INTERNATIONAL DESIGN CONFERENCE – DESIGN 2020
https://doi.org/10.1017/dsd.2020.313

THREE WAYS OF INTEGRATING COMPUTER-AIDED DESIGN AND KNOWLEDGE-BASED ENGINEERING

P. C. Gembarski ✉

Leibniz Universität Hannover, Germany
✉ gembarski@ipeg.uni-hannover.de

Abstract

Knowledge-based engineering (KBE) systems allow an easy adaption of designed artefacts to new functional or design requirements and automating routine design tasks. In the following article the author wants to focus on the three main concepts of linking CAD and KBE and answer the research questions (1) in which way is integration, embedding and coupling of KBE to a standard CAD system like Autodesk Inventor available and (2) how can the single approaches be compared in terms of modelling effort, user competences and system performance.

Keywords: knowledge-based engineering (KBE), design automation, problem solving, design knowledge

1. Introduction

Computer-aided design (CAD) and engineering (CAE) have increased the competitiveness and innovation ability of mechanical engineering companies for more than 20 years. Parametric design systems in particular offer high potential for adaptive and variant design tasks (Vajna et al., 2009). The ability to define constraints between parameters in a CAD system enables the implementation of explicit knowledge in digital prototypes. Thus the designer must also define the control and configuration concept for his artefacts in addition to the shape (Shah, 2009). Knowledge-based design (KBD) and knowledge-based engineering (KBE) go a step further to make it even easier to adapt a designed artefact to new functional or design requirements and automating routine design tasks (Hirz et al., 2013).

Although rather not new, KBE and KBD are still only used today in single aviation or automotive engineering applications or niche design activities, e.g. fixture design (La Rocca, 2012; Boyle and Brown, 2011). As stated in the past, formalizing and explicit modelling of knowledge is still a challenge and effort, especially when the created artefacts should be of reusable manner (Stokes, 2001). In contrast, describing a solution space within a KBE system is beneficial for the derivation of product variants and the reduction of uncertainties when restrictions from production or combinatorial limitations of product components in the sense of product logic are modelled as well (Gembarski and Lachmayer, 2017; Li et al., 2018). Looking from the tool perspective, today's commercial CAD systems, like e.g. Autodesk Inventor or CATIA, offer KBE modelling concepts like in-system-programming of knowledge artefacts like if-then design rules, spreadsheet integration or knowledge workbenches as extension for the CAD system (Skarka, 2007; Gembarski et al., 2017).

Literature and guidelines, like VDI (2017), report about different possibilities of linking a KBE to a CAD system but there is a lack of modelling principles and comparative best practices for practical

implementation. In the following article, the author wants to focus on the three main concepts of linking CAD and KBE and answer the research questions (1) in which way is integration, embedding and coupling of KBE to a standard CAD system available and (2) how can the single approaches be compared in terms of modelling effort, user competences and system performance? The remainder is structured as follows: In section 2, the theoretical background about KBE systems and their problem-solving behaviour is presented. Afterwards in section 3, a locating jig is introduced as application example and modelled according to the three different linking possibilities. Section 4 then briefly discusses the established KBE systems before section 5 presents a short conclusion.

2. Theoretical background

2.1. Knowledge-based engineering and design systems

Knowledge-based engineering (KBE) and design systems are a subgroup of knowledge-based systems, which have to be understood as a collective term for computer-aided problem-solving tools (Milton, 2008). The problem-solving behavior is generally based on that of a human expert, therefore the term expert systems developed as a synonym for knowledge-based systems of all kinds especially in the 1980s and 1990s. Examples include assistance or diagnostic systems in medicine, speech recognition tools or automatic classification systems (Clancy, 1983; Hayes-Roth, 1995; Hopgood, 2012).
La Rocca (2012) understands KBE systems as software tools which capture product and process development knowledge and apply it to new situations. Chapman and Pinfold (2001) clarify this and consider these tools as an evolutionary step in computer-aided engineering which results from the combination of object-oriented programming, artificial intelligence and CAD. In the broadest sense, the systems generate product descriptions based on predefined functions, components, relationships, restrictions and preference criteria (Schreiber, 2008). Their use supports the automation of routine tasks in product development and variant design as well as the draft and dimensioning of technical systems or product components (Lutz, 2012). KBE systems are built from the basic components knowledge base, inference engine and interfaces (Figure 1). The knowledge base is the storage for the expert knowledge, the inference engine applies the knowledge base to the given problem by using inference and task knowledge (Milton, 2008). Optionally, further components such as e.g. dialogues for knowledge acquisition or an explanation component, which presents the individual conclusions made by the system to the user in an understandable way, can be included (Hopgood, 2012). In addition, there are interfaces for user interaction and to other hardware, software or data storage systems. One of these interfaces usually integrates CAD systems or geometry models (La Rocca, 2012).

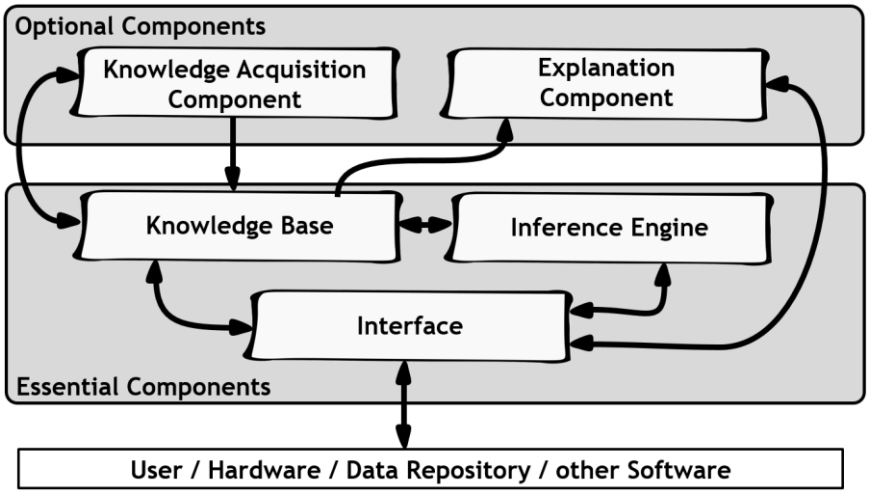

Figure 1. Main components of a knowledge-based system (according to Hopgood, 2012)

The German VDI guideline 5610-2 distinguishes an integrated and a coupled approach (VDI, 2017). In the first case, the knowledge base is fully integrated in the CAD system, the latter means that the

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press

KBE system remotely controls the CAD system, so both systems are separate. So, the knowledge base can be edited independently of the CAD model and is usually available in a separate file (Lutz, 2012). Individual CAD system manufacturers also follow a third path (Figure 2). With the embedded approach, parts of the knowledge base are modelled in external systems (e.g. in a spreadsheet program), but the corresponding file is not saved separately but embedded in the CAD model and can only be called or edited here (Gembarski et al., 2017). This approach can also be assigned to knowledge-based CAD, since the CAD model indirectly represents the storage location for the illustrated knowledge. Individual authors take up this differentiation and distinguish between KBE- and knowledge-based design systems. While the first focus more on conception, design as well as layout and represent separate software systems, the latter are synonymous with knowledge-based CAD (Hirz et al., 2013).

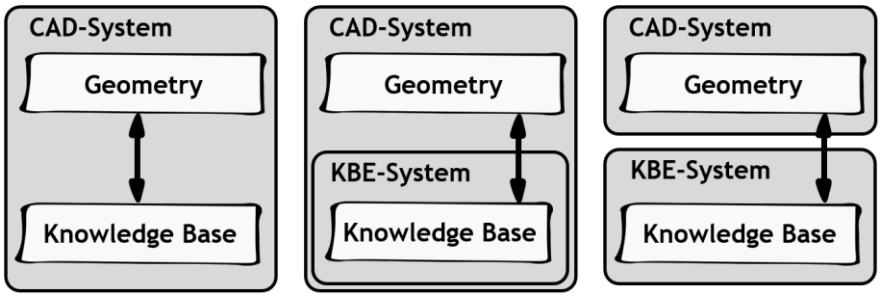

Figure 2. Integration approaches for CAD and KBE - left: integrated; centre: embedded; right: coupled (modified according to VDI, 2017)

2.2. Problem-solving in knowledge-based engineering systems

For the implementation and later use of a KBE system, it must first be determined which tasks the system is to perform and how the programmed knowledge should be applied in order to solve a problem (VDI, 2017). Of central importance is the decomposition of a task into individual elementary problem-solving steps (Koller, 1991). With regard to design engineering, three basic types of synthesis operations can be distinguished (Cunis et al., 1991; Schreiber, 2008; Gembarski et al., 2016).

Synthetic design involves designing a system that meets specified requirements (Schreiber, 2008). Koller (1991) explicitly names the creation of a functional structure from given requirements as an example. In relation to a product, however, the design of new, previously non-existent components or products also belongs to this task. Within their work on KADS and CommonKADS, which are both development methods for knowledge-based systems in general, Schreiber (2000) also defines structures and pseudocode for synthetic design (Figure 3). As input, the user first formulates requirements which are operationalized by the KBE system and translated into hard and soft ones. The first can be seen as fixed requirements and enable the KBE system to filter possible system designs that have been generated on the basis of knowledge about system creation. As knowledge elements for this step, the relationships between system design and requirements must be modelled. The system designs which were validated in this way are then evaluated and ranked on the basis of the soft requirements, which are to be understood as target requirements or wishes.

Configuration is a subgroup of synthetic design, where the system is composed of fully predefined elements. These are coupled via defined interfaces so that no constructive adjustments are necessary (Milton, 2008). Different problem-solving mechanisms exist for configuration, one of them is purpose-and-revise. The KBE system uses information about templates and the knowledge for system creation analogous to the synthetic design to propose an initial configuration which is tested against the operationalized requirements in the following step. Therefore, a so called Truth-Maintenance-System is used which is a component of the problem-solving mechanism and responsible for detecting violations of requirements (Chandrasekaran, 1990). Based on a predefined reaction pool, the initial configuration can be modified, then checked again by the Truth-Maintenance-System and either further modified or output (see Figure 4). Since there may be several ways of correcting individual errors, the reaction pool is usually weighted to determine which solutions are to be used as a priority. If such a solution does not lead to satisfactory results, the Truth-Maintenance-System will return to this point and test the next corrective

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press

action out of the reaction pool (Marcus and McDermott, 1989). From an information science point of view, configuration tasks can be written and solved as constraint satisfaction problem (Barták et al., 2010).

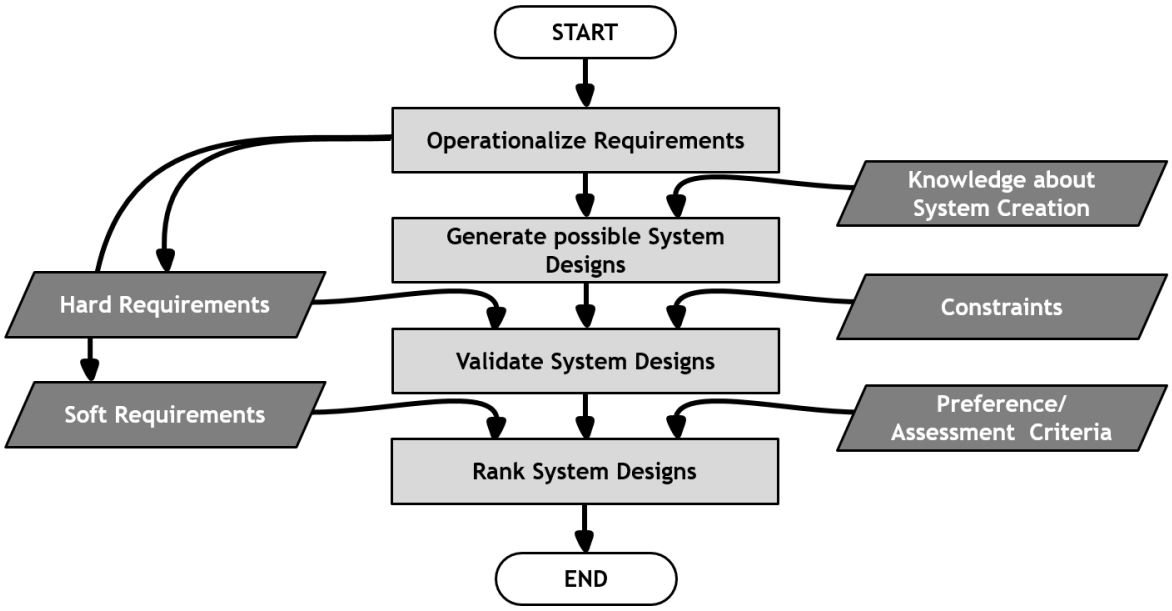

Figure 3. Problem-solving for synthetic design (according to Schreiber, 2000)

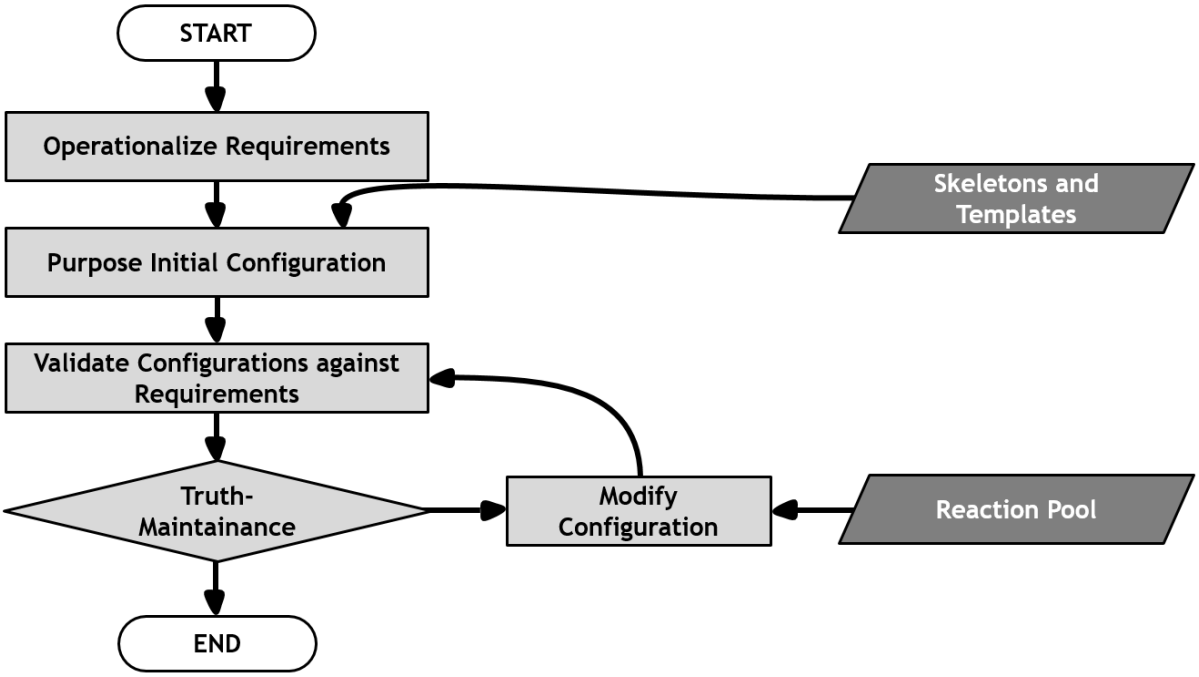

Figure 4. Problem-solving for configuration (according to Chandrasekaran, 1990)

Although the building blocks used in configuration themselves do not have any degree of freedom, a very large solution space can be created with an appropriate design. The decisive factor here is the number of combination interfaces and rules. This was shown by Durhuus and Eilers (2005) using the following experiment: Six LEGO bricks of size 2 x 4 are given. They should be connected in such a way that no bricks are free-standing, the height of the structure or its base area are not restricted. However, the bricks must be arranged in a rectangular grid, configurations in which a brick is tied at an angle to a corner are not permitted. This task was then passed on to a configuration system, which had to determine the number of valid configurations by Generate-and-Test. 915,103,765 valid variants were found.

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press

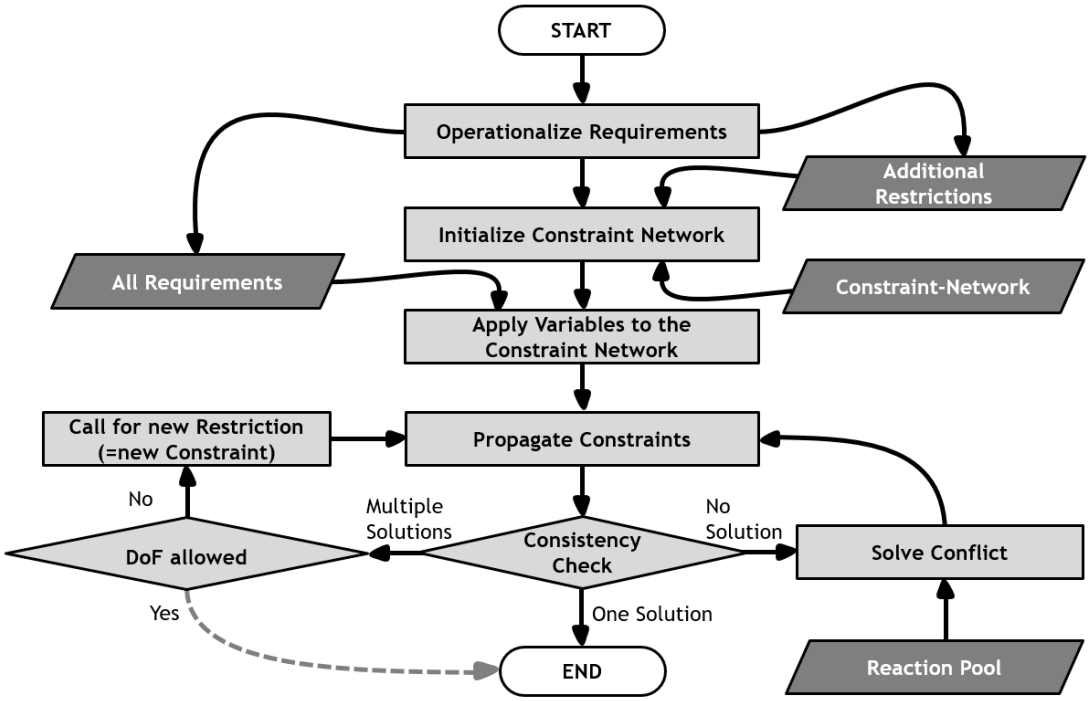

Figure 5. Problem-solving for parametrization (Gembarski et al., 2016)

The third synthesis operation that can be performed by KBE systems is to solve a constraint network (Cunis et al., 1991) and is named *parametrization*. In practice, in a given system described by variables and dependencies, all degrees of freedom (e.g. with regard to dimensions or activation of individual design elements or components) are eliminated step by step by setting parameter values (Gembarski et al., 2016). The requirements formulated by the user are first operationalized into concrete values, which are created and propagated after the initialization of the constraint network. A value can not only be instantiated as a variable, but also as a restriction. Three states may occur after the consistency check: If the system determines exactly one solution, the synthesis operation is finished. If no solution is found, an attempt can be made to solve existing conflicts in the constraint network with an existing reaction pool. If the consistency check results in multiple solutions and this is permitted, the synthesis operation ends as well. If multiple solutions are not allowed, a new restriction is requested by the user, which extends the constraint network then. This process is repeated until one or no solution is available.

3. Modelling example: Locating jig

As modelling example, a task from computer aided fixture design was chosen since design knowledge in this domain is well documented and formalized (Alarcón et al., 2010; Boyle and Brown, 2011; Hesse et al., 2012). The task is composed of configuration and parametrization problem solving activities for designing a locating jig (Figure 6). The jig follows the locating principle *2-pin + support* where parts are located via two holes and a face (Rong and Zhu, 1999). It is assembled from standard parts of a modular design kit that consists of multiple fixed components (configuration task) as well as parts that can be adjusted e.g. to a needed height with good accuracy (parametrization task).

As input data for the fixture configurator, the user first selects five reference faces. The plane faces of the primary and secondary datum are not relevant for the determination of the component; they serve either to support the component or to determine the height position of the locating machine elements. This results in the following parameters that need to be processed:

- Diameter of primary and secondary datum face
- Height difference between primary and secondary datum
- Height difference between secondary and tertiary datum

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press

- X-Y-position of secondary datum related to primary datum
- X-Y-Position des of tertiary datum related to primary datum

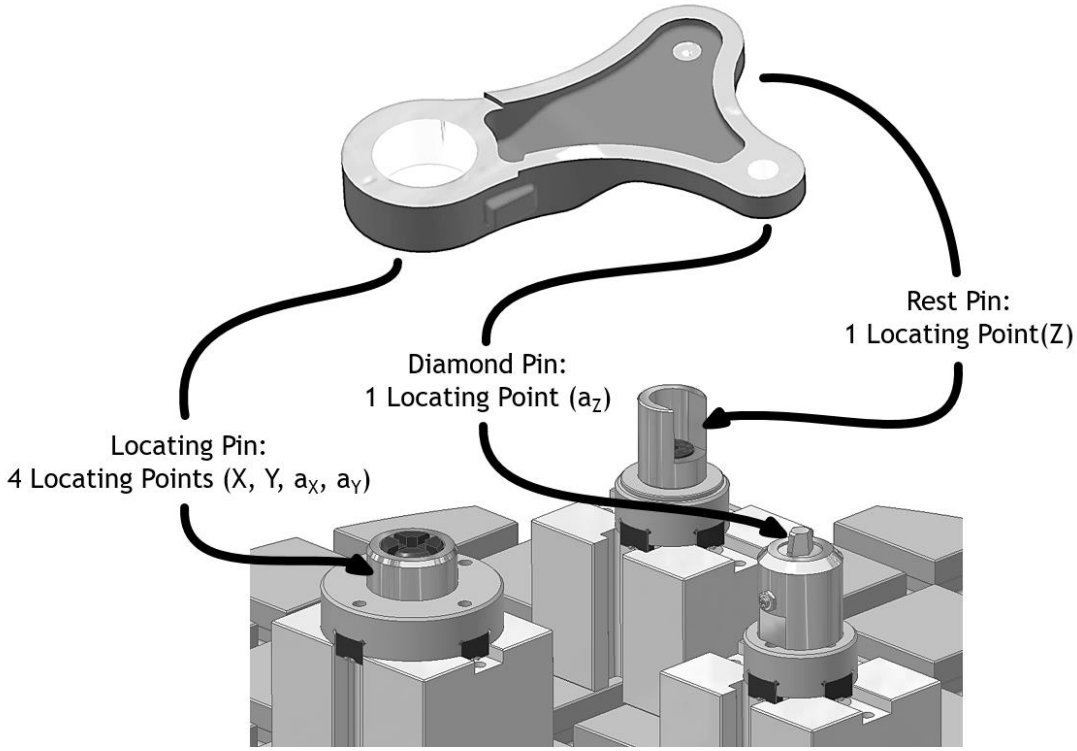

Figure 6. Main components and locating principle of the modelled jig

As implementation environment, Autodesk Inventor was chosen since it has the necessary KBE-modelling techniques, like programming of iLogic rules, an open API and spreadsheet integration, available (Gembarski et al., 2017) that will be briefly presented in the following parts. Since the design operations within the CAD system comprise modifying the suppression state of components in the assembly and changing parameter values of dimensions and constraints, the jig uses basic assembly modelling features.

3.1. Integrated CAD-KBE-system based on iLogic

For the integrated CAD-KBE modelling approach, the iLogic programming language was used which is similar to script languages. Common structures like if-then-else statements, select-case decision trees and loops are available. The code can be structured in sub functions and sub procedures, variables may be used globally and locally. Inventor offers a snipped library for almost every modelling context.

```
'Rule 1106: choose diamond pin, calculate distances and configure support elements

IF secondary datum = 20  AND  height1-2 > 0 THEN
                iP:PLB = iS:P + 20
                iP:PLT = iS:P + 50
                iP:BB = iS:B - iS:P - iP:PLB
                Component.IsActive(„11306_21020")= True
                Component.IsActive(„11305_20XXX")= True
                iPart.ChangeRow(" 11305_20XXX :1", „ 11305_20202")
END IF
```

Figure 7. iLogic rule (excerpt)

Based upon the input parameters, iLogic rules determine the necessary machine elements for building the jig (Figure 7) and sets either the suppression state or choses the right member of the according iPart (parametric part families). Afterwards, the machine elements are placed on the base plate. In

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press

order to accelerate the model update, the force-rebuild-command is disabled while the rules are fired. Normally, each parameter update would result in a propagation of all constraints in the CAD model.

3.2. Coupled CAD-KBE-system based on VB.net

The design problem of the jig can the transformed into a configuration problem from information science. Therefore, the task must be modelled as constraint satisfaction problem (Figure 8 shows part of the according constraint network). As domains the components of the jig were formulated as value tuples from the respective order number and the relevant dimensions for the assembly. The task of the reasoning algorithm is now to compensate for the height differences of the three datum planes. For this purpose, the input variables such as diameters and heights are first assigned to the base of the datum. Constraint C2 first restricts the number of possible locating pins that include their respective heights in the calculation of the height differences. The constraint C3 also connects the retaining rings, which are evaluated with regard to their bore diameter and also included in the height difference calculation. C1 then establishes the relationship between the two height differences and, if necessary, initiates the addition of the spacers and height cylinders.

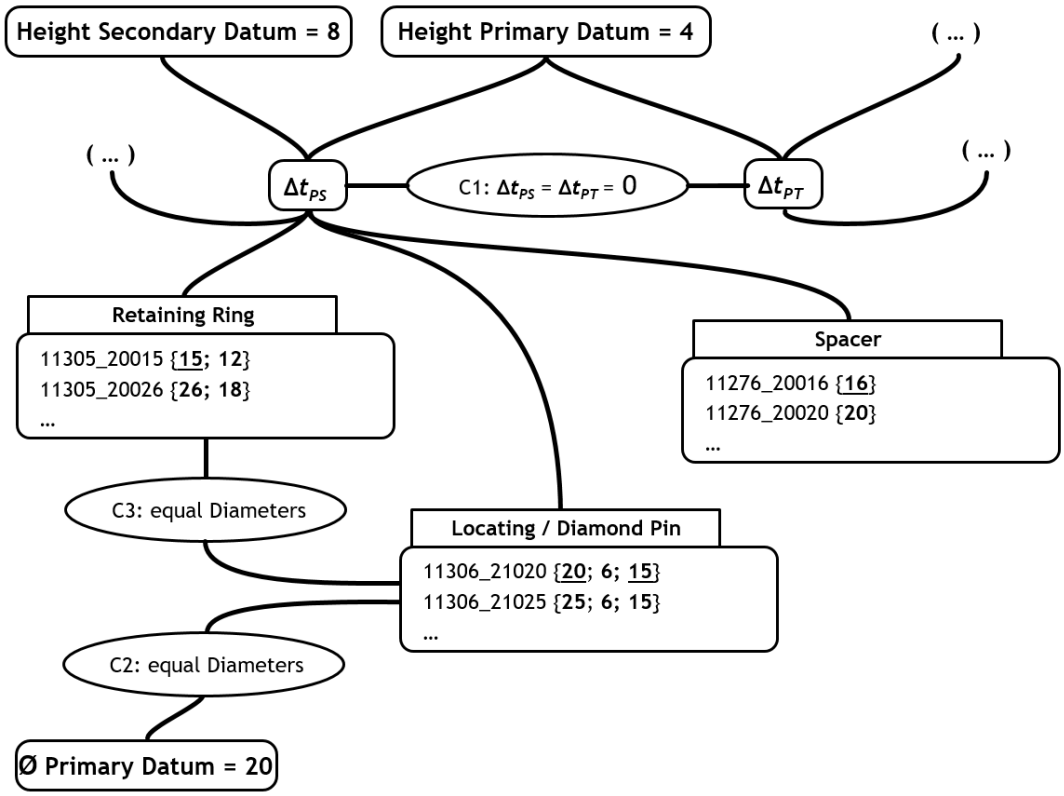

Figure 8. Constraint network and domain model (excerpt)

The implementation took place in VB.Net which has a good integration into the Inventor API. The higher object and event orientation enabled a more flexible program flow compared to the iLogic implementation. The constraints of a domain can be propagated directly with the OnChange event trigger. After the constraint satisfaction problem is solved, the KBE system remote controls Inventor and passes all parameters as well as the component suppression states to the CAD system where the jig is assembled.

3.3. Embedded CAD-KBE-system based on spreadsheet integration

For the embedded linking approach, the spreadsheet integration within Inventor was used. Therefore, an Excel spreadsheet was first created that consists of different worksheets. The first one contains the parameter setup for the CAD assembly and is organized strictly by the parameter definition of Inventor which includes information about name, value, unit and comment in the corresponding formatting. Other

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press

worksheets contain information about the available machine elements which are linked to the parameter setup by different mathematical and logical constraints. An interesting point here is the availability of the VLOOKUP-function in Excel since it can call for parameters of the single machine elements based upon an input parameter. In this way, the choice of the locating and diamond pins was realized. Additionally, if more or specific functionality is needed, the spreadsheet can be enriched with VBA macros, e.g. for user communication in form of input dialogues or message boxes or for extended plausibility checks.

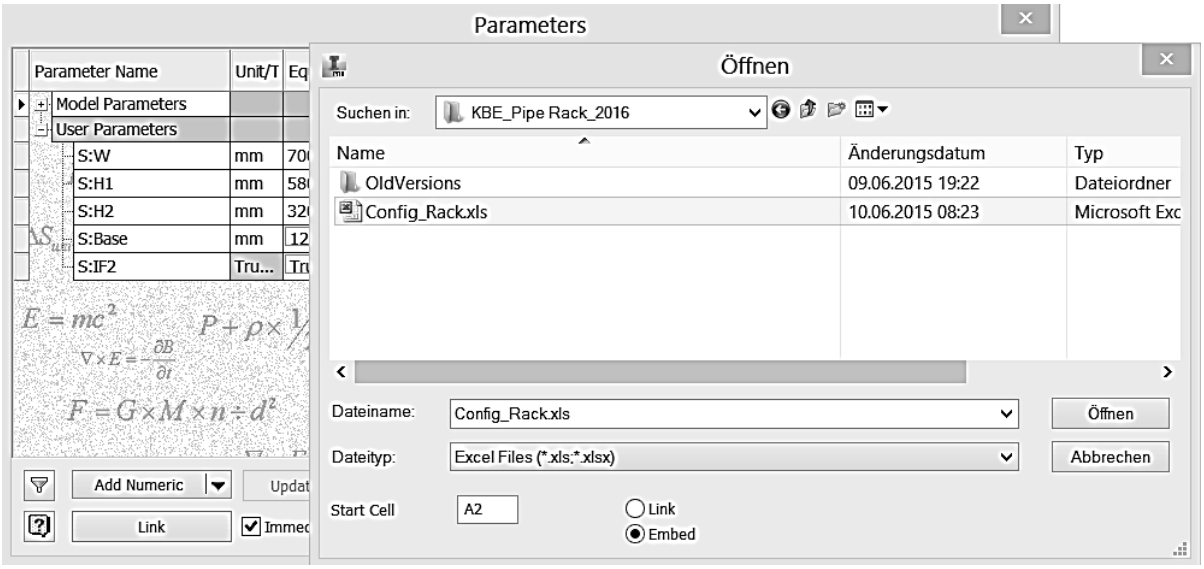

Figure 9. Linking dialog excel to inventor

The processing of the Excel spreadsheet can be done in basically three ways: (1) the table can be linked to an Inventor model. This option then necessitates the user to hardcode the path for the table. If the file is moved or stored on a central network server, this link can easily break. Recovery is possible only with the Inventor API. (2) Embedding the table moves the file directly into the Inventor CAD model as central information hub for parameter control. (3) The table can be extended to an iAssembly table which means that not only the parameter control is organized in the table but also the suppression state of component occurrences and iPart member information. Especially in the last variant, the functionality equals the one from the other approaches described above. In case of the other variants, additional iLogic rules have to be implemented in order to change standard part members and execute the component suppression states.

4. Discussion

At a first glance, the three realized CAD-KBE implementations seem very different to each other although they have comparable features and fulfil completely the same tasks in configuring and parametrizing the locating jig. Nonetheless, the systems can be compared regarding system performance, modelling effort and the necessary user competences for their creation.

System performance: The rebuild time of the jig assembly was measured after different configuration tasks. Interestingly, it differed only to a minor degree between the three implementations as long as the force rebuild-command in Inventor is suppressed so that the model does not regenerate with each parameter change. In the iLogic implementation this had to be actively coded, the other two variants, by nature, transfer calculated parameters as block to Inventor. Summed up, no performance advantages of one against the other approaches could be found.

Modelling effort: The written code of the iLogic implementation was about 8000, expressed in 329 rules characters (for an example, see excerpt in Figure 7). Regarding the coupled approach, constraint network and solver were programmed with about 4000 characters of code in VBA including all domain variable arrays. For the spreadsheet implementation, the effort is even less, since knowledge and constraints are coded mostly within the tables where Excel supports the linking and creation of series. Nevertheless, additional effort is needed when linking the CAD models either by rules or the iAssembly.

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press

Focussing on maintainability in this context, the coupled and the embedded approach are easy to extend since domain variables and tables can be modified independently from all constraints and normally no further check for consistency is necessary. The iLogic implementation differs from this: If a new rule is introduced into the rule base, the consistency has to be checked manually so that no unwanted interdependencies occur.

User competences: When an engineer needs to formulate such systems, the competence level rises from excel spreadsheet over iLogic to the constraint satisfaction problem. In the first case, the organization of the spreadsheet workbooks has to be planned and features like the VLOOKUP must be known. But generally, operating Excel is a task that is quite common in design engineering, so that the competence level is, compared to the other approaches, low. Regarding the iLogic implementation, programming competences must be at hand, since the rule base is processed as procedure. For large rule bases it is advisable to implement a meta-structure in the program in order to avoid actualization loops and establish a more flexible program flow. Modelling the system as coupled CAD-KBE system necessitates a different view on the design problem since it has to be translated into a constraint satisfaction problem. This calls for thinking and modelling of domains instead of rules and parameters and is a different level of abstraction. Generally, the basic implementation of the KBE part can also be a rule-based system and thus strongly correspond to the iLogic variant.

It has to be mentioned that there are also mixed forms of linking the functionalities. E.g., an embedded Excel spreadsheet can be applied for calculating only parameters and pass them to Inventor, where iLogic is used only for the execution of suppression states and the addressing of iParts.

5. Conclusion

In the present article, different methods and functionalities for problem solving and linking KBE to CAD systems have been introduced. In order to answer the research question if and in which way the proposed approaches of integration, embedding and coupling of KBE to a standard CAD system like Autodesk Inventor is possible, the case study of a locating jig indicated the basic availability. The comparative study showed that none of the concepts is clearly superior to the others. Both configuration and parametrization tasks which were needed to model the jig are addressable regarding functionality. Nonetheless, at least basic competences in software engineering are advisable but work-arounds like the use of spreadsheets for parameter control and calculation are available and show good results.

A major disadvantage from the modelling approaches is the knowledge engineering that has to be done beforehand. All modelled systems belong to the field of weak artificial intelligence which means that domain and control knowledge have to be modelled explicitly. As a result, knowledge needs to be formalized in rules, formulae or other computer readable representations. Moreover, these knowledge artefacts must be linked to the CAD models explicitly so that discreet parameter IDs or other model elements are influenced. This hinders the re-use of such knowledge artefacts since they have to be re-parameterized in a new modelling context. So, although the tools have changed over the last thirty years, the major challenge of KBE is still the same.

Targeting on reusability, the use of ontologies as intermediator between the knowledge model and the CAD model could be of interest. While an ontology-based application takes the role of the KBE-system, an interface to the CAD files needs to couple parameters and other entities necessary for control to the ontology itself. This would widely correspond to the coupled approach.

As another point, the use of strong artificial intelligence to model knowledge implicitly is highly interesting. Beside the use of machine learning to identify patterns in design and features, the use of e.g. in-CAD multi-agent systems for the analysis of product characteristics according to design guidelines is a possible direction.

References

Alarcón, J.R., García, J. and Idoipe, A.V. (2010), "Fixture knowledge model development and implementation based on a functional design approach", *Robotics and Computer-Integrated Manufacturing*, Vol. 26 No. 1, pp. 56-66.

Barták, R., Salido, M.A. and Rossi, F. (2010), "Constraint satisfaction techniques in planning and scheduling", *Journal of Intelligent Manufacturing*, Vol. 21 No. 1, pp. 5-15.

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press

Boyle, Y. and Brown, D.C. (2011), "A review and analysis of current computer-aided fixture design approaches", *Robotics and Computer-Integrated Manufacturing*, Vol. 27 No. 1, pp. 1-12.
Chandrasekaran, B. (1990), "Design problem solving: A task analysis", *AI magazine*, Vol. 11 No. 4, pp. 59-71.
Chapman, C.B. and Pinfold, M. (2001), "The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure", *Advances in Engineering Software*, Vol. 32 No. 12, pp. 903-912.
Clancy, W.J. (1983), "The epistemology of a rule-based expert system: a framework for explanation", *Artificial intelligence*, Vol. 20 No. 3, pp. 215-251.
Cunis, R., Günter, A. and Strecker, H. (1991), *Das PLAKON-Buch: ein Expertensystemkern für Planungs-und Konfigurierungsaufgaben in technischen Domänen*, Springer, Berlin.
Durhuus, B. and Eilers, S. (2014), "On the entropy of LEGO®", *Journal of Applied Mathematics and Computing*, Vol. 45 No. 1-2, pp. 433-448.
Gembarski, P.C., Bibani, M. and Lachmayer, R. (2016), "Design Catalogues: Knowledge Repositories for Knowledge-Based-Engineering Applications", In: Marjanović, D., Štorga, M., Pavković, N., Bojčetić, N., Škec, S. (Eds.), *Proceedings of the DESIGN 2016 14th International Design Conference*, The Design Society, pp. 2007-2016.
Gembarski, P.C. and Lachmayer, R. (2017), "A Business Typological Framework for the Management of Product Complexity", In: Bellemare, J., Carrier, S., Nielsen, K. and Piller, F.T. (Eds.), *Managing Complexity – Proceedings of the 8th World Conference on Mass Customization, Personalization and Co-Creation (MCPC 2015)*, Springer, Berlin, pp. 235-247. https://doi.org/10.1007/978-3-319-29058-4_18
Gembarski, P.C., Li, H. and Lachmayer, R. (2017), "KBE-Modeling Techniques in Standard CAD-Systems: Case Study Autodesk Inventor Professional", In: Bellemare, J., Carrier, S., Nielsen, K. and Piller, F.T. (Eds.), *Managing Complexity – Proceedings of the 8th World Conference on Mass Customization, Personalization and Co-Creation (MCPC 2015)*, Springer, Berlin, pp. 215-233. https://doi.org/10.1007/978-3-319-29058-4_17
Hayes-Roth, B. (1995), "An architecture for adaptive intelligent systems", *Artificial intelligence*, Vol. 72 No. 1, pp. 329-365.
Hesse, S., Krahn, H. and Eh, D. (2012), *Betriebsmittel Vorrichtung: Grundlagen und kommentierte Beispiele*, Carl Hanser, Munich.
Hirz, M. et al. (2013), *Integrated computer aided design in automotive development*, Springer, Berlin.
Hopgood, A.A. (2012), *Intelligent systems for engineers and scientists*, CRC press, Boca Raton.
Koller, R. (1991), "CAD- und Expertensysteme der Konstruktion – Stand und Möglichkeiten", In: *Berechnung, Gestaltung und Fertigung von Schweißkonstruktionen im Zeitalter der Expertensysteme*, DVS-Berichte, Düsseldorf, pp. 1-3.
La Rocca, G. (2012), "Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design", *Advanced engineering informatics*, Vol. 26 No. 2, pp. 159-179.
Li, H., Gembarski, P.C. and Lachmayer, R. (2018), "Template-Based Design for Design Co-Creation", In: *Proceedings of the 5th International Conference on Design Creativity (ICDC2018)*, Bath, United Kingdom, 31.01. – 02.02.2018, pp. 387-394.
Lutz, C. (2012), *Rechnergestütztes Konfigurieren und Auslegen individualisierter Produkte: Rahmenwerk für die Konzeption und Einführung wissensbasierter Assistenzsysteme in die Konstruktion*, Dr. Hut, Munich.
Marcus, S. and McDermott, J. (1998), "SALT: A knowledge acquisition language for propose-and-revise systems", *Artificial Intelligence*, Vol. 39 No. 1, pp. 1-37.
Milton, N.R. (2008), *Knowledge technologies*, Bd. 3, Polimetrica sas, Milano.
Rong, Y. and Zhu, Y. (1999), *Computer-Aided Fixture Design*, Taylor & Francis, Milton Park.
Schreiber, G. (2000), *Knowledge engineering and management: the CommonKADS methodology*, MIT press, Cambridge.
Schreiber, G. (2008), "Knowledge engineering", *Foundations of Artificial Intelligence*, Vol. 3, pp. 929-946.
Shah, J.J. (2009), "Designing with parametric cad: Classification and comparison of construction techniques", *Geometric Modelling*, pp. 53-68.
Skarka, W. (2007), "Application of MOKA methodology in generative model creation using CATIA", *Engineering Applications of Artificial Intelligence*, Vol. 20 No. 5, pp. 677-690.
Stokes, M. (2001), *Managing engineering knowledge: MOKA: methodology for knowledge based engineering applications*, Professional Engineering Publishing, London.
Vajna, S. et al. (2009), *CAx für Ingenieure: eine praxisbezogene Einführung*, Springer, Berlin.
VDI (2017), *VDI-Richtlinie 5610 Blatt 2:2017-05: Wissensmanagement im Ingenieurwesen – Wissensbasierte Konstruktion (KBE)*, Beuth, Berlin.

https://doi.org/10.1017/dsd.2020.313 Published online by Cambridge University Press