Campos et al., *Cogent Engineering* (2014), 1: 939737
http://dx.doi.org/10.1080/23311916.2014.939737

cogent engineering

PRODUCTION & MANUFACTURING | RESEARCH ARTICLE

Industrial open source solutions for product life cycle management

Jaime Campos[1*], Juha Kortelainen[2] and Erkki Jantunen[2]

Received: 05 March 2014
Accepted: 19 June 2014
Published: 22 August 2014

*Corresponding author: Jaime Campos, Department of Informatics, Linnaeus University, Växjö, Sweden
E-mail: jaime.campos@lnu.se

Reviewing editor:
Zude Zhou, Wuhan University of Technology, China

Additional article information is available at the end of the article

Abstract: The authors go through the open source for product life cycle management (PLM) and the efforts done from communities such as the open source initiative. The characteristics of the open source solutions are highlighted as well. Next, the authors go through the requirements for PLM. This is an area where more attention has been given as the manufacturers are competing with the quality and life cycle costs of their products. Especially, the need of companies to try to get a strong position in providing services for their products and thus to make themselves less vulnerable to changes in the market has led to high interest in product life cycle simulation. The potential of applying semantic data management to solve these problems discussed in the light of recent developments. In addition, a basic roadmap is presented as to how the above-described problems could be tackled with open software solutions.

Subjects: Engineering & Technology, Information & Communication Technology (ICT), Technology

Keywords: product life cycle, simulation, data management, Semantic Web, open source software

ABOUT THE AUTHOR

Dr Jaime Campos is an associate professor at the Department of Informatics, Linnaeus University, Sweden. He graduated in both Informatics and Business Economics from the School of Business at the University of Gothenburg. His PhD focuses on the development of mobile and desktop applications within the EU finance project DYNAMITE (Dynamic Decisions in Maintenance). Within this project, he has worked directly with VTT (Technical Research Centre of Finland) (www.vtt.fi) which was the project coordinator. The application developed is based on Web and embedded technologies and designed to provide various services for the production industry. He has been a reviewer of a number of scientific journals and a project manager of research projects. His main research interests include the Information and Communication Technologies, especially Web technologies as the Semantic Web, Web 2.0, Agent and Mobile technologies in the industrial domain, especially e-maintenance.

PUBLIC INTEREST STATEMENT

Managing product and business data in product life cycle process are becoming increasingly challenging. Companies are operating in a global environment, their processes and product cycles are becoming faster, and the need for having the right data at the right place in right time is becoming crucial. Large information technology (IT) systems are the tools to manage this complexity, but designing and implementing these IT systems is a complex and demanding challenge itself. This article goes through the concept of applying open source development model for product life cycle management (PLM) and the efforts done from communities such as the open source initiative (OSI). The characteristics of the open source solutions are highlighted and the authors go through the requirements for PLM. The potential of applying semantic data management to solve the challenges is discussed in the light of recent developments.

1. Introduction

The challenge in manufacturing and service-based business can be compressed into the following statement: to optimise quality with minimum resources fast into the market. This multi-objective optimisation task has three complex sub-objectives that are often conflicting with each other. Increasing quality usually increases costs and may also require more time in order to get the product to the market. Simulation-based product development methods provide help for this by providing the ability for concurrent design of different subsystems of the product and enabling the coupling of the different design domains early in the development process. The trend of increasingly using simulation in the early phases of the product life cycle can be seen e.g. in vehicle and aeroplane industry, where the current short length of the design cycle is mostly achieved by using computational methods and concurrent product development. The application of simulation in the product life cycle process has been seen as one of the key factors for the success of industry (Glotzer et al., 2009).

As the competition in the markets keeps increasing, due to e.g. transition to global markets adoption of new design and manufacturing technologies, the companies are forced to look for new areas for their business to increase the solidity of the business. Especially in the mechanical engineering industry, the service business has become the second and equally important supporting pillar for the business. This, on the other hand, sets more expectations and demands on the application of simulation in the whole product process. Because the service business has increased its importance for the overall business, the influence of the design of the product has to be taken into account for the whole product life cycle, including the service business. For a product of 25 years of expected life cycle, design decisions that hold back the service business for the whole product life cycle have high importance. Consequently, to optimise the whole business and product life cycle, all the aspects of the product life cycle need to be taken into account in the product development phase. In addition to technical and business point of views, ecological aspects and legislation set both opportunities and constraints to the whole product process management, in which simulation can provide valuable help. The simulation process can be divided into the following phases: modelling (creation of the computational representation of the system to be studied), simulation (numerical solving the computational model with given initial values of the system and the simulation case), and analysis of the results (necessary actions to manipulate the results data easier processing and making conclusions of the data). For reliable simulation, the models have to be accurate and the modelling data have to be relevant and up-to-date. For large-scale simulations, such as business scenarios and durability analyses, gathering and managing modelling data becomes a challenge. However, the basic purpose of ICTs, such as information systems, decision support systems and simulation, is to acquire and represent information and knowledge (Fernandez, Labib, Walmsley, & Petty, 2003; Huber & Carlson, 1990; Nagarur & Kaewlang, 1999). It is believed that the more complete data, information and knowledge a company has in different situations, the more accurate decisions are made (Turban, Leidner, McLean, & Wetherbe, 2007). The authors also state that in such a situation, the decision-maker can be viewed as a perfect predictor of the future. In this article, the characteristics of open source software and the advantages of standardisation and open specification are highlighted. Thereafter, the challenges and requirements for product life cycle simulation (PLCS) are discussed. In addition, the bottlenecks of the development of PLCS solutions and discussion on actions to remove or decrease them are provided. In section four, an example of an open source software approach for a PLCS solution utilising semantic data model for simulation data is introduced. The example emphasises design data management, but the concept can be extended to the other areas of product life cycle management (PLM).

2. Open source development model and standardisation for PLCS

Centralised software solutions for enterprise data management and design data management have the tendency to become large, complex and monolithic. The size and complexity of such systems is difficult to avoid, due to the size and complexity of the system that they are designed to represent, but the architecture and design of the software solution have several options and justifications. Monolithic architectures have advantages and disadvantages. The advantages of the monolithic architecture are e.g. high-level of integration of the components in the software system and more

simple software management in the enterprise than with a solution composed of separate, independent components. The disadvantages of such architecture are e.g. dependency of one software vendor and difficulties in extending the software system with third-party software components.

The opposite of the centralised software architecture is a component-based architecture, in which the interfaces of the components in the system are open and well-defined. This architecture is more suitable for an ecosystem, because it also allows small vendors, such as software component providers, service providers and end-users, to provide components for the system. The advantages of this architecture are e.g. independence of one dominant software vendor, possibility to change individual components without changing the whole system and natural evolution of the overall system development due to built-in opportunity for healthy competition. In addition, this approach enables small software vendors to penetrate the ecosystem by selecting a segment that fits their requirements and possibilities. The disadvantages are e.g. the need for fruitful environment, including creation and maintenance of the standardisation and necessary software infrastructure, and lack of one obvious leader of the development. The increasing interest in open source software development model in the past years has shown not only the potential of the model, but also the challenges.

2.1. Open source development model in software business

The open source initiative (OSI) is a non-profit organisation with a global scope for the purposes of education and active support of the benefits of open source, as well as collaboration between partners in the open source community (http://www.opensource.org/). The characteristic of the open source software is that it is developed through code contributions, i.e. in the form of patches (Hertel, Niedner, & Hermann, 2003). This is done through the modification of the existing open source code. The aim is, for example, to increase the quality of the code and add more functions to it. Typically, several programmers, who might work independently, make changes to the code, i.e. to various patches. The open source projects are successful because they are more beneficial economically, since the open source projects involve developers from many different locations and organisations that share code to develop software application for free (Lerner & Tirole, 2002). The authors, Lerner and Tirole (2002), mention that the economic benefits are linked to the openness of the code, which is beneficial, for instance, when complex software is needed where the patches play an important role, since there are many developers involved that support each other in fixing errors. Paulson, Succi, and Eberlein (2004) describe an empirical study, which compares the open source and closed source software projects. They found, for instance, evidence that the open source software increases the creativity of the software developers, what was validated through the metric functions, i.e. software functions that are added over time to the different software applications. Another interesting finding was that open source projects generally have fewer defects than closed source projects, as defects are found and fixed more rapidly. This was validated through the metric amount of functions modified over time as well as the metric functions modified as a percentage of total functions. The OSI mentions that one of their main missions is to support the benefits of the distributed peer review and transparency in the development process since the benefits to be expected from this approach are better quality, higher reliability, more flexibility, lower cost and an end to predatory vendor lock-in. There are also some major players that have built their business around open source projects. Examples of these are Novell, Red Hat, as well as, Internet companies such as Google, Yahoo and Facebook. All these companies have built their business primarily based on open source projects (DeKoenigsberg, 2008). It is well known that in the late 80s the commercialised software was a common way to acquire the software applications for many companies, i.e. closed source projects (Muselli, 2008). After this period, many companies started to switch into the open source software application licences. This provided the companies with the rights of the software such as free use, copy, modification, distribution and modification of the source code. Consequently, the factors to consider when it concerns the open source software for companies are, for example, the different types of software licences, business models, i.e. the value provided to customers and its costs. In addition, the open source publishers are normally not the company that earns profit from the various resources they implement. It is, therefore, crucial for companies to understand the possibilities to earn profit from the open source software, because the

business model differs from the closed software (Teece, 1986). However, commercial developers have established processes to ensure software quality, while the open source software depends mostly on the community, and the defect reporting to achieve some quality level (Syed-Mohamad & McBride, 2008). In addition to this, there are other aspects that a company needs to consider when the open source solutions are taken into account, for instance, the software licence, i.e. the conditions of both the value creation and the revenue that are captured, since the licence type influences the business model. Muselli (2008) identifies four strategic goals that influence the choice of a software licence. These can be mixed together resulting in different final strategies. The first main objective of the strategy is to establish effective development collaboration with a community of users. The second main objective is to diffuse the software, i.e. through free use or free distribution. The third is to take control over competitors. This is done, for example, through licence clauses in order to control the imitation, and finally, the fourth called patrimonial valuation has to do with the possibility of getting direct revenues from license fees. The two first strategies lean towards value creation and the two last into revenue capture. All of these in different combinations result in either leaning towards the value creation or profit earning strategy. The combination between value creation and profit licences strategy provides a wanted balance.

2.2. Standardisation in PLM

According to Duran (2007), there are three categories of standards in use today. The first one is the open standards, i.e. an agreement that people enable that products and systems made by different parties work together (Srinivasan, 2005). The second category is the industrial standards which are technologies that are commonly used, but are not necessarily open and democratically managed by a group of users e.g. Java (Srinivasan, 2005). The last category is de facto standards in use today because of their value or association with other technologies and not necessarily because they were produced by a standard organisation (Srinivasan, 2005). The PLM software vendors usually want to make their products in line with the de facto standard, because this allows them to control the content and the price of their products (Srinivasan, 2005). This conflict adds to the difficulty in providing really interoperable systems despite similarity in PLM suppliers (Fasoli et al., 2011).

Today there are a large number of standards that companies are using, as shown in Figure 1 (Fasoli et al., 2011). The map in Figure 1 shows that different types of standards are needed at different stages of the product's life cycle. The standards have been classified based on the work of Terzi, Bouras, Dutta, Garetti, and Kiritsis (2010) and according to their position among the life cycle and content: product, process or enterprise service (Fasoli et al., 2011). According to Fasoli et al. (2011), the only development that has allowed each of the above systems to really work for their customers is the integration of the Internet into PLM systems and the development of service-oriented architecture (SOA). This feature allows companies to be able to access other companies' product data at any time and from anywhere. A well-designed SOA should use a native XML database-powered metadata repository and data orchestration engine at its core.

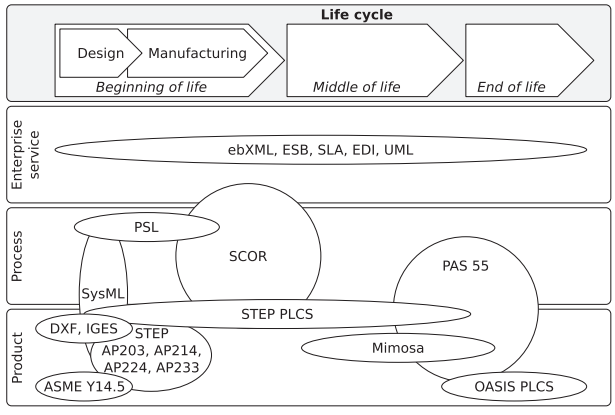

Figure 1. Map of interoperability standards in the product life cycle.

The process of standardisation requires the identification of relevant standards that support the existing ones, i.e. address the hierarchy of existing and evolving standards and how they might be useful to PLM for instance to support the exchange of products, processes, operations and the supply chain information, etc. (Sudarsan, Fenves, Sriram, & Wang, 2005). However, the vendors of PLM systems promise improved product quality, time-to-market, costs benefits, etc., but, for its successful implementation, the use of ontologies becomes important, since it provides critical semantic foundations, which support both interoperability between software platforms and data integration. In addition, it provides the software developers and information scientists with the possibility to concentrate on the various data and information that the system will utilise instead of emphasising functionalities and other aspects of the system themselves (McGuinness, 2005; Smith & Welty, 2001). Sudarsan et al. (2005) mention that the PLM concepts promise consistent and logical integration of all the information through all the different phases of the product life cycle providing the possibility of a more comprehensive decision-making. They, therefore, propose a framework with the intention to support the PLM information needs and semantics of the specific domain. In addition, another work proposes ontology-based semantic standards for PLM (Kiritsis, 2011). In another publication by the same author, he presents an overview of the research done in the area of semantics and ontology-based technologies for product and asset life cycle management (Kiritsis, 2013). It is recommended for purposes of interoperability of industrial systems, such as the PLM applications to use the ISO 15926 (Industrial automation systems and integration—Integration of life-cycle data for process plants including oil and gas production facilities), which is a standard for data modelling and interoperability mainly for process plant data that utilises the Semantic Web technologies, such as RDF, OWL and FOL RuleML. The ISO 15926 standard contains an upper level ontology and reference data ontology. In addition, there are two other ISO standards to be considered such as, ISO 10303 (STEP) and ISO 15288. The first one is a standard that describes how to present and exchange digital product information. The second, i.e. ISO 15288 is a standard for system and software engineering life cycle processes. It provides a common framework, which describes the life cycle of systems created by humans and defines a set of processes and associated terminology within the framework.

However, there are several methodologies or approaches available for developing ontologies, such as those proposed by Uschold and Grunninger (1996), Gomez-Perez (1997), Suguri, Kodama, Miyazaki, Nunokawa, and Noguchi (2001), and Obitko and Marik (2002). There is no well-accepted methodology for constructing an ontology and there is still much work to do for achieving a unified view or acceptance of a methodology for ontology development. In any case, Guarino (1998) mentions that the next thing to model and develop in a system, after the database, is the application programs (software) ontology. There lay the knowledge and business rules from the application domain. These rules provide decision-making support and prognostics possibilities. The rules normally use techniques based on statistical, artificial intelligence, model-based approaches, etc.

One of the major problems when developing user interfaces is to provide the user with queries they can make. If it is not known by the user, the querying and exploration of data will be difficult and the software system will not be used up to its optimal level. Furthermore, since the ontologies represent relationships that exist among the concepts in a domain, it makes it convenient to design a user interface that offers the user the possibility to ask comprehensive, but complex and meaningful questions (Bechhofer, Stevens, Ng, Jacoby, & Goble, 1999). User interfaces need thorough design because they are connected to semantic information such as constraints on classes and relationships in a certain domain (Guarino, 1998). Dahlbom (1995) mentions the Infological equation, which is an important factor when developing user interfaces, since it highlights the importance of the user pre-knowledge. Other important works that highlight the characteristics of user interfaces and human computer interactions are that of Stephanidis, Karagiannidis, and Koumpis (1997), Lewis (1998), Agah and Tanie (2000), and Höök (2000). A detailed discussion about the use of ontologies and the challenges in semantic integration that the domain faces can be found in a paper written by Uschold and Gruninger (2004). Moreover, Kalfoglou and Schorlemmer (2003) provide a state of the

art paper on using ontologies for semantic integration. Another aspect, not present in the map of interoperability standards in the product life cycle presented in Figure 1, is the processes and communications point of view. For this, the standard ISO 15288 (Systems and software engineering—System life cycle processes) provides definitions of concepts and practices. Combining the understanding of the importance of PLM with supporting PLM systems and well-defined and utilised product life cycle processes enable the efficient management of an overall and globally optimised product process.

3. Requirements for PLCS

The interest towards PLCS has increased remarkably when manufacturers have got interested in providing services for their products in order to make themselves less vulnerable to competition and variation in sales. Another factor that has raised the interest in PLCS is the possibility of comparing varying design choices in order to make the product technically superior, environmental friendly and optimal design when the complete life cycle is taken into account. The historical development of PLM, i.e. its evolution, has originated from two directions, (Lee, Ma, Thimm, & Verstraeten, 2008). The first one starts from enterprise management and continues with the material resource planning, enterprise resource planning, customer relationship management and ends in the supply chain management before it evolves into one of the parts of the PLM. The other part has its background in the management of product information, i.e. computer aided design, manufacturing, and engineering (CAD/CAM/CAE), and product data management (PDM) systems. The early systems were limited to engineering information, which required engineering skill and knowledge. The PLM emerged during the late 90s and its main objective is to manage all the information that passes through all phases of the product life cycle such as design, manufacturing, sales and after sales. The evolution of PLM has led to the fragmentation of the overall PLM solution to targeted sub-solutions that do not form a whole and solid PLM system for the enterprise. Often, enterprises have separate systems for the two categories emphasised above, one system or several linked systems for enterprise management and another system or several linked systems for product and design data management. And in the area of product and design data management, it is common that some of the computational tools in the design chain are not included into the centralised data management at all.

However, the computational simulation of the product life cycle has become important as well since it provides solutions for the complexity of decision-making, while taking into consideration the interest of the whole enterprise. The complexity derives from the situation of dealing with various layers of decision-making within a system (Jahangirian, Eldabi, Naseer, Stergioulas, & Young, 2010). The availability and quality of the data for simulation are crucial. Thus, data management is one of the central components of PLCS.

While the application of modelling and simulation in product process is becoming more common and widely used, the challenge of organising, managing and sharing the data related to it is becoming more critical. This is due to the amount of the data, because of scattering of the data into many information systems, because of the variety of the form of the data, and due to the requirement of easy accessibility of the data. The PDM and PLM systems are attempting to answer to the request for centralised information systems for data management, but they are still lacking many important features, such as flexible business and organisational simulation capabilities.

Today many companies use PLM programs in order to handle the design process of their products. Naturally, it could be assumed that this kind of systems should provide a good basis for PLCS. Unfortunately, the current level of PLM systems is far from optimal (Fasoli et al., 2011). In their paper, Fasoli et al. (2011) discuss the features of the most widely used PLM systems and show that there are quite remarkable gaps in these systems when their capability of covering the complete life cycle of products is considered. Another important topic is that the capability of existing PLM systems to pass information between various design and manufacturing programs is very limited. The

reason to this is that typically a great number of programs are used to cover the different aspects of a product's life cycle. As these programs are separate from each other they do not easily pass data between each other. In fact, in some cases the only way of passing information between programs is to do that manually. Clearly, from this follows the most important aspect that needs to be solved when PLCS systems are built, i.e. there is a need to have a working solution for passing data between separate programs and systems. In addition, the level of detail must increase as the product goes through different phases in its life cycle. When defining an ideal PLCS system, after the data issue has been solved, an important aspect to consider is what kind of simulation should be available at the different stages of the product's life cycle. However, the same simulation studies and tests should be possible to carry out either based on more limited data in the early stages or with more reliable data at later stages. The principal design choices, such as, does it run on skis or wheels? What are the loads and stresses of the product for required performance? What are the cost impacts of the selected structures and components on the product life cycle? What kind of maintenance policy is optimal?

4. Semantic data model for PLCS data management

The increasing application of simulation in product development has raised a new bottleneck of data transfer between software applications. Unlike in enterprise data management, the software development of computational tools has focused on single computational methods and algorithms instead of large-scale data modelling and system integration. In the area of engineering simulation software, this has led to wide variety of data models and file formats, even inside a single computational domain, such as finite element method (FEM) for structural analysis. The trend towards integrated design systems and integration of computational tools has emphasised the need for centralised data management and common data models for design and engineering simulation data. The integration of different computational tools in product development is cumbersome. As an example, the exchange of data between different CAD systems has been a challenge for the whole time these tools have been in use in industry. One of the most important reasons for this is the complexity and differences in the internal data presentation of these software applications. The other important reason is the closed nature of commercial tools. Supporting the data models of other software applications, even inside the same computational domain, is difficult. Extending the communication and data integration of software applications to different engineering disciplines and even further to include software applications focusing on business, environment and organisational process management, makes the challenge even more difficult. This is due to the complexity of the data and numerical computation, and the amount of data involved in the process. In addition, supporting data exchange with software applications of other software vendors may not be motivating from the business point of view.

A straightforward solution for the architecture for managing product data, including modelling and simulation data, is to use a common database for the whole organisation involved in the product process. This enables everyone in the process to access the up-to-date data and makes possible the centralised version control of the data. In a common database architecture, the selection of the database technology may become crucial for the flexibility and efficiency of the system. The modelling and simulation data of different engineering disciplines are usually diverse what comes to data models and the amount of data involved in one simulation. For example, the modelling data of a FEM for structural analysis includes tabular form data for geometry discretisation, i.e. the element mesh, and the overall structure of the model data does not vary much from a model to a model. On the other hand, a multibody system (MBS) simulation model does not usually contain large tabular data, and the structure of the data is more varying, which is common to all systemic type of models. According to this, the approach of applying a common database for all the modelling and simulation data requires the database to support all the different data models and forms in the same database.

Semantic data modelling has shown to be an interesting approach for describing complex and heterogeneous data. The semantic data model is based on describing all the data using simple data

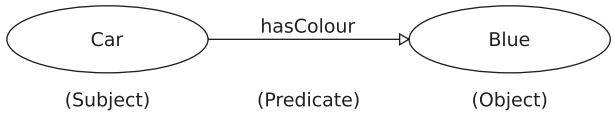

Figure 2. An example of a data triple.

structures that are flexible to describe different forms of information. Describing knowledge in machine-processable form for software applications requires the data to be described in formal manner, but so that it does not restrict the content of the data. For the low-level data model, the form of a data triple has gained popularity. A data triple can be presented as a statement of a subject having a property (predicate) that has a value (object). This is depicted in Figure 2 with a simple example of presenting the information of the colour of a car.

The form of data in semantic data model is usually defined using ontologies, which can be seen as application-specific vocabularies that define the meaning of the data together with the structure and connectivity of the data components (i.e. data resources, such as "Car" and "Blue" represented by ellipses, and relations, such as "hasColour" represented by an arrow in Figure 2). On the other hand, ontologies can be seen as an analogy to class definitions in object-oriented programming. Thus, the data triple is the low-level data structure for data nuggets, and ontologies add a data description layer to the system. Due to the form of a data triple, i.e. having a connecting relation (predicate) between two resources (subject and object), mapping pieces of data is a built-in feature of the semantic data model. Semantic data models are often visualised as data graphs. The semantic data model is especially suitable for representing system simulation data, which is typically strictly defined and can be naturally visualised as data graphs. As an example of the semantic approach, the application of semantic data model on MBS simulation modelling data management is discussed in more detail in Kortelainen and Mikkola (2010) and Kortelainen (2011).

4.1. Structure for product life cycle data model

Product life cycle data includes e.g. technical data of the product, such as design and manufacturing data, but also other data related to the product process and the business. Thus, there is no one complete data model for collecting and storing the product life cycle data, but a set of concepts and technologies that enable creating and modifying the date model based on the requirements for the specific product process and the users.

The structure and details of the data model for a specific product process depends, among other things, on the product or engineering domain, organisation and its structure, existing software tools and data systems, and working practices and processes. Integrating e.g. the engineering software tools and data into the common data model requires mapping software application specific data into the product model. This may mean that tens of different kind of data models and data file formats are managed and mapped into the common product data model. Figure 3 illustrates the complexity of the product life cycle data from the engineering software application integration point of view. An example of defining a generic MBS domain ontology, i.e. the data model for MBS modelling data, using Semantic Web technologies and open source ontology modelling tools is discussed in detail in Kortelainen and Mikkola (2010) and Kortelainen and Mikkola (2013).

In Figure 3, grey rectangles inside the semantic data management system represent domain ontologies. Arrows with open heads inside the semantics data management system represent data mappings and possible data conversions between different ontology representations of the product data. Arrows with shaded heads represent data links between the semantic data management system and external software applications or data systems. Rectangles with dashed lines represent grouping of domain ontologies or software applications.

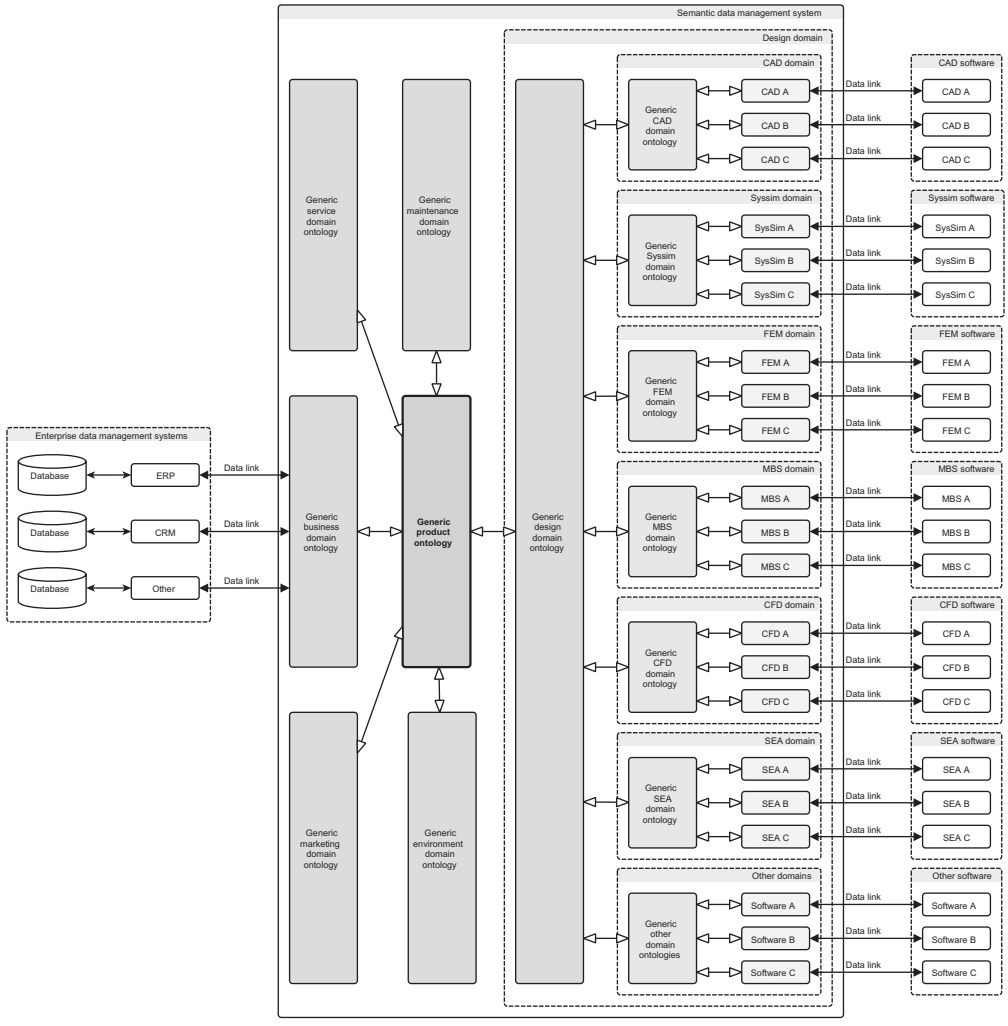

Figure 3. Illustration of the semantic data model architecture for product process data.

The semantic data management system that is holding the semantic data management system is presented as the large rectangle in the centre of the figure. The core of the product data model is the generic product ontology (highlighted with bold text and thick line of the rectangle in the figure), which defines the upper-level concepts of the product data model. The product ontology is defined in more detail in different generic domain ontologies, such as generic design domain ontology. The generic domain ontologies are further defined in more detail in the generic sub-domain ontologies, such as generic CAD domain ontology for the design domain. The generic sub-domain ontologies define the common concepts and components of the particular sub-domain. On the right side of Figure 3, examples of engineering software applications that are used in product design are presented. The data from the third-party software applications is linked with the semantic data management system. To simplify the integration of especially third-party software applications, the semantic data management system has software application specific ontologies so that there is no need for data transformations in the data transfer between the software application and the semantic data management system. The software application specific data is mapped to the general domain ontology and via that to the overall product model. There may be more ontology layers between the generic product ontology (the core of the product model) and the individual software application specific ontologies. Similar data model structure can be defined for all the domain ontologies, such as the maintenance and marketing domain ontology.

Figure 3 shows that the overall data model for product data may become very large and complex. On the right side of the figure are some possible engineering software applications, such as CAD and FEM. It should be noticed that it is common that there are several different software applications present in the product process. Subcontractors may have different tools than the process owner, and even inside a large company, different units and departments can use different tools for the same purpose. The data transfer between the semantic data management system and the individual software applications can be implemented e.g. file-based using file formats that the software application supports, or the data transfer can utilise some other communication method, such as interprocess communication sockets.

The semantic data model has a built-in feature of mapping data details, i.e. linking data pieces in the data model is straightforward. To take advantage of this feature, all data mappings between e.g. the data of different software applications and data transformations should be done in the semantic data management system. The data from software applications, such as FEM and MBS software applications, should be read as is into the semantic database. The semantic ontology related to a specific software application should be in principle the same as the native data model of that software application, thus there would be no need for any data transformations in the data link. To map the software application specific data with the data from other software application data should be done via general domain ontology to minimise the propagation of data model changes into the whole product data model, and thus, to minimise the maintenance work for the data management system.

The general engineering sub-domain ontologies as well as the general design domain ontology should utilise standardised data models. An example of a standardised data model for product structure data is ISO 10303 AP 203 *Configuration controlled 3D designs of mechanical parts and assemblies*. An example of the data mappings between two different engineering software application data is illustrated in Figure 4.

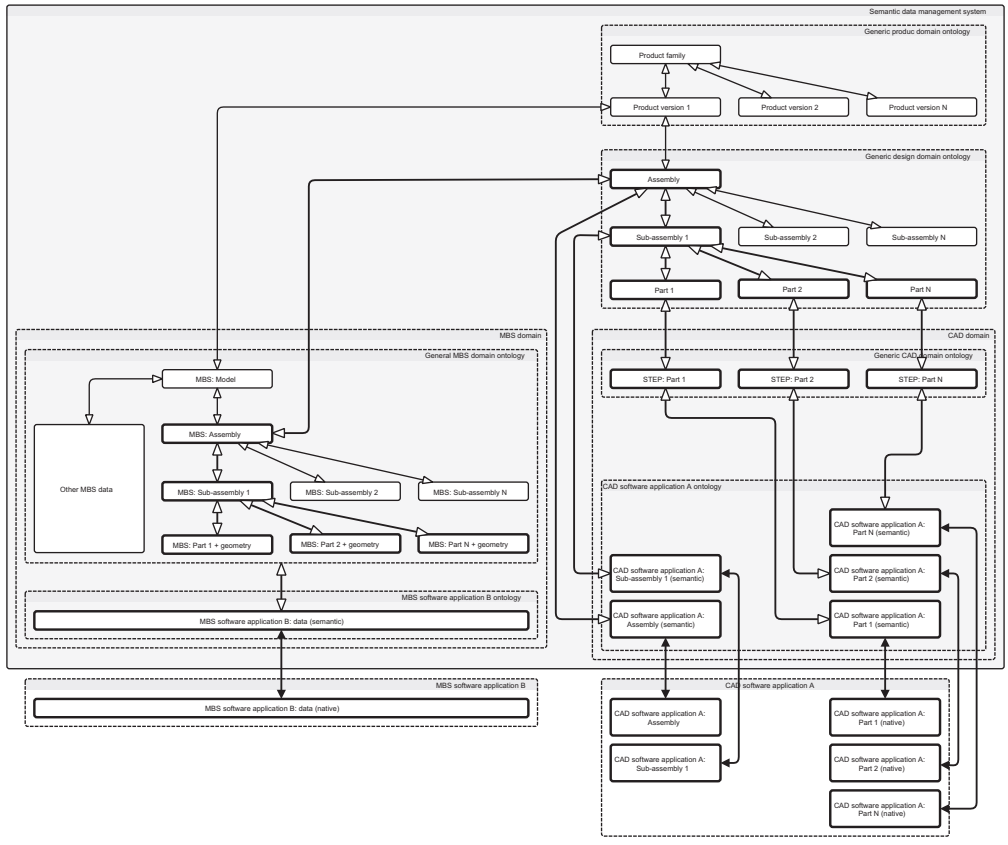

Figure 4. Illustration of the data mapping for an MBS model in the semantic data management system.

In Figure 4, the concept of mapping CAD geometry data with MBS model data is illustrated. A CAD model defines the physical dimensions and shape of the components e.g. in a mechanism assembly. This information can be used as the starting point for defining an MBS model of the mechanism. The CAD geometry defines the dimensions of the system, design of the system parts, locations and orientation of the joints, and together with the material (density) information, the mass properties of the parts of the mechanism. The CAD model can be seen as the primary information document for these properties. In Figure 4, the primary CAD information from CAD software A is linked with the data management system (lower right corner of the graph). The primary data is read into the CAD software A specific ontology in the system. The CAD model data is then linked with the generic CAD domain ontology and the generic design domain ontology. In the similar manner, the primary MBS modelling data comes from the MBS software B. The primary data are read into the MBS software B specific ontology in the data management system. Further, the data are mapped with the general MBS ontology. The CAD data are then linked with the MBS data in the semantic data management system in the general domain ontology level. In this approach, changes e.g. in the MBS software specific data, such as changing the MBS software, do not propagate further than the software application specific ontology and the ontology mappings between the software application specific ontology and the general MBS domain ontology.

How the ontologies are described depends on the data management system implementation. In systems that rely on the Semantics Web technologies, the ontology definition can be provided e.g. by utilising Resource Description Framework (RDF) specification. In other systems, such as the Simantics platform described in more detail the next section, the ontology is defined in the system specific form (in case of the Simantics platform, the ontology can be defined as a Simantics graph format layer0).

4.2. Example of open source development model for data management solution

The request for centralised data modelling and management, and the need for common mechanisms for describing complex and heterogeneous data and domain knowledge initiated the development of the Simantics platform (Karhela, Villberg, & Niemistö, 2012), a software platform and a semantic database system for simulation data management (http://www.simantics.org). The Simantics platform utilises the server-client architecture. The client and the graphical user interface of the system (Simantics Workbench) is built on the Eclipse platform, an open source software platform used for many software development tools and application (http://www.eclipse.org). The server of the system (Simantics Core) is a triplestore type of a database management system that holds the data in the form of data triples. The server and the client form an entity in such a way that the data mass is stored into the server side but the main functionality of the system is on the client side. The client system contains functional components, such as editors, visualisation components and different kind of data views. The simulation functionality of the system is added by connecting so-called plug-in components to the system. Plug-ins components are e.g. numerical solvers or other external components for producing or modifying the data. In addition, the Simantics platform contains a common type and data interface system (Simantics Databoard). The software architecture of the Simantics platform is illustrated in Figure 5. The overall design and the selected software components make the Simantics platform both a software platform and a software development environment for semantic applications. The common software platform for application implemented on the Simantics platform provides unified look and feel, which has been one of the positive arguments for e.g. integrated design environments in general. On the other hand, the common semantic database system of the Simantics platform provides the foundation for data integration and semantic data analyses. The Simantics platform is licensed under an open source software license (Eclipse Public License).

The open source development model has been utilised in several areas of the development of the Simantics platform. The platform itself is open source software and it utilises another open source software platform, Eclipse, as one of its main components. In addition to the open source

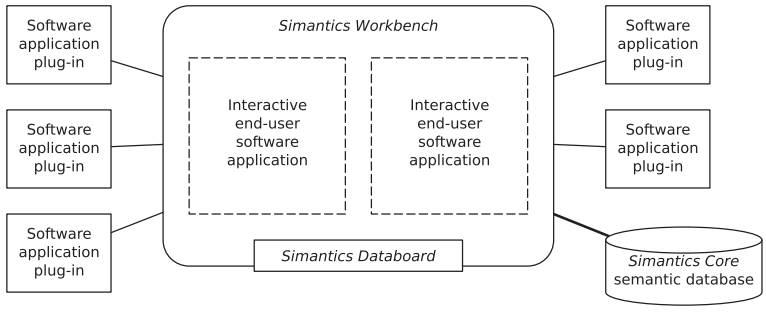

Figure 5. The software architecture of the Simantics platform.

licensing of the Simantics platform, an open source community, the THTH Simantics Division (https://www.simantics.org/simantics/about-simantics/thth-simantics, https://www.simantics.org/members/index.php/Main_Page), is initiated around the platform development. The community is organised into a non-profit association that holds all the rights of the software and manages the development of the platform. One of the main objectives of the THTH Simantics Division is to form a healthy and vital ecosystem around the Simantics platform and to accelerate the development of simulation software solutions.

The THTH Simantics Division has members that pay a participation fee for the association and, as the return, have a position in the association's board that makes decisions on the development directions of the the Simantics platform. In addition, the members of the association have access to some specific software components that are not included into the open source release of the Simantics platform. The plain Simantics platform is publicly available as source code through the SVN service. The licensing of the Simantics platform enables both open source as well as closed source software applications to be developed on the Simantics platform.

The actual software development of the Simantics platform is done mostly in the software application development projects, such as the development of the APROS software (http://www.apros.fi/en/). The licensing model for the Simantics platform requires that the modifications done for the Simantics platform itself have to be published under the same license as the platform itself. The software development process for the Simantics platform follows the principles of the Scrum process (http://www.scrum.org/).

5. Discussion

The developments of the open source systems and their concepts provide the companies with new business models on how to attain the latest technology. It is, therefore, important to understand its characteristics and how companies can benefit from this development, what is a reality for all companies using ICTs for different purposes. Consequently, the popularity of the open source applications is rising, and has become a subject of real interest for many organisations. This development has taken place since the commercialised applications are considered to be having problems, such as making the development expensive and inefficient since they are complex, difficult to extend, update and change. The user interfaces are problematic as well, because they are not self-adaptable, but standardised. Consequently, the standardisation is an important factor within the software development since standardising the data models and interfaces can enable forming an ecosystem for different actors, such as small software vendors to provide their solutions for some parts of the overall system. The Eclipse platform development is a good example of this. Furthermore, the cost of the above-mentioned applications is increasing, respectively. Many companies adapt their already existing software systems with partly open source modules for the emerging needs of their business. This shows that the open source philosophy is here to stay and something that companies have to consider when developing their ICT applications. For instance, companies implementing open source software into their already existing software need to

consider if the open source organisation/company is a software distributor, a software producer or a service provider, what is mentioned among other aspects as well in Munga, Fogwill, and Williams, (2009). It is, therefore, even convenient to analyse the business model of the company, while using, for example, a framework developed by researchers, such as Munga and Fogwill (2009) or Holck and Zicari (2006). This enables one to understand the benefits of using the open source software and the value for its business model. In addition, to integrate open source software entails the understanding of the specific needs of the company's business. Moreover, it is vital to be able to answer such questions as how the open source software will be used (value offering), how the open source software will affect the other parts of business (the market), how the open source software will be implemented in the company, the connected costs and how the maintenance of it will be performed, etc. The open source is an important phenomenon since it provides control over the software code, which provides possibilities to update it and make adequate changes that fit into the specific company. The reason of the former mentioned is because of the characteristics of the open source, for instance, the authorisations provide the right to use the software code, examine, change and distribute the software to other partners in the specific companies supply chain. Continuously, the acquired open source software will not be updated or new versions appear forcing the company to buy newer versions of the software. The update of the open software is done by the community and it can be downloaded for free. Another important aspect deals with the risk of the software to disappear if the original developers of the software stop producing it, since the open software is owned by the community. In addition, the access to emergent technologies is simplified due to the fact that innovations in the open source software are normally on the forefront of research. All the above-mentioned aspects are important for software in any domain as well as for the PLM software. Consequently, the use of open source provides a new business model enables a new business model to develop PLM and PLCS software and includes factors into the application in a faster way than in a traditional business model, i.e. proprietary model and the product can be faster in use as well with less costs.

It is also known that the recent developments conducted in the sphere of the ICTs and their use on the open source software, especially Web technologies, such as the Web 2.0 and the Web services, etc. are providing software applications and their organisations with new opportunities (Campos, 2009). In the case of PLCS, there are new possibilities to integrate various processes and different systems. In addition, it is also known that the PDM becomes more adaptable and flexible if it is running on a web infrastructure. This has also been acknowledged in Lee et al. (2008). The developments of the latest ICTs, such as the Web Services, provide new ways for integration of the companies' data, information and even software applications. All this results in more comprehensive decisions which can be taken, considering the whole life cycle of a product, leading to advantages in productivity, and competitiveness for the organisation.

6. Conclusions

The Open Source Solutions provide the companies with not only various opportunities, but also challenges that need to be carefully considered, such as licences, community to choose, etc. to achieve a successful implementation. The capability of carrying out PLCS has raised increasing interest due to a number of reasons. This may specially refer to companies which are trying to improve their design process and also increase their role in providing services for their products and thus become less sensitive to changes in the market. In spite of the growing interest, it seems that the capability of carrying out PLCS today has not really reached a high level. It seems that the biggest obstacle is the lack of reliable data for simulation since companies might have the needed data but this data is scattered between numerous programs and tools that are used for different purposes and that are not easily transferred between these systems, i.e. integrated. It is, therefore, believed that semantic data structures and ontologies can support the software simulation of the product life cycle, what has become important because they give solutions to the complexity of decision-making. In addition, it is believed that companies can increase and optimise their requirements specification for the PLCS through the understanding of the ontologies, i.e. the data and information that needs to be

gathered, stored and transferred into the different parts of the software business logic for purposes of simulation. This together with the developments of the Web technologies, such as the Semantic Web and the Web services, provide new possibilities to integrate heterogeneous data as well as distributed applications, which offers possibilities to achieve the full simulation of the entire enterprise resulting in more comprehensive decision-making.

Funding
This research was supported by the project Computational methods in mechanical engineering product development (SIMPRO), funded by Tekes—the Finnish Funding Agency for Innovation and the participating organisations.

Author details
Jaime Campos[1]
E-mail: jaime.campos@lnu.se
Juha Kortelainen[2]
E-mail: juha.kortelainen@vtt.fi
Erkki Jantunen[2]
E-mail: erkki.jantunen@vtt.fi
[1] Department of Informatics, Linnaeus University, Växjö, Sweden.
[2] VTT Technical Research Centre of Finland, Espoo, Finland.

Citation information
Cite this article as: Industrial open source solutions for product life cycle management, J. Campos, J. Kortelainen & E. Jantunen, *Cogent Engineering* (2014), 1: 939737.

References

Agah, A., & Tanie, K. (2000). Intelligent graphical user interface design utilizing multiple fuzzy agents. *Interacting with Computers, 12*, 529–542. doi:10.1016/S0953-5438(99)00022-3

Bechhofer, S., Stevens, R., Ng, G., Jacoby, A., & Goble, C. (1999). Guiding the user: An ontology driven interface. In *Proceedings user interfaces to data intensive systems* (pp. 158–161). http://dx.doi.org/10.1109/UIDIS.1999.791472

Campos, J. (2009). Development in the application of ICT in condition monitoring and maintenance. *Computers in Industry, 60*(1), 1–20. http://dx.doi.org/10.1016/j.compind.2008.09.007

Dahlbom, B. (Ed.). (1995). *The infological equation: Essays in honor of Börje Langefors*. Gothenburg: Gothenburg University, Department of Informatics.

DeKoenigsberg, G. (2008). How successful open source projects work, and how and why to introduce students to the open source world. In *21 Conference on software engineering education and training (CSEET 2008)* (pp. 274–276). Charleston, SC, USA.

Duran, F. (2007). *Interoperability and standardization between PLM systems: How different vendors can exchange information across different operating systems and programming languages*. ESI4628 – IE Computer Applications. University of Central Florida.

Fasoli, T., Terzi, S., Jantunen, E., Kortelainen, J., Sääski, J., & Salonen, T. (2011, May 2–4). Challenges in data management in product life cycle engineering. In J. Hesselbacj & C. Herrmann (Eds.), Glocalized solutions for sustainability in manufacturing: Proceedings of the 18th CIRP International Conference on life cycle engineering (pp. 525–530). Technische Universität Braunschweig, Germany: Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-642-19692-8_91

Fernandez, O., Labib, A. W., Walmsley, R., & Petty, D. J. (2003). A decision support maintenance management system: Development and implementation. *International Journal of Quality Reliability Management, 20*, 965–979. http://dx.doi.org/10.1108/02656710310493652

Glotzer, S., Kim, S., Cummings, P., Deshmukh, A., Head-Gordon, M., Karniadakis, G., … Shinozuka, M. (2009). *International assessment of research and development in simulation-based engineering and science* (Report). Baltimore, MD: World Technology Evaluation Center, WTEC.

Gomez-Perez, A. (1997). Knowledge sharing and reuse. In J. Liebowitz (Ed.), *Handbook of applied expert systems* (Vol. 10, pp. 1–35). Boca Raton, FL: CRC Press.

Guarino, N. (1998, June 6–8). Formal ontology and information systems, formal ontology in information systems. In *Proceedings of the 1st International Conference on formal ontology in information systems (FOIS98)* (pp. 3–15). Trento, Italy.

Hertel, G., Niedner, S., & Hermann, S. (2003). Motivation of software developers in the F/OSS projects: An Internet-based survey of contributors to the Linux kernel. *Elsevier, Research Policy, Open Source Software Development, 32*, 1159–1177.

Holck, J., & Zicari, R. V. (2006). *A framework analysis of business models for open source software products with dual licensing*. CBS/INF Working Paper, No. 1, January 2007. Copenhagen Business School, Department of Informatics.

Höök, K. (2000). Steps to take before intelligent user interfaces become real. *Interacting with Computers, 12*, 409–426. http://dx.doi.org/10.1016/S0953-5438(99)00006-5

Huber, L. E., & Carlson, H. R. (1990, May 20–23). Information. The undermanaged resource. In *Proceedings of the International Industrial Engineering Conference* (pp. 17–22). San Francisco, CA, USA.

Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L. K., & Young, T. (2010). Simulation in manufacturing and business: A review. *European Journal of Operational Research, 203*(1), 1–13. http://dx.doi.org/10.1016/j.ejor.2009.06.004

Kalfoglou, Y., & Schorlemmer, M. (2003). Ontology mapping: The state of the art. *The Knowledge Engineering Review, 18*(1), 1–31. http://dx.doi.org/10.1017/S0269888903000651

Karhela, T., Villberg, A., & Niemistö, H. (2012). Open ontology based integration platform for modeling and simulation in engineering. *International Journal of Modeling, Simulation, and Scientific Computing, 03*, 1250004, 36 pp. http://dx.doi.org/10.1142/S1793962312500043

Kiritsis, D. (2011). Closed-loop PLM for intelligent products in the era of the Internet of things. *Computer-Aided Design, 43*, 479–501. http://dx.doi.org/10.1016/j.cad.2010.03.002

Kiritsis, D. (2013). Semantic technologies for engineering asset life cycle management. *International Journal of Production Research, 51*, 7345–7371. http://dx.doi.org/10.1080/00207543.2012.761364

Kortelainen, J. (2011). *Semantic data model for multibody system modelling* (doctoral thesis). VTT Publications 766, 119 p. + app. 34 p. ISBN: 978-951-38-7742-2.

Kortelainen, J., & Mikkola, A. (2010). Semantic data model in multibody system simulation. *Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-Body Dynamics, 224*, 341–352. http://dx.doi.org/10.1243/14644193JMBD257

Kortelainen, J., & Mikkola, A. (2013). Semantic restrictions and rules in applications of multibody dynamics. *The Journal of Engineering with Computers*. Retrieved September 29, 2013, from http://dx.doi.org/10.1007/s00366-013-0326-x

Lee, S. G., Ma, Y.-S., Thimm, G. L., & Verstraeten, J. (2008). Product lifecycle management in aviation maintenance, repair and overhaul. *Computers in Industry, 59*, 296–303. http://dx.doi.org/10.1016/j.compind.2007.06.022

Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics, 50*, 197–234.

Lewis, M. (1998). Designing for human-agent interaction. *AI Magazine, 19*, 67–78.

McGuinness, D. L. (2005). Ontologies come of age. In D. Fensel, J. A. Hendler, H. Lieberman, & W. Wahlster (Eds.), Spinning the Semantic Web: Bringing the World Wide Web to its full potential (pp. 171–191). Cambridge, MA: MIT Press. Retrieved from http://www.google.com.ua/books?id=zQ34EoZO2IYC&printsec=frontcover"\l"v=onepage&q&f=false http://www.google.com.ua/books?id=zQ34EoZO2IYC&printsec=frontcover#v=onepage&q&f=false

Munga, N., & Fogwill, T. A. (2009, May 6–8). Analysis of the value that open source contributes to business models. In IST-Africa 2009 Conference and Exhibition. Kampala, Uganda.

Munga, N., Fogwill, T., & Williams, Q. (2009). The adoption of open source software in business models: A Red Hat and IBM case study. In B. Dwolatzky, J. Cohen, & S. Hazelhurst (Eds.), *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists* (pp. 112–121). New York, NY: ACM. doi:10.1145/1632149.1632165

Muselli, J. (2008, September 21–24). Open source software publishers business models: The strategic role of business licenses. In Proceedings of the 2008 IEEE ICMIT. Bangkok, Thailand.

Nagarur, N. N., & Kaewlang, J. (1999). An object-oriented decision support system for maintenance management. *Journal of Quality in Maintenance Engineering, 5*, 247–257.

Obitko, M., & Marik, V. (2002). Ontologies for multi-agent systems in manufacturing domain. In *Proceedings 13th international workshop on database and expert systems applications* (pp. 597–602). Aix en Provence, France.

Paulson, J. W., Succi, G., & Eberlein, A. (2004). An empirical study of open-source and closed-source software products. *IEEE Transactions on Software Engineering, 30*, 246–256. http://dx.doi.org/10.1109/TSE.2004.1274044

Smith, B., & Welty, C. (2001). FOIS introduction: Ontology-towards a new synthesis. FOIS '01 Proceedings of the International Conference on formal ontology in information systems (Vol. 2001, pp. 3–9). New York, NY: ACM. doi:10.1145/505168.505201

Srinivasan, V. (2005, July 11–13). Open standards for product lifecycle management. In A. Bouras, B. Gurumoorthy, & R. Sudarsan (Eds.), *Proceedings of the International Conference on product lifecycle management: Emerging solutions and challenges for global networked enterprizes (PLM'05)* (pp. 475–484, Inderscience Enterprizes Ltd). Lyon: Lumière University.

Stephanidis, C., Karagiannidis, C., & Koumpis, A. (1997). Decision making in intelligent user interfaces. In Proceedings of the 2nd International Conference on intelligent user interfaces (pp. 195–202). New York, NY: ACM. doi:10.1145/238218.238323

Sudarsan, R., Fenves, S. J., Sriram, R. D., & Wang, F. (2005). A product information modeling framework for product lifecycle management. *Computer-Aided Design, 37*, 1399–1411. http://dx.doi.org/10.1016/j.cad.2005.02.010

Suguri, H., Kodama, E., Miyazaki, M., Nunokawa, H., & Noguchi, S. (2001, May 28–June 1). Implementation of FIPA ontology service. In Workshop on ontologies in agent systems, 5th International Conference on autonomous agents, Montreal, Canada.

Syed-Mohamad, S. M., & McBride, T. (2008). A comparison of the reliability growth of open source and in-house software. In 15th Asia-Pacific Software Engineering Conference (APSEC08). Beijing, China.

Teece, D. J. (1986). Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy, 15*, 285–305. http://dx.doi.org/10.1016/0048-7333(86)90027-2

Terzi, S., Bouras, A., Dutta, D., Garetti, M., & Kiritsis, D. (2010). Product lifecycle management – From its history to its new role. *International Journal of Product Lifecycle Management, 4*, 360–389. http://dx.doi.org/10.1504/IJPLM.2010.036489

Turban, E., Leidner, D., McLean, E., & Wetherbe, J. (2007). *Information technology for management – Transforming organisations in the digital economy.* Hoboken, NJ: Wiley.

Uschold, M., & Grunninger, M. (1996). Ontologies: Principles, methods and application. *Knowledge Engineering Review, 11*, 93–136.

Uschold, M., & Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *SIGMOD Record, 33*, 58–64.