Title:
Multi-CAD Approach in Knowledge-Based Design

Authors:
Markus Salchner, markus.salchner@tugraz.at, Graz University of Technology
Severin Stadler, severin.stadler@tugraz.at, Graz University of Technology
Mario Hirz, mario.hirz@tugraz.at, Graz University of Technology
Johannes Mayr, johannes.mayr@magna.com, MAGNA STEYR Engineering AG & Co KG
Jonathan Ameye, jonathan.ameye@magna.com, MAGNA STEYR Engineering AG & Co KG

Keywords:
Knowledge-Based Design, Design Automation, Multi-CAD

DOI: 10.14733/cadconfP.2015.212-216

Introduction:
During the past decades, the development period of a new car has been reduced from five years and more to about two years. These considerable time savings can be attributed to a continuous optimization and improvement of development processes. In this context, virtual development methods play a prominent role. Product Lifecycle Management (PLM) solutions are used for data organization, distribution and storage within different computer-aided design and simulation applications. This comprises computer-aided design (CAD), styling (CAS), manufacturing (CAM), as well as computational toolkits, e.g. multi-body simulation (MBS), computational fluid dynamics (CFD) or finite element method simulation (FEM). Furthermore, lifecycle analyses (LCA) can also be implemented in modern PLM systems. The broad field of computational design and engineering disciplines within automotive development requires a strong interaction in view of efficient software applications. Especially the CAD-based design phase is characterized by a cooperation of manufacturer and supplier, which often use CAD software with different version or even different vendors. In this context, the paper focusses on the development and application of knowledge-based engineering (KBE) within multi-CAD environment. KBE deals with the storage and reuse of knowledge in product development processes [11]. A design-oriented specialization of KBE is knowledge-based design (KBD) [3], [5].

Usually, KBD methods are developed within and for specific CAD systems, respectively specific software configurations or releases [1], [6]. Especially in case of problem-oriented knowledge-based applications, the compatibility to different CAD software is restricted due to a high level of programmed customized features. Engineering and component supplier companies are faced with the problem that car manufacturers (OEM) work with different CAD software solutions as can be seen in Figure 1. In this example, an automotive supplier has to support different CAD systems, including numerous OEM-related methodical requirements. This results in a need of high sophisticated levels in different CAD-systems including a complex application of knowledge-based design tools. A multi-CAD strategy could support the handling and organization of different CAD-related project environments, especially in view of knowledge-based and automated design applications, as well as data management.

Main Idea:
In contrast to the general understanding of a multi-CAD environment, the present strategy represents an unrestricted method. Functionalities in commercial available CAD systems are restricted to the provision of neutral geometry data interfaces for the exchange of CAD models, e.g. STEP, IGES, STL.

These converters allow an exchange of different CAD formats, but there is no possibility to handle intelligent knowledge-based software modules, e.g. templates for automated geometry creation. The presented approach provides an efficient development and implementation of multi-CAD capable knowledge-based design methods [2], [9], [10].

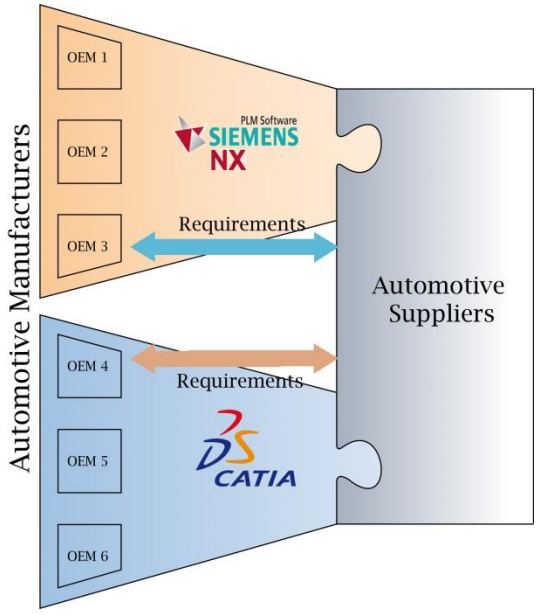

Fig. 1: Automotive supplier challenge: OEM with different CAD environments.

Figure 2 illustrates the environment of the present approach including the internal main modules and their docked satellites. The abstract methods, which cover universal functions, are stored inside the framework. This includes “CAD base”, “Utilities”, “Infrastructure” and “Settings” classes. These base classes are managed by the KBD-Expert and define fundamental functionalities of the docked satellites. The development and implementation of these objects are not explained in detail in this paper, except the “CAD base” class. This superordinate functional definition allows docking of different applications in an efficient way and grants the consequent enhancement of the framework. The KBD-Engineer implements the framework in the development process of a new KBD-Tool. The developer can access – on require – all implemented methods via the novel framework. The final KBD-Tool has to be analyzed by the KBD-Expert regarding their implemented functionalities. In the next step, the KBD-Expert performs an abstraction of the functionalities and defines the adequate implementation of new base functions. Thus, the base functionality rises with each single application and expands the predefined functions for future projects.

The CAD related assemblies are illustrated at the bottom of Figure 2, by Siemens’ NX [10], Dassault’s CATIA V5 [2] and any additional CAD system – like PTC Creo Elements [9]. Furthermore, third party applications like Excel, Acrobat or Matlab are also supported by the interface. Different PDM systems, databases and finally single specialized KBD tools are optional satellites, which can be implemented in the multi-CAD-Framework. The predefined overall structure grants the integrity and the intuitive working principle for all KBD-Engineers. Furthermore, the environment includes a defined strategy for serviceability, reuse, as well as protection and licensing aspects.

The database interface gives the developer the ability to use different database functionalities. This includes the ease creation of user defined local or server databases and allows quick storage and reuse of application related information. Furthermore, database-applications provide useful functionalities for listing, sorting, or searching of data and enable a dynamic application layout.

The overall implementation expands the function scope enormously and enables an efficient interaction between CAD and third party tools. Exemplary, interdisciplinary KBD tools are geometry-based analysis and optimization routines, e.g. specific suspension optimization. For example, this enables a transfer of main suspension points, which are defined and stored in the CAD system, to a Matlab optimization routine. The routine performs an optimization with regard to occurring forces, the maximum possible steering angle or dive angle. The results are displayed in the CAD system and the engineer can adapt the design model. This application is exemplary illustrated as KBD-Tool 1, whereby the KBD-Tool 2 represents a separate exemplary application, shown in Figure 2. Each implemented application expands the framework and enables a continuous and efficient enhancement.

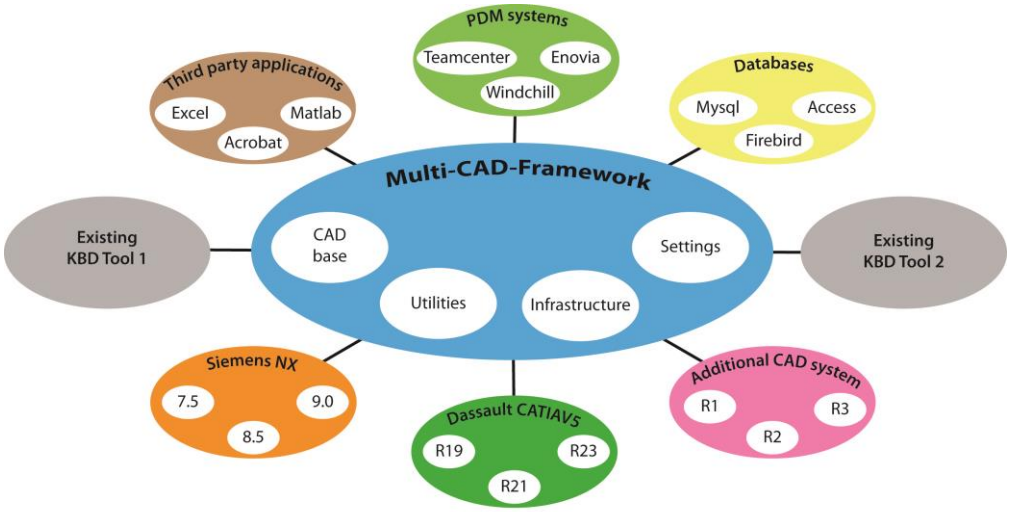

Fig. 2: Multi-CAD-Framework.

The “CAD base” class initializes all CAD related developed functions and methods. This library does not implement the functionality itself, but it defines them. Each single implemented CAD system is a derived class and is forced to implement the defined functions. Relatively simple CAD basic functions, like the creation of a point, a line or a feature, are not implemented in this level, because these CAD related functions are supported by the API and it is not useful to reproduce all of them. In contrast, it includes a collection of commonly used and know-how based functions. On basis of an object oriented programming technique, it is possible to force the developer to implement the designed function for all required CAD solutions. One example of application is a function for capturing of images. The code for the image creation is not challenging, but to adjust a whole environment to the defined settings can be very complex. The implemented function performs the following actions in addition to the passed parameters: Perspective, position, zoom, visibility of constraints, annotations, treeview, components, change the color of parts, surfaces or even the entire assembly.

The functional scope of the automation routines is only limited by the provided CAD-API, whereby this functionality is often coupled with the used programming language and available license packages. The common denominator of different CAD systems regarding functionality, costs and development within the presented approach is stated by the .NET framework [8]. It is a software framework developed by Microsoft [7] since 2002, which supports different coding languages like Visual Basic (VB), C# or J#. Due to the fact, that many CAD solutions deliver integrated VB programming editors and simultaneously provide VB.NET, the multi-CAD strategy is developed based on this programming language. The main disadvantage, compared to programming languages like C, is a lack of code protection. Nevertheless, there are some options to protect the code in an efficient and sufficient way. Considering that today’s most KBD tools in automotive industry are specific in-house applications, the protection of knowledge plays an important role [2], [9], [10].

Besides the mentioned differences, the initial linkage of the presented automation application to the desired CAD system is also defined by the API. CATIA for example, is based on the Component Object Model (COM) of Microsoft, and therefore a connection to the actual active CATIA window is reached out due to the registered application name of CATIA. Thus, different CATIA releases cannot be differentiated and therefore multiple window applications are not allowed. The API of NX on the other hand is not based on the COM strategy. In this case the connection is performed due to the Transmission Control Protocol/Internet Protocol (TCP/IP) by use of a server and a client application, whereby both can be applied on one single computer [2], [8], [10].

The explained framework can be implemented and used as references by each KBD-Engineer in the development environment of Visual Studio. The depicted satellites in Figure 2 are standalone libraries, whereby an individual implementation is enabled. Some applications do not need a database - or the CATIA interface, and therefore an implementation of these libraries is not effective. The current framework comprises the following main libraries:

- Multi-CAD-Framework
- Dassault CATIAV5
- Siemens NX
- Third party application – e.g. Excel
- Firebird databases [4]

Additional functionalities with numerous overloading methods are stored in a separate snippet library. They provide a large amount of different input parameters regarding to their use. An exemplary CAD related function is the "CreatePoint" method. The design of a point can be done in various ways - by coordinates, on a line, a surface, a body or as a center point of a circle. Twenty overloading methods in NX are available, whereby each of them uses different NX related objects, which in turn are overloaded. Other exemplary methods are folder selection, file selection – or save methods, which strongly depend on the current requirements. The snippets are implemented in the graphical interface of Visual Studio and can be easily accessed. The snippets are based on XML and copy the stored source code elements into the current project. The KBD-Engineer can customize the scripting lines to the current requirements. The developed snippet library uses the same structure as the framework library and grants an intuitive use.

Conclusions:

The presented approach provides a platform for the efficient development of KBD applications for multi-CAD environments. This includes a multi-CAD interface and different software modules that manage inconsistent CAD APIs, capture knowledge, provide reusable functions and libraries, and have an enhanced service and error handling. In contrast to common approaches, where one single KBD tool has to be released separately for each single CAD-system and release, this novel strategy allows a standalone-releasing of different KBD tools. This simplifies company-internal roll out procedure, update management and administration. Of course this strategy is part of the framework and uses the mentioned Firebird database as application storage, user management and administration. Furthermore, it plays an important role regarding the protection of the application and consequently of the knowledge. On the other hand, the change or update management in view of the KBD-Engineer is therefore optimized, because of a centralized project management. In this way, the KBD-User is not forced to execute different application files and the standardized consistent GUI of all KBD tools minimizes the training period significantly.

References:

[1] Chapman, C.B.; Pinfold, M.: The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure, Advances in Engineering Software, 32, 2001, 903–912, http://dx.doi.org/10.1016/S0965-9978(01)00041-2.

[2] Dassault Systemes, http://www.3ds.com, access date: 2015-01-27.

[3] Eigner, M.; Stelzer, R.: Product Lifecycle Management: Ein Leitfaden für Product Development und Life Cycle Management, Springer, 2009, ISBN: 9783540443735, http://dx.doi.org/10.1007/b93672.

[4] Firebird, http://www.firebirdsql.org/, access date: 2015-04-13.

[5] Hirz M.; Dietrich W.; Gfrerrer A.; Lang J.: Integrated computer-aided design in automotive development: development processes, geometric fundamentals, methods of CAD, knowledge-based engineering data management; Springer, 2013, ISBN:9783642119392, http://dx.doi.org/10.1007/978-3-642-11940-8..
[6] Ma, Q.C.; Liu, X.W.: Review of Knowledge Based Engineering with PLM, Applied Mechanics and Materials, 10-12, 2007, 127-13, http://dx.doi.org/10.4028/www.scientific.net/AMM.10-12.127.
[7] Microsoft, http://microsoft.com, access date: 2015-01-27.
[8] Microsoft Developer Network, http://msdn.microsoft.com/de-at/dn308572.aspx.
[9] PTC, http://www.ptc.com, access date: 2015-01-27.
[10] Siemens PLM, http://www.plm.automation.siemens.com, access date: 2015-01-27
[11] Stokes, M.: Managing Engineering Knowledge – MOKA: Methodology for Knowledge Based Engineering Applications, Professional Engineering Publishing Limited, ASME Press, 2001.