will be the input of the algorithm. The corresponding computed sequence is shown in figure 7. The rectangles represent documents, the dashed line arrows point to the instances of a template and the full line arrows target a document contained in another. The figure shows that four ranks were created. The next section will explain how to use this sequence to help with the update of instances.
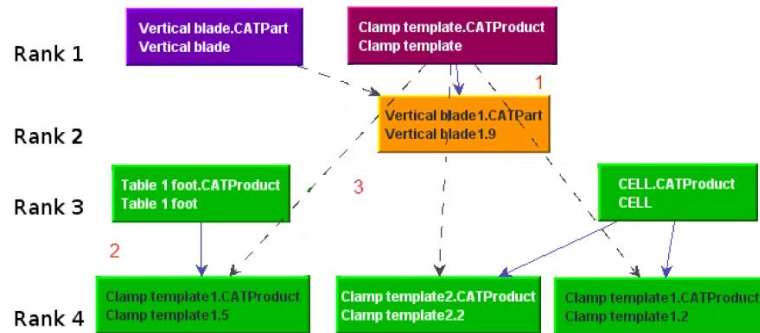
Fig. 7. Example of a computed update sequence

6.6   Instances update

Figure 7 shows the 8 documents among a total of 92 that are concerned by the update process. The "vertical blade" is the updated template and has an instance located in the "clamp template". Through this relationship, the "clamp template" is automatically included in the update process at rank 1. The actions to undertake are first to load the "clamp template" document that will implicitly load the "vertical blade" instance. Now the engineer checks in the IBIS-based system what modifications were applied to the "vertical blade" template. Then the context, which is the clamp template, has to be updated to in order provide the new input parameter to the "vertical blade". Then at rank 2 the old "vertical blade" instance is replaced. This results in a modification of the "clamp template" template, which also has to be propagated to its instances that are located in the "table 1 foot" and the "CELL" products. This is addressed by rank 3 and 4.

7   Outlook

This paper addresses the update of KBE templates and focuses on the update issues solving and the corresponding resolution process, as well as on the application of the modifications to the instances. A framework is proposed that provides a methodology and tools to enhance the process of template update, from the cause of the update (bug fix, new requirement, improvement, etc.) to the propagation of the changes to the instances of the template. It is composed of two main systems: a decision support and design rationale system as well as a template instances update support system.

The proposed framework presents various benefits. First, it helps to solve template issues through collaboration and provides a way to document the evolution of the templates. Regarding the update of instances, the framework generates a sequence of necessary updates from knowledge present in the ontology and thus saves time by avoiding the analysis of the assemblies in order to evaluate the impacts of the changes implemented in the template. Furthermore, the sequence is based on ranks which allow parallelising update tasks. The designed ontology can be easily reused for other

applications such as the visualisation of dependencies between documents at a large scale. The separation between the knowledge and the algorithm facilitates the maintenance as well as the evolution of the system. The coupling with another CAD system is easier as it consists in an evolution of the ontology with few modifications.

Further work would include the enhancement of the system by automating the last step of the process: the instances updates. Another possibility would be to extend the decision support system with case-based reasoning in order to ruse past experiences for new problems. Finally the validation of the scalability of our approach on large assembly sets as well as the visualisation of large graphs remain as further objectives.

References

1. Shen W., Hao Q., Li, W.: Computer supported collaborative design: next term retrospective and perspective, Comput. Ind. 59 (9) (2008) 855-862.
2. Dutra M. L., Ghodous P., Kuhn O., Minh T.: A Generic and Synchronous Ontology-based Architecture for Collaborative Design, Concurrent Eng. Res. Appl., 18 (1), (2010) 65-74; URL http://liris.cnrs.fr/publis/?id=4569
3. Hu Y., Zhou X., Li C.: Internet-based intelligent service-oriented system architecture for collaborative product development, Comput. Integr. Manuf. Syst. 23 (2) (2010) 113-125.
4. Mizoguchi R.: Tutorial on ontological engineering - part 1: Introduction to ontological engineering, in: New Generation Computing, Vol. 21, Ohmsha, 2003, pp. 365-384.
5. Cebrian-Tarrason D., Vidal R.: How an ontology can infer knowledge to be used in product conceptual design, in: B. Springer (Ed.), Computer-Aided Innovation (CAI), Vol. 277 of IFIP International Federation for Information Processing, Gaetano Cascini, 2008, pp. 57-68.
6. Katzenbach A., Bergholz W., Rolinger A.: Knowledge-based design – an integrated approach, in: S. B. Heidelberg (Ed.), The Future of Product Development, 2007, pp. 13-22.
7. Lukibanov O.: Use of ontologies to support design activities at DaimlerChrysler, in: 8th International Protege Conference, 2005.
8. Zha X. F., Sriram R. D., Fernandez M. G., Mistree F.: Knowledge-intensive collaborative decision support for design processes: A hybrid decision support model and agent, Comput. Ind. 59 (9) (2008) 905-922.
9. Chitta A., Shankar K., Jain V. K.: A decision support system for process planning, Comput. Ind. 14 (4) (2008) 307-318.
10. Regli W., Hu X., Atwood M., Sun W.: A survey of design rationale systems: Approaches, representation, capture and retrieval, Eng. Comput. 16 (3-4) (2000) 209-235. doi:10.1007/PL00013715. URL http://www.springerlink.com/content/w7ltxaufpay1rpwm/
11. Arndt H.: Eine Ontologie-basierte Methode zur Entscheidungsunterstützung in der Produktentwicklung, Ph.D. thesis, Technische Universität Berlin (2007)
12. Baader F., Calvanese D., McGuinness D. L., Nardi D., Patel-Schneider P. F. (Eds.): The description logic handbook: theory, implementation, and applications, Cambridge University Press, 2003.
13. Tsarkov D., Horrocks I., Fact++ description logic reasoner: System description in: International Joint Conference on Automated Reasoning, Vol. 4130, 2006, pp. 292-297.
14. Sugiyama, K., Tagawa, S., and Toda, M. Methods for visual understanding of hierarchical system structures. IEEE Transactions on Systems, Man, And Cybernetics, 11(2):pp. 109–125, 1981.