A Lightweight Approach to Manage Engineering Parameters in Mechatronic Design Processes

Lukas Weingartner, Peter Hehenberger, Michael Friedl, Andreas Kellner, Stefan Boschert, Roland Rosen

A Lightweight Approach to Manage Engineering Parameters in Mechatronic Design Processes

Lukas Weingartner[1], Peter Hehenberger[1], Michael Friedl[2], Andreas Kellner[1], Stefan Boschert[3], and Roland Rosen[3]

[1]Institute of Mechatronic Design and Production, Johannes Kepler University Linz, Austria
[2]Institute of Machine Design and Hydraulic Drives, Johannes Kepler University Linz, Austria
firstname.secondname@jku.at

[3]Siemens Corporate Technology Munich, Germany
firstname.secondname@siemens.com

Abstract. In mechatronic design processes the exchange of information between stakeholders from different disciplines is essential to enable simultaneous engineering and a successful integration of domain specific subsystems. Although there is a comprehensive range of methodologies and tools to support collaboration, intentions to implement a central data management platform covering all stakeholders often do not succeed. Reasons are the heterogeneous model landscape, the variety of stand-alone authoring tools, departments', disciplines' or companies' boundaries and a lack of flexibility of established solutions regarding the support of unpredictable and quickly changing design processes. This paper focuses on the management of individual parameters and parameter instances (values) within a multi-disciplinary development team. The presented lightweight approach can extend existing methods and data management infrastructure by adding functionalities to provide access to individual parameters using a dedicated database. The procedure is shown by the example of a technical mechanism.

Keywords: Mechatronic design process · Engineering parameter management · Parameter database · Data exchange process · Model management

1 Introduction

During the course of the multi-firm project SyMMDe [1], mechatronic design processes of four different companies were analyzed in detail and presented in the form of model dependency maps [2]. From this point of view, every model needs input parameters and generates results (e.g., output parameters in various forms) for a particular purpose. Heterogeneous models on various hierarchical levels are usually created and implemented within specific authoring systems (e.g., CAx-tools) by different stakeholders and experts (e.g., from the disciplines of mechanics, electronics and software). In most cases, models (e.g., parametric geometry models) are stored using a kind of file-based model storage. There is a wide range of software solutions

(e.g., file systems, “PDM - product data management” systems, “ECM - enterprise content management” systems) to manage (mostly proprietary) files within a central platform covering essential functionalities like versioning, access management, configuration management, and much more. For pragmatic reasons like performance advantages and interoperability, individual model storage systems can be settled locally at the expert’s computer or close to the authoring systems (e.g., team data management). From our observations, there is usually no overall (e.g., company-wide) consistent model backbone for all kinds of models. Although tailor-made configured PLM systems can meet these challenges in principle, in many cases their potential has not yet been exhausted, since only a limited scope (e.g., PDM aspects) of possible functionalities is used, without covering all stakeholders and models.

During mechatronic modeling processes, different types of parameters appear. As discussed in [3], a discipline-specific component can consist of a user-defined part described by design parameters and a standardized part described by configuration parameters. In addition to product descriptive parameters (including, for example, process parameters in process models) there are many other parameters that are relevant for the various discipline-specific models (e.g., regarding rules and know-how, or solver parameters and settings in simulation processes). The subset of parameters that influence more than one stakeholder or rather multiple models play a significant role across the product development process. It is essential that all stakeholders gain a common understanding of these important parameters. Globally unique and therefore complex names and descriptions of all parameters used are usually inconvenient. Experts from the different disciplines naturally have varied interpretations and views at the system and demand their own naming conventions. If the parameter management concept fails to unify these different views or to regulate access on the parameters, redundancies and consistency problems may follow. To achieve a close integration of domain specific subsystems, domain experts with diversified knowledge and skills often are in typical conflicts of interests. Conflicting objectives and targets concerning subsystems force stakeholders to negotiate about parameters (e.g., available design space). Therefore, parameter and parameter values can change several times due to improvements or coordination between stakeholders. Such engineering changes can have various triggers (e.g., unforeseeable events) and arise throughout the whole design process [4]. To enable an integration of components and solutions developed from several disciplines, substantial main parameters must be defined and made available to all stakeholders. Lacking access to a central parameter management system often leads to mainly spread, unregulated, and not standardized information exchange (e.g., using telephone, post-it notes, social media, email or additional stand-alone groupware or project management applications). Despite the availability of PLM tools, one commonly used mostly manual and file-based method is to manage parameters in rather simple lists with ordinary spreadsheet tools. Although this is a proper solution for documentation purposes, problems occur if multiple users have frequent change requests, as in the case of usual iterations during design processes. Fig. 1 shows a comparison of the basic mechanisms with a central database (type 3) and without (type 1 and 2).

The aim of this paper is to introduce and implement a methodology that covers the following needs and requirements related to parameter management as observed during typical interdisciplinary development processes at our industrial examples: (i)

Access to individual parameters across multiple models (e.g., from proprietary files), without having to use the respective authoring system, e.g., by different stakeholders across multiple departments/locations. (ii) Multi-user access including rights management. (iii) Management of parameters including capabilities to keep an overview by using metadata and attributes or functionalities like filtering and highlighting of major and product defining parameters. (iv) Representation of relationships between parameters and possibilities to analyze impacts in the case of changes. (v) Traceability and history of changes including the possibility to add knowledge regarding decision making (e.g., rules) and documentation about decisions made (e.g., implicit assumptions, findings, reasons for occurring iterations). (vi) Several models (e.g., on different hierarchical levels created in specific authoring tools) should be able to access individual parameters using simple adapters and interfaces without the need of an additional overall authoring tool. (vii) Widespread introduction of the parameter management system should be possible, e.g., by using thin clients or web interfaces. (viii) Ability to extend and customize the platform using standardized tools or common programming languages. (ix) No adverse impacts and limitations on the established development process and flexibility to deal with highly dynamic changes within the development process.

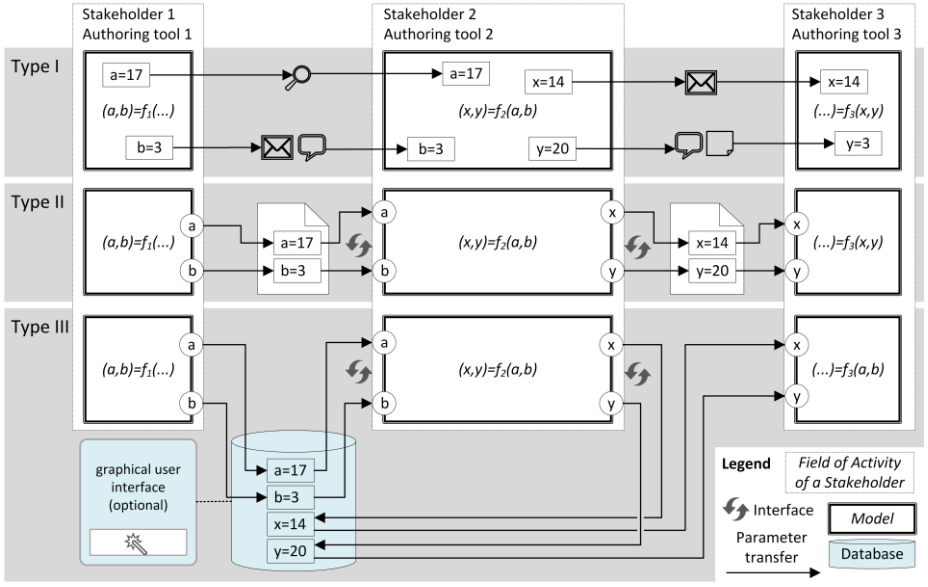

Fig. 1. This example shows three models (f_1, f_2, f_3) and three basic types of information exchange (parameter values of a, b, x, y) between different stakeholders and authoring tools: (I) Manual exchange of parameter values. (II) File-based exchange of parameter values. (III) Parameter value exchange using a central repository with an additional graphical user interface.

2 Discussion of established methodologies and tool concepts

Badin et al. [5] present a "KCModel" methodology to "Capitalize, Trace, Re-use and ensure the Consistency" of technical data using centralized knowledge management. Parameters are shared between different models and a consistency management (e.g., expert rules) between configurations can detect conflicts in parameter values. Other concepts and tools to support collaborative design based on web technologies were already introduced back in 2002 by Riboulet et al. [6]. The presented application "CoDISS" (cooperative data & information sharing system) enables communication (e.g., sharing of knowledge, concepts and related parameters) between various models and different stakeholders [7].

Although it's difficult to draw a clear line in between, there are mainly three categories of established tool concepts that are used successfully in the industry to manage product data in a central repository accessible to users and experts from various disciplines. Similar differentiations are discussed in the work of Panchal et. al. [8, 9]. Differentiations between PDM and "SDM - Simulation Data Management" [5] as well as weaknesses regarding management of fine grained parameters are also discussed in [10]. Each of the following three options has strengths and weaknesses in different applications.

The *first option* ("model backbone") are IT systems that focus on file-based product data management to enable collaboration between the stakeholders. There are powerful options to classify and structure the files by using metadata or structural modeling, but their focus is not to manage fine grained information like individual parameters within the files. The most commonly used concepts and tools can be grouped under the headings ECM, "DMS - document management system" and PDM as a part of "PLM - product lifecycle management". Prominent representatives in the field of mechatronic engineering tools are, e.g., Siemens Teamcenter® [11] or Dassault Systèmes ENOVIA® [12]. PLM systems (as an extension of PDM systems) even provide possibilities to extract and exchange fine-grained information and parameters from files, e.g., to initiate and execute customized and purpose-built workflows or simulation processes. Some authoring systems can be seamlessly integrated including bidirectional associativity of attributes and parameter values. However, the common fundamental PLM tool concepts are not designed or intended to support highly dynamic engineering processes of different models *on a parameter level*. Functionalities to analyze changes of parameter values and their impact within the model landscape are the aim of additional tools (e.g., requirements management and systems engineering applications), that can be connected to or installed within the PLM system. Clearly defined processes – even on parameter-level – are well-supported by tool adapters and workflows customized by PLM experts. To set up quick changing processes (e.g., experimental simulation processes) domain experts need ability and flexibility to implement specific processes by their own, e.g., using an end-user-friendly interface.

This leads to a *second option* ("model-based workflow management"), that is provided by systems and tools that focus on models and parameter exchange to realize workflows. So-called "PIDO - process integration and design optimization" frameworks or SDM solutions have to deal with a huge amount of individual parameters. Within a framework workflows consisting of interconnected (mainly

external) models can be defined and executed. This means that parameters from different sources can be connected and exchanged within the tool environment. In the area of simulation processes, the second option is well suited for a limited number of users. The central simulation process model within the framework (usually a proprietary system platform) allows to run preconfigured simulations and calculations including parameter studies or optimization. The simulations can run at the local machine, without the need to connect to a server, which causes improved performance in some cases. Prominent simulation environments like Siemens LMS Imagine.Lab™ Amesim with Sysdm [11] or MathWorks® MATLAB® Simulink® [13] provide extensive possibilities to manage parameters within the associated environment.

A *third option* ("descriptive system model") is to build a model that contains essential knowledge (parameters, relationships, rules). Such approaches can be implemented using (model based) systems engineering methodologies and modelling languages (e.g., SysML) [14]. A major advantage of general-purpose modeling languages like SysML is the standardization, which is a basis for a tool-neutral integration. A well-known problem is that systems engineering tools follow a generic approach and must be adapted to fit concrete applications. Like a survey [15] pointed out, the large learning curve to understand SysML is a large inhibitor. In traditional engineering disciplines, only few domain experts have sufficient knowledge in this kind of modeling, mainly inspired by computer science (e.g., UML). From our observations, the role of systems engineers is not widespread introduced in industry at the present time. In the recent past, tool vendors developed modeling applications that enable analysis and execution of diagrams, e.g., SysML parametric diagrams, including expressions in form of constraints [14] and management of instances.

As a conclusion, there are many individual solutions but no universal parameter management approach, which meets all the requirements as stated in chapter 1. The objective was now to develop a powerful but not overloaded ("lightweight") approach, that can be adapted and used productively with reasonable implementation effort, e.g., as a supplement to established design processes.

3 Generic methodology for lightweight parameter management

Bearing in mind that parameters only get a meaning and a significance in conjunction with models, the following approach is based on a separation of models and parameters and values. There are advantages (e.g., regarding transparency and simplified access to parameters) if models and parameters are considered in separate terms and also get technically implemented separately (e.g., in different databases). As mentioned above, models are nowadays managed successfully using different methodologies and tools (e.g., PLM systems) within a "model backbone". In addition, to manage fine grained parameters, a "parameter backbone" is introduced. In contrast to other approaches, already existing models do not have to be changed and there is only one additional data model necessary, but no centralized knowledge configuration model or dependency model between parameters. Any information (e.g., functional or mathematical relations) how OUT-parameters are generated from the IN-parameters are provided by existing models.

Fig. 2 describes the basic methodology by the example of two models that are created with different (non-integrated) authoring systems. Once stakeholder 1 creates "model 1" using a specific tool, the model is transferred to the "model backbone" (e.g., a common model management system). In addition to this task, the essential parameters are transferred to a separate "parameter backbone" (e.g., by manual input or by using tailored interfaces). These essential parameters can be identified by the fact, that they are generated as (interim) results for a particular purpose and they are used several times as inputs for different models. If parameters appear only once within the development process (e.g., as an input for one downstream model within a seamless tool chain), the central storage may be not appropriate. For every parameter supplementing attributes (e.g., description, context information) need to be specified. Dependent on the type of every attribute, that user task can also be achieved either manually, (semi-)automatically or supported by using functionalities of the parameter backbone (e.g., DBMS "database management system") like (automatic) creation of unique IDs, access and change logs, or versioning. One important attribute that needs to be defined is which model generated the parameter. If stakeholder 2 starts creating "model 2" using another authoring system, the parameter management system provides access to already entered parameters. It is essential, that parameters can be found and identified due to unique attributes. There are supporting functionalities to retrieve the desired parameters, like using filtering by multiple criteria such as parameter name, description, creation date, creator, owner and/or many other context information. To establish the link between "model 2" and every required parameter, attributes are set. Every user of the parameter backbone now has the information, who (which model) generates and which models utilize the parameters. This information is helpful to estimate effects on other models in case of parameter changes. If "model 2" generates further essential parameters, they can be taken into account in the same way.

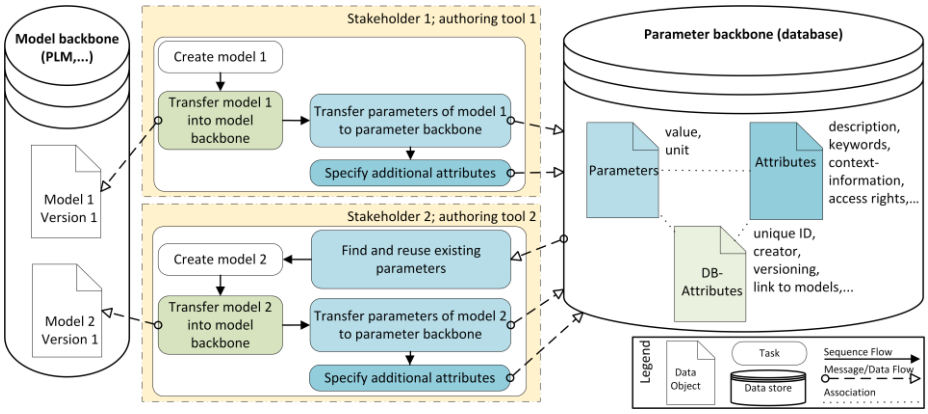

Fig. 2. Example how to use the introduced methodology in case of two models created by two stakeholders using different authoring tools.

To implement the parameter management approach (e.g., in an enterprise), a well-designed data model is a prerequisite for a transparent and performant implementation (e.g., within a DBMS). To develop the data model, at first an overview of the

parameters and parameter types to be managed must be created. Parameters (and parameter values) are usually spread and redundantly stored at different locations within physical files, that contain a particular selection (view) of parameters. Pragmatic reasons for the occurrence of such parameter sets can be: (i) Software tools need certain machine-readable file formats that can be imported/exported. (ii) Sets contain all parameters that belong to a model. (iii) Sets contain parameters structured by areas of responsibility (e.g., due to expert knowledge or organizational reasons). (iv) Sets contain all relevant parameters that relate to a subsystem (e.g., parameters that are relevant to production). If parameters appear multiple times within different files, all influences have to be considered in the database concept. At the implementation of a central parameter backbone, required views can be provided using adequate attributes to allow grouping or filtering. As an example, there are 10000 essential parameters necessary to describe a certain machine tool. Thereof a set of 50 parameters belong to an electric motor of a drivetrain (e.g., type, manufacturer, power, weight, nominal speed). Various stakeholders from mechanical, electrical and automation departments need different views at these motor parameters. While some of the parameters are created and solely used by one department, other parameters (e.g., like the nominal speed) play an important role for all stakeholders. At this point, the approach brings enormous advantages, since all departments have access to current parameter data (following specified rules) including history and transparency (regarding affected stakeholders and dependent models). To sort or filter parameters, additional attributes like responsible department, project name, assembly can be defined easily. In case of using a relational database for the implementation, there are significant advantages in terms of consistency, transparency and data integrity. Relations between separate tables containing sets of essential parameters (e.g., common motor parameters) avoid data duplication and make changes easier.

4 Example: Implementation of a parameter management system

The presented methodology is currently under test in industrial case studies in the field of mechatronic product development. The parameter management system is implemented using MySQL™ [16] as a DBMS and Visual Studio [17] VBA programming to build a graphical user interface for user login, manual parameter/attribute input or manipulation and to visualize database queries. In following example of the development of a mechanism, there are many models used by different stakeholders from different departments. A first stakeholder uses a model to perform a two dimensional multi criteria optimization of a mechanism using MATLAB® [13]. The output parameters of this model are characteristic dimensions (e.g., L_1=350mm … length of connection rod) that can be stored within the database using the ODBC API. In addition to every parameter, further corresponding attributes must be defined by using the following relations: “Model” (list and description of models used), “Units”, “Projects”, “Visualization” (e.g., a sketch to show the context information of a parameter or even a lightweight representation for example based on a JT visualization format [18]), “Person” (general user management) and “Status”

(e.g., to indicate a release status of a parameter). These relations are shown in Fig. 3 with an exemplary data model.

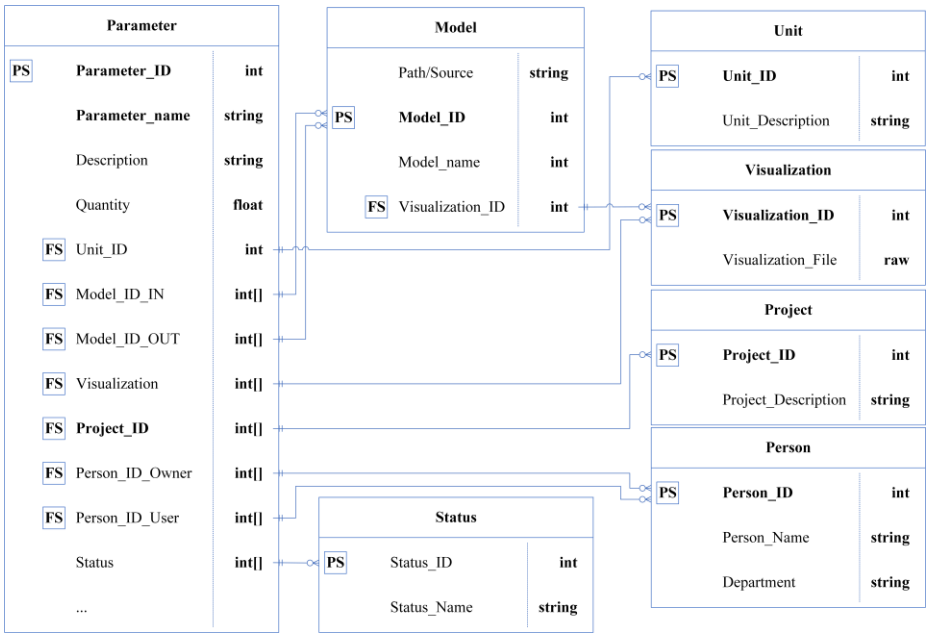

Fig. 3. Data model as a description of individual parameters and assigned attributes using relations (crow's foot database notation).

Based on the created database information, a second stakeholder is able to identify and reuse essential parameters to parameterize a CAD representation of the mechanism. This three-dimensional model is used to perform a collision check using the advanced capabilities of the Siemens NX™ CAD system. Input-parameters are loaded semi-automatically by the help of NX journaling functionalities to import expressions. Solutions of the collision check are also stored as parameters within the database, since they are important for other stakeholders. This means, every user of the parameter backbone has access to all actual parameter values, even without having access to the discipline-specific models (e.g., 3D-CAD model), and without the need to open the models within specific authoring systems.

Fig. 4 shows the prototype of the graphical user interface to create and manipulate individual parameters like the displayed parameter “L1”. In the lower area, a database viewer shows the query results depending on actual filter settings. The button “load model list” gives access to a separate list of models, that can be linked to the parameter. Write access (e.g., to parameter values) can be limited (e.g., to the “owner”/creator of the parameter) by generating access permissions, e.g., using the MySQL Access Privilege System and/or GUI programming.

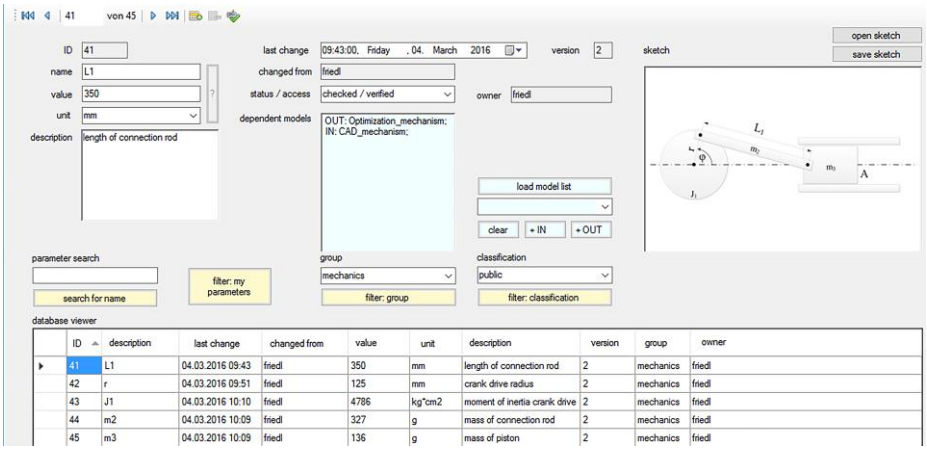

Fig. 4. Prototype of a graphical user interface to access the parameter backbone.

5 Summary and Conclusion

The presented approach essentially fulfills the initial requirements regarding parameter management, as described in chapter 1. The exemplary implementation is done using a DBMS. Interfaces between models (authoring tools) and the central database are set up using several APIs. Although, the attempts to keep it lightweight lead to deliberate limitations regarding traceability. The approach shows direct effects to subsequent models in case of parameter value changes, but effects over several models in series are not displayable, since all the information about parameter correlations remain within the models. Nevertheless, first implementations showed that the presented methodology brings transparency into multi-disciplinary development processes. Several users get access to parameter values and receive now additional information about relationships between parameters and models considering responsibilities of stakeholders, after a reasonably small effort in implementation. A difficulty that has to be addressed in future work is to keep the parameter values synchronized, also considering release status and access permissions. It is not always possible to design models in a way that they can handle input parameters or directly link to the parameter database. As a result, parameter values are stored within the models that are probably not up-to-date. Here a notification mechanism could be implemented, to inform the responsible stakeholder to update the model in case of relevant parameter changes.

Acknowledgments. This work has been supported by the Austrian COMET-K2 programme of the Linz Center of Mechatronics (LCM), and was funded by the Austrian federal government and the federal state of Upper Austria.

References

1. SyMMDe - System Models for Mechatronic Design, Multi-Firm-Project, Johannes Kepler University Linz, 2013-2016, http://symmde.jku.at (visited on 15/04/2016)
2. Friedl, M., Weingartner, L., Hehenberger, P., Scheidl, R.: Model Dependency Maps for transparent concurrent engineering processes, In: De Vin, L.J., Solis, J. (eds.): Proceedings of the 14th Mechatronics Forum International Conference. Mechatronics 2014, Karlstad, Sweden, pp. 614--621, (2014)
3. Hehenberger, P., Eynard, B., Bricogne, M., Le Duigou, J.: Meta-model of PLM for Design of Systems of Systems. In: Proceedings of the 12th IFIP International Conference on Product Lifecycle Management (PLM15), Doha (2015)
4. Abramovici, M., Aidi, Y.: A Knowledge-based Assistant for Real-time Planning and Execution of PSS Engineering Change Processes. In: 7th Industrial Product-Service Systems Conference - PSS, industry transformation for sustainability and business, Procedia CIRP, vol. 30, pp. 445--450, (2015). doi: 10.1016/j.procir.2015.03.026.
5. Badin, J., Chamoret, D., Gomes, S., Monticolo, D.: Knowledge Configuration Management for Product Design and Numerical Simulation. In: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Denmark (2011)
6. Riboulet, V., Marin, P., Noël, F., Delinchant, B., Gerbaud, L., Tools for Dynamic Sharing of Collaborative Design Information. In: Recent Advances in Integrated Design and Manufacturing in Mechanical Engineering. Dordrecht: Springer Netherlands, pp. 493--502, (2003). doi: 10.1007/978-94-017-0161-7_48
7. Vu-Thi, H., Marin, P., Noël, F., Bernard, A., Integrating Product Model and Whiteboard to Ease Collaborative Work in Global Product Development. In: Global Product Development: Proceedings of the 20th CIRP Design Conference, Nantes, France, Springer Berlin Heidelberg, pp. 217--226, (2011). doi: 10.1007/978-3-642-15973-2_21
8. Panchal, J.H., Fernández, M.G., Paredis, C.J.J., Allen, J.K., Mistree, F.: A Modular Decision-centric Approach for Reusable Design Processes. In: Concurrent Engineering, vol. 17, pp. 5--19, (2009). doi: 10.1177/1063293X09102251
9. Panchal, J.H., Fernández, M.G., Allen, J.K., Paredis, C.J.J., Mistree, F.: Facilitating Meta-Design via Separation of Problem, Product, and Process Information. In: ASME International Mechanical Engineering Congress and Exposition, Orlando, pp. 49--62 (2005)
10. Penciuc, D., Durupt, A., Belkadi, F., Eynard, B., Rowson, H.: Towards a PLM Interoperability for a Collaborative Design Support System. In: 8th International Conference on Digital Enterprise Technology – DET, Procedia CIRP 25, vol. 42, pp. 369--376, (2014). doi: 10.1016/j.procir.2014.10.051
11. SIEMENS LMS Imagine.Lab™ System Synthesis software, NX™ software and Teamcenter® software, http://www.plm.automation.siemens.com. (visited on 15/04/2016)
12. Enovia software, http://www.3ds.com, Dassault Systèmes (visited on 15/04/2016)
13. MathWorks® software, https://www.mathworks.com (visited on 15/04/2016)
14. Sakairi, T., Palachi, E., Cohen, C., Hatsutori, Y., Shimizu, J., Miyashita, H.: Model Based Control System Design Using SysML, Simulink, and Computer Algebra System. In: Journal of Control Science and Engineering, vol. 2013, Article ID 485380, (2013). doi: 10.1155/2013/485380
15. Bone, M., Cloutier, R.: The Current State of Model Based Systems Engineering: Survey Results from the OMG™ SysML Request for Information 2009. In: 8th Conference on Systems Engineering Research, Hoboken, (2010)
16. MySQL™ https://www.mysql.com (visited on 15/04/2016)
17. Microsoft Visual Studio software, http://www.visualstudio.com (visited on 15/04/2016)
18. Ding, L., Ball, A., Matthews, J., McMahon, C.A., Patel, M.: Product representation in lightweight formats for product lifecycle management (PLM). In: 4th International Conference on Digital Enterprise Technology, 2007, Bath (2007)