Semantic Web-Techniques and Software Agents for the Automatic Integration of Virtual Prototypes

Rafael Radkowski[1] and Florian Weidemann[2]

[1] Heinz Nixdorf Institute, Fürstenallee 11, 33102 Paderborn, Germany
[2] University of Paderborn, Warburger Straße 100, 33100 Paderborn, Germany
rafael.radkowski@hni.uni-paderborn.de, fweidema@uni-paderborn.de

Abstract. A virtual prototype is a computer internal representation of a real prototype. It is composited by a set of different aspect models. A common technique to analyze a virtual prototype is the usage of virtual reality applications. However, this requires a composition of different aspect models and their integration into a virtual environment. In this paper, an agent-based technique is presented, which facilitates this automatic integration of different aspect models. The Resource Description Framework is utilized to annotate the aspect models. A software agent compares the annotations of different models. By this it identifies similar models. A software prototype has been created that shows the usefulness of the approach.

1 Introduction

A virtual prototype (VP) is the computer internal representation of a real prototype or product [1]. It is based on the digital mock-up (DMU). The DMU represents the shape and the structure of the product. Two types of models are the origin of the DMU: 3D-CAD models and the logical structure of the product. The VP extends the DMU by further aspects. Aspects like the kinematic, dynamic, tension or information processing. A computer-internal model represents each of these aspects [2].

For the analysis of VPs the technology virtual reality (VR) and VR-applications are utilized commonly. VR displaces the user into a three-dimensional, computer-generated virtual environment (VE) [3]. To analyze a virtual prototype in the VE, the VP is integrated as a 3D model. Aspects like the kinematic behavior, tensions, etc. are calculated using simulation software. The results of these calculations also have to be integrated into the VE and to be combined with a certain VP. Therefore, it is necessary to combine different models, which represent the different aspects of a VP. The relations between the aspect models have to be identified, the data has to be specified that describe these relations, and finally everything have to be implemented. Today, different engineers do this integration manually, supported by several graphical interfaces that visualize the relations.

However, the manual integration is error-prone and time-consuming. The engineers have to know very well the meaning of different models. This is difficult when they do not know these models or use black-box models. Furthermore, the manual integration is labor consuming. Meanwhile, the engineers cannot focus on their intrinsic engineering tasks.

R. Shumaker (Ed.): Virtual and Mixed Reality, Part II, HCII 2011, LNCS 6774, pp. 387–396, 2011.

To face this problem, the Collaborative Research Center (CRC) 614 “Self-optimizing Concepts and Structures for Mechanical Engineering” follows a novel approach: Aspect models of VPs are annotated by techniques commonly used in the field of the semantic web. Software agents use these annotations in order to composite VPs autonomously and finally, they integrate them into a VE.

This paper is structured as following: The next section introduces some related work. The concept is explained in section three. Then an application example is presented. The paper closes with a summary and an outlook.

2 Related Work

The related work is separated into two sections. First the related work in the field of software agents supported engineering is reviewed. The second part introduces the utilization of the Resource Description Framework (RDF) to model technical systems.

In engineering, software agents are utilized in many different application fields. Mainly agents are used to support the design process by making decisions, which are based on a large amount of data. Mendez et al. describe an agent-based software architecture for agents in virtual environments [4]. They introduce the concept of expert agents. Expert agents are software agents with an expert knowledge in a specific technical domain. Based on this knowledge, the expert agent is capable to find a solution to solve a specific problem. This paper introduces a similar idea. However, their desired tasks are training tasks. Galea et al. present a framework for an intelligent design tool that assists a designer while working on micro-scale components [5]. They do not label their framework as software agent, but they use a similar artificial intelligence technique to model the knowledge and the reasoning system. Multi-agent systems have also been used to support engineers in time-critical tasks [6]. An agent aggregates relevant information from other agents that represent different members of an engineering team. Thus, an engineer gets the right information at the right time. Baolu et al. propose the so-called Multi-Agent Cooperative Model (MACM) [7]. It is a product design system that allows an easy access to similar data of different products. The system facilitates the product design and manages product data. With its aid the product design cycle will be shortened. Geiger et al. introduce the agent modeling language SAM (Solid Agents in Motion), a language to describe 3D models in virtual environments and their behavior. This work is similar to our approach. However, the agents generate rule-based animations to explain the kinematic behavior [8].

Some researchers have already used RDF and the related reasoning mechanism for the engineering of technical systems. For instance, Bludau and Welp have developed a framework, which supports engineers during the development of mechatronic systems [9]. Their framework searches for active principles and solution elements, which meet a given specification. Restrepo uses Semantic Web (SW) techniques to search for design solutions for a given problem [10]. He has developed a database, which contains different design solutions; every solution is annotated by RDF. A

reasoning mechanism searches for solutions, which meets the given design problem. Ding et al. use XML-based annotations to annotate CAD models with design constrains, goals, relationships, and bounds [11]. They mainly annotate geometric, topological and kinematic properties of a given design. Their approach can be utilized to find an optimal design solution during the product development process. The authors use XML as notation basis, but their notation is similar to a RDF notation. Ding et al. developed an XML-based product representation that allows an annotation of geometrical properties, too [12]. For further information, Li et al. present a classification of different annotation approaches [13].

In summary, the work supports the importance of software agents and annotation techniques in engineering design.

3 Concept for the Automatic Integration of Virtual Prototypes

Fig 1 shows an overview of the concept. On the bottom of the figure, the grey circle indicates the virtual environment. The boxes inside the virtual environment symbolize the VPs. A software agent represents each VP.

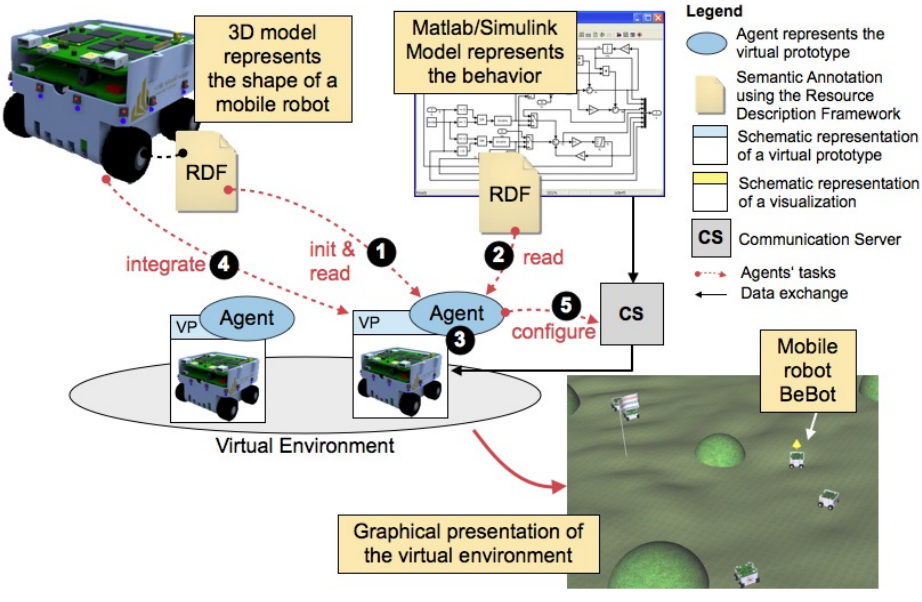

Fig. 1. Overview of the concept

In this example, a VP includes two models: a 3D model and a behavior model. Both models are shown on top of the figure. Left, the 3D model, the behavior model on the right. In this case it is a screenshot of Matlab/Simulink. The application

contains and processes a model that simulates the behavior. Both models are annotated. Therefore, an RDF-notation is utilized; the annotations describe the purpose of the models. Normally, more than two aspect models (3D model, behavior) and one VP are used. The small example should facilitate the understanding of the concept.

Objective of the software agent is to combine both aspect models (3D model, behavior) to one virtual prototype and to integrate them into the VE. In order to realize this a VP-template exits inside the virtual environment. The task of the agent is to integrate the aspect models into that template. It proceeds as following: The first step is an initialization by a user (1). Normally, the user specifies one model (3D model or behavior model) as origin. The objective of the agent is to identify the other models and to integrate them into the template of the VP. Therefore, it searches for every available model. A service directory of the agent platform references them. The annotation of every available model is read (2). The agent compares the RDF model of the 3D model with the RDF model of the behavior model (3). A set of production rules is used for this task. If two models pass the production rules, the agent assumes them as similar.

Then the 3D model is integrated into the VE by loading the model and including it into an internal data model (4). The behavior model cannot be included by loading. Normally, its processing would exceed the resources of every computer very fast. The behavior model is executed on a different computer system. However, only the results of the simulation are necessary for an analysis of the VP. Therefore, the results need to be transmitted to the VP. A communication server (CS) is used for that purpose. It manages the communication between the simulation software and the VP/VE. The task of the agent is to establish the communication between the behavior model and the 3D model. Therefore, the agent has to configure this communication server (5).

3.1 Semantic Annotations with the Resource Description Framework

The Resource Description Framework (RDF) is a description language, which is used to annotate the content of a web page in the context of the Semantic Web [14]; it is a syntax for meta data of a web page. The underlying model is based on a directed graph. The nodes of the graph are denoted as resources, the edges are denoted as properties. The idea of RDF is to describe complex facts by a network of simple RDF statements. A RDF statement consists of a subject, a predicate and an object.

In our concept the semantic annotation with RDF describes the purpose of each model. The challenge during the generation of the annotation has been to figure out the elements of a certain model, which need to be annotated in order to describe the purpose of a model and finally to facilitate the automatic integration of the model. We have specified models for following aspects of a VP: shape (3D model), behavior, functions, and activates. In the following the annotation of a 3D-model is explained by an example. **Fig 2** shows a 3D model of a shock absorber and its annotated items. The symbols in the figure show a graphical representation of an RDF-based semantic annotation.

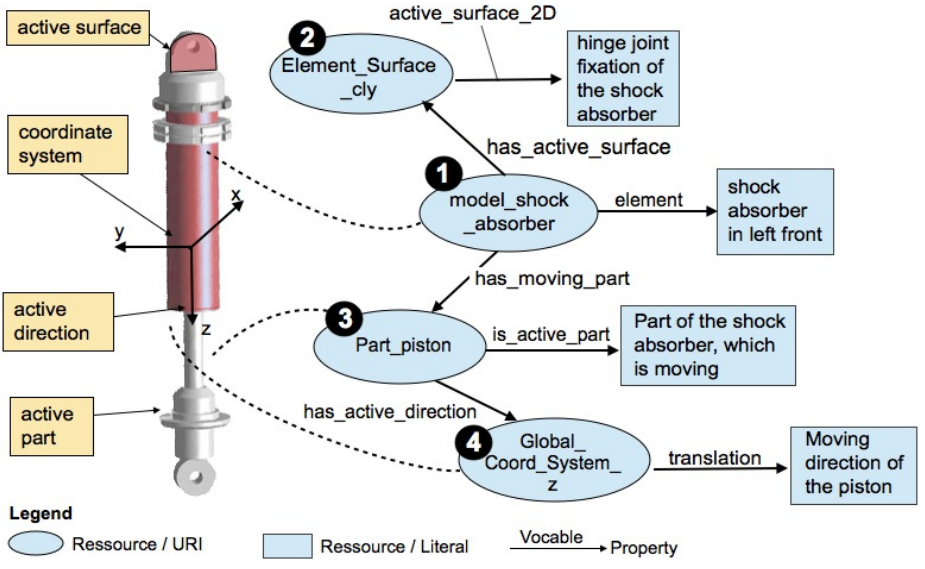

Fig. 2. Example of the semantic annotation of a 3D model

Four items of a 3D model has to be annotated: the entire part, the active surfaces, the sub-parts and the active directions:

- Entire part (1): The resource is linked to the variable, which represents the model, normally a file. In this example, the name of the model is *model_shock_absorber*. The variable is annotated by the predicate *element*. To describe the *element*, a literal is used. In this example it describes the part as "shock absorber in front left".
- Active surface (2): The active surfaces of a component are the surfaces, which fulfill the functions of this component [15]. The resource is linked to the variable in the data structure of the 3D model, which represents the active surface. In the example, the name of the variable is *element_surface_cly*. As predicate the word *active_surface_2D* is used. This predicate specifies the item as active surface.
- Active part (3): This type of part is moving to cause an effect of the entire model. The resource refers to the variable, which describes the main part in the data structure of a 3D model; here it is the entire pisiton. The word *has_moving_part* is used as RDF predicate to annotate the sub-part, which describes the part in the structure of the entire 3D model; the variables name is *part_piston*. Furthermore, to describe this active part, a predicate *is_active_part* is used. It facilitates the annotation with a literal. In this case the literal contains: "Part of the shock absorber, which is moving."
- Active direction (4): The fourth annotated type of item is the active direction. According to Pahl/Beitz [15] the active direction describes the direction, into which a function of a component effects. In figure 2, the piston of the shock absorber is the active part, which active direction should be described. The variable *global_coord_system_z* describes the coordinate system; it describes the direction of moving. It is attached to the resource *part_piston* by the predicate

has_active_direction. In addition, the resource *global_coord_system_z* needs to be annotated with a human understandable literal. In the example, the literal says "*Moving direction of the piston"*. It is attached to the resource by the predicate *translation*.

The entire description shows one example only. The example should give an impression, how the semantic annotation with RDF is working and which elements of a 3D model are necessary in order to describe the purpose and functionality of a 3D model in a natural way (literals). Altogether 36 RDF keywords were defined to describe the active surfaces and directions as well as the parts / sub-parts of an assembly. Further details of the model are explained in [16].

3.2 Software Agent Reasoning

The software agent has two major tasks: First, it has to find similar aspect models and second it has to establish the communication and the exchange of data between different software tools. In the following the reasoning mechanism is described. **Fig 3** shows the principle by an example.

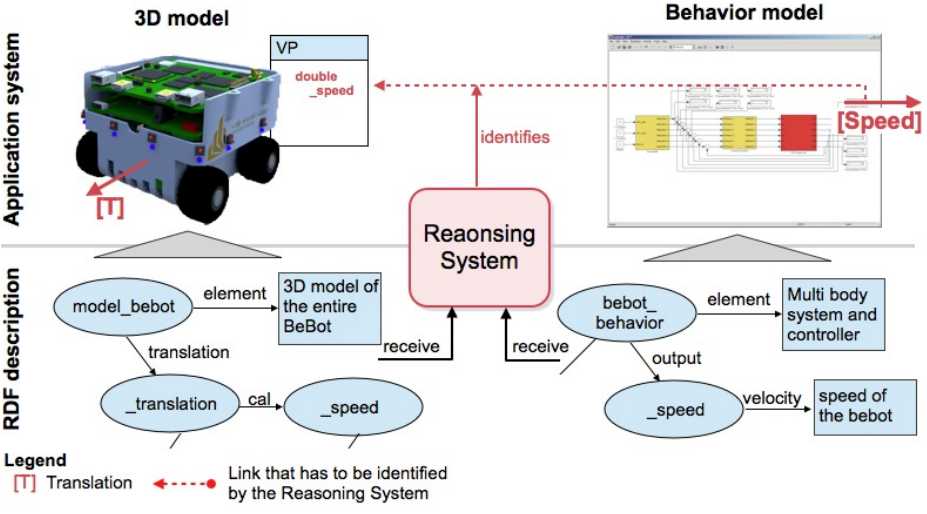

Fig. 3. Schematic overview of the reasoning by an example

On the left side of the figure a 3D model of a mobile robot is visualized, on the right side the behavior model of the robot is indicated. Both models are annotated using an RDF notation; the figure shows a part of the RDF description only. The mobile robot is the so-called BeBot, which is under development at the Heinz Nixdorf Institute. The BeBot is an autonomous driving robot; it can fulfill different tasks in a team. To get a VP of the BeBot the variables of the behavior model need to be linked with the related variables of the 3D model. In this example the variable *speed* is shown in the figure.

The reasoning mechanism identifies variables that are related to each other. In general the software agent compares the RDF models, two at the same time, and converts the results of the comparison into a numerical value. This numerical value expresses the similarity of two models respectively their variables. The comparison is based on production rules. Each production rule has the form:

$$\text{IF (Condition } C_1 \text{ \& Condition } B_1 \text{ \&… \& Condition } C_n \text{ \& Condition } B_m) \text{ THEN } A_1; \ldots; A_o \quad (1)$$

Conditions of type *C* are predicates of the 3D model, conditions of type *B* are predicates of the behavior model. By these production rules a set of corresponding predicates is identified. As corresponding predicates each pair of predicates is defined, which describes the same meaning of an item. For instance, condition *C* states *translation & cal* (calculate) and condition *B* states *output & velocity* are defined as corresponding predicates; they result in an output $A = 1$. Otherwise they result in an output $A = 0$. The result of this calculation is weighted by a weight value *g*:

$$A_o = A\ g + E \quad (2)$$

The value *g* indicates the importance of a production rule. The term *E* is an offset. It is calculated by a comparison of the literals of each pair of corresponding predicates. This is done by a statistical phrase analysis (for further details [16]). The results of every production rule are described as a vector:

$$A_{similar} = \{A_1, A_2, \ldots, A_o\} \quad (3)$$

This vector is a rating scale for the quality of the similarity in a certain task.

After the vector is determined the agent ranges all results $A_{similar,i}$, where the index *i* refers to a certain production rule of two compared models. A statistical method is used for this comparison, the so-called squared ranking. This method calculates a likelihood value *p(i)* for each corresponding pair of predicates:

$$p(j) = \frac{1}{size} \cdot \left(E_{max} - \left(E_{max} - E_{min} \right) \cdot \frac{(R_{similar,j} - 1)^2}{size - 1} \right) \quad (4)$$

With two rating values E_{max} and E_{min}. These values express the estimated amount of minimal and maximal corresponding predicates, respectively the number of possible relations. The equation assigns a numerical value to each production rule and expresses the fulfilled rules by a numerical value. A high value indicates the similarity of the compared variables.

The agent links all data, whose value *p(j)* exceeds a threshold $p_{threshold}$.:

$$p(i) > p_{threshold}$$

At this time the threshold is determined empirically. After this decision, the agent establishes the communication between the behavior model and the 3D model. Further information about the communication infrastructure and the behave of the agent inside the virtual environment has been presented in [17].

4 Application Example

To test the entire concept a software prototype has be developed and the described concept has been realized and integrated into this application. The software prototype consists of four components: The first component, a virtual environment is based on OpenSceneGraph (www.openscengraph.org), an open source scenegraph library for the development of 3D graphic applications. The second component is a simulation for mobile robots. This simulation is carried out with Open Steer (http://opensteer.sourceforge.net/), an open source software library. The third component is JADE (Java Agent DEvelopment Framework). JADE is a software framework that facilitates the implementation of multi-agent systems through a middleware that complies with the FIPA (Foundation for Intelligent Physical Agents) specifications, a standard specification for software agents. The fourth component is a communication server. It realizes the exchange of data between the three components, mentioned before. The technical aspects of the server are described in [17].

A test application has been developed to analyze the usefulness of the developed models and the production rules. Objective has been to find out whether the discussed way facilitates integration of two models to one virtual prototype. Therefore, a Capture-the-Flag (CtF) application has been used. Originally CtF is a game where a hunter needs to capture a flag, the other players chase the hunter and try to prevent him from capturing the flag. In our case the players are the BeBots; one is the hunter, the other try to chase it. The BeBots operate autonomously without any interaction by a user. Figure 4 shows two screenshots of the application. The left figure shows an overview of the virtual environment. The flag stands in the middle of the environment. Spheres are placed as obstacles. The BeBots need to avoid them. The right figure shows a detailed view to the scene. The BeBot with the yellow diamond on top is the hunter and tries to capture the flag.

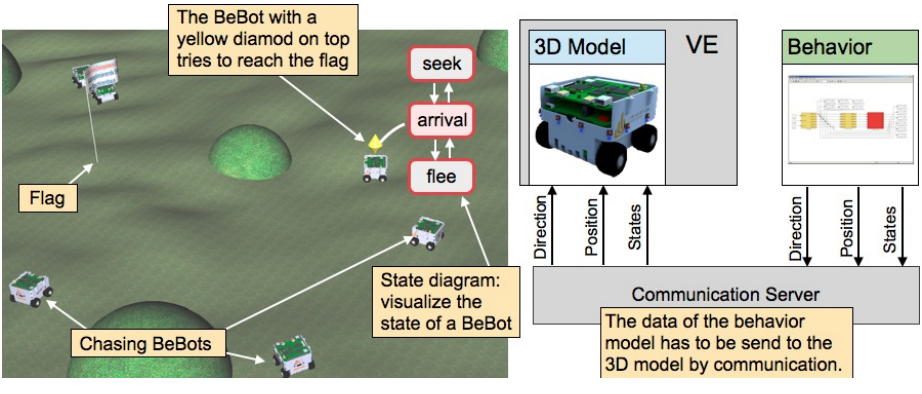

Fig. 4. Overview of the application (left), the architecture of the application (right)

Each robot is represented by a 3D model and a behavior model. Both models have been annotated by the RDF notation [16]. The annotation of the 3D model describes the input variables to set the position and direction of a robot as well as a state

diagram to visualize its current state. The behavior model provides the position and direction of each robot. Both applications (VE and behavior) need to be linked by the agent respectively the agent has to identify the variables and to link them.

In summary the agent has been able to realize the communication between both models / applications utilizing the RDF-based annotations of both models. The desired application could be realized, without any need for a user to describe the communication manually.

5 Conclusion and Outlook

This paper introduces software agents that facilitate the automatic integration of different aspects model of virtual prototypes. The aspect models become annotated by and RDF-notation. The notations of two models are processed and a vector is determined that expresses the similarity of two models. The concept has been tested. Therefore, an application has been developed that includes different mobile robots. Task of the agent has been to integrate the models into a template of a virtual prototype. The results show us that the concept can be utilized for the desired task. The results are a first step, but two things could be shown by this work:

First, a RDF description for the different aspect models is the right way to annotate the different models. The resources and properties of the developed RDF notation facilitate the computer-internal annotation of the task, the visualization, and the functions. They can be used as meta data for these aspects.

Second, software agents are a suitable solution to structure the software for the desired task. The agent communication allows the loading of different RDF notation. Furthermore, production rules can be utilized as internal agent model, as knowledge base, that drives the decision.

In summary the approach works well, when two models are used that represent different aspects (shape, behavior) of an equal technical system like the BeBot.

Altogether the results are a good starting point for further research. Until now, the carried out tests show us the usefulness of the concept at separated applications. The concept has not been generalized. In order to generalize it, a common set of production rules needs to be identified. For this purpose, we will perform a study utilizing different application examples with different VPs. On this basis a common set of production rules will be developed.

Acknowledgement. This contribution was developed and published in the course of the Collaborative Research Center 614 “Self-Optimizing Concepts and Structures in Mechanical Engineering” funded by the German Research Foundation (DFG) under grant number SFB 614.

References

1. Krause, F.-L., Jansen, C., Kind, C., Rothenburg, U.: Virtual Product Development as an Engine for Innovation. In: Krause, F.-L. (ed.) Proceedings of the 17th CIRP Design Conference on The Future of Product Development. Springer, Berlin (2007)

2. Gausemeier, J., Ebbesmeyer, P., Kallmeyer, F.: Produktinnovation. Carl Hanser Verlag, München (2001)
3. Gutierrez, M., Vexo, F., Thalmann, D.: Stepping into Virtual Reality. Springer, Berlin (2007)
4. Mendez, G., de Antonio, A.: An Agent-Based Architecture for Collaborative Virtual Environments for Training. In: Proceedings of the 5th WSEAS Int. Conf. on Multimedia, Internet and Video Technologies, Corfu, Greece, August 17-19, pp. 29–34 (2005)
5. Galea, A., Borg, J., Grech, A., Farrugia, P.: Towards Intelligent Design Tools for Micro-scale components. In: International Conference on Engineering Design, ICED 2009, Stanford, CA, August 24-27, pp. 5-73 – 5-84 (2009)
6. Payne, T.R.: Agent-based Team Aiding in a Time Critical Task. In: HICSS 2000, Proceeding of the 44th Hawaii International Conference on System Sciences, vol. 1 (2000)
7. Baolu, G., Shibo, X., Meili, C.: Research and Application of a Product Cooperative Design System Based on Multi-Agent. In: Third International Symposium on Intelligent Information Technology Application, pp. 198–201 (2009)
8. Geiger, C., Lehrenfeld, G., Mueller, W.: Authoring communicating agents in virtual environments. In: Proceedings of the Computer Human Interaction, Adelaide, SA, Australia, pp. 22–29 (1998)
9. Bludau, C., Welp, E.: Semantic Web Services for the Knowledge-based Design of Mechatronic Systems. In: Proceedings of the International Conference on Engineering Design, ICED 2005, Melbourne, Australia, August 15-18 (2005)
10. Restrepo, J.: A Visual Lexicon to Handle Semantic Similarity in Design Precedents. In: Proc. of the 16th International Conference on Engineering Design, ICED 2007, Paris (2007)
11. Ding, L., Matthews, J., Mullineux, G.: Annacon: Annotation with constrains to support design. In: International Conference on Engineering Design, ICED 2009, Stanford, CA, August 24-27, pp. 5-37 – 5-48 (2009)
12. Ding, L., Davies, D., McMahon, C.A.: The integration of lightweight representation and annotation for collaborative design representation. Research in Engineering Design 19(4), 223–238 (2009)
13. Li, C., McMahon, C., Newnes, L.: Annotation in Design Processes: Classification of Approcches. In: International Conference on Engineering Design, ICED 2009, Stanford, CA, August 24-27, pp. 8-251 – 8-262 (2009)
14. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
15. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H.: Engineering Design: A Systematic Approach, 3rd edn. Springer, Berlin (2006)
16. Radkowski, R.: Towards Semantic Virtual Prototypes for Automatic Model Combination. In: 20th CIRP Design Conference, Global Product Development, Nantes, France (2010)
17. Radkowski, R., Waßmann, H.: Software-Agent Supported Virtual Experimental Environment for Virtual Prototypes of Mechatronic Systems. In: Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality WINVR 2010, Ames, Iowa, USA, May 12-14 (2010)