AN AGENT-BASED SYSTEM FOR SUPPORTING DESIGN ENGINEERS IN THE EMBODIMENT DESIGN PHASE

Martin Kratzer[1], Michael Rauscher[2], Hansgeorg Binz[1], Peter Goehner[2]
(1) Institute for Engineering Design and Industrial Design (IKTD), University of Stuttgart
(2) Institute of Industrial Automation and Software Engineering (IAS), University of Stuttgart

ABSTRACT
Today, product development is dominated inter alia by a complex and interdisciplinary working environment. As a consequence, several problems come up: design engineers need a huge amount of knowledge to design high quality products, non-compliance with basic requirements and non-compliance of products with different design guidelines like Design for X. One possibility to overcome these problems is the use of knowledge-based systems in engineering design. This paper follows a different approach: the use of an agent-based system in combination with a CAD System to support design engineers in the embodiment design phase. This system is some kind of agent-based design support system (MADS) called ProKon (Proactive Support of Design Engineering Processes with Agent-based Systems). Because of its novelty, this approach has to be investigated and several problems have to be tackled. Firstly, it is not clear in which way the MADS has to be built up. Secondly, there is no structured method for integrating engineering design knowledge into MADSs. Due to these two problems, there is no generic procedure for developing these MADSs. Answers regarding these problems will be given in this paper.

Keywords: Agent-based Design Support System, MADS, Engineering Design Knowledge

1 INTRODUCTION

1.1 Motivation
Currently, design engineers work in a complex and interdisciplinary working environment and need a huge amount of knowledge to design high quality products [1]. They have to consider a variety of relations between machine elements (e. g. shaft and hub) and take care of constraints concerning different design aspects, like functionality, manufacturing technology or costs. Even experienced design engineers fail to handle this complexity and the possibility of mistakes rises. For this reason, the amount of time for improvement and re-engineering of products rises too [2], [3]. While working on interdisciplinary products, design engineers have also to deal with different disciplines so that they have to gather a lot of information. This determines the amount of time for information research, assessment and transformation into the product development process [4], [5]. A further important aspect is the compliance of machine elements with basic requirements. To meet the given requirements for the development of a complex product is a challenge for design engineers in every phase of the product development process. A last problem is the low level of design according to certain “Design for X”-criteria (DfX). By raising this level, it is possible to increase the efficiency of product-related business units, e. g. manufacturing and assembly [6].
Accordingly, one possibility of meeting the problems named above is using knowledge-based systems (KBS). A characteristic feature of common KBS is the separation of knowledge bases from inference mechanisms. Therefore, knowledge engineers are able to change and maintain knowledge stored in knowledge bases without touching inference mechanisms [7]. Due to an increasing number of tasks the engineer has to deal with and the rising amount of knowledge he has to process, the support by common knowledge based systems is no longer sufficient. The goal is to have a design support system that proactively checks the CAD model while the engineer works on it. The system possesses knowledge about standards and other boundary conditions and monitors the CAD model. If the system detects a problem (e. g. the shaft-hub connection does not comply with the requirements), it interferes and informs the engineer about the problem. Furthermore, the system will try to find a solution

regarding the problem which may consist of several geometrical and semantical adjustments, e. g. adjusting the diameter of a bearing surface. These adjustments are presented to the engineer and will be implemented automatically with the engineer's acknowledgement.
A first application scenario is the shaft-hub connection. It is a relatively simple example, but it allows the demonstration of all functions of the MADS. If, for example, a design engineer wants to modify an existing interference fit there are three fundamental possibilities: changing the material, changing the applied loads or changing the geometry. Although changing the material is the easiest way, it has the largest impact and should be handled with care. Changing the applied loads is difficult or impossible due to the specifications predefined by the customer. Most design engineers would change the geometry. In the first instance of this application scenario, the required torque could not be transferred by the interference fit. Furthermore, there are several geometric constraints, in which the design engineer has to search for solutions.

1.2 Problem description and objectives

Considering the previously discussed application scenario, the system has to monitor the shaft-hub connection represented as a CAD model while the design engineer revises the interference fit. Therefore, it must work autonomously without activation by the engineer. If problems are detected in the CAD model (e. g. the pressure between shaft and hub is too high, so that the material begins to yield), the system should interfere actively and thus needs an active behaviour. Due to the uniqueness of a design, the system should be able to deal flexibly with different problems. Furthermore, it needs a certain flexibility of the knowledge base so that the system can be extended with further knowledge, e. g. new technologies or standards. Finally, a design is characterised by many dependencies between and within single machine elements. Hence, the system has to handle complex problems where the dependencies on the created design are very specific and not predictable during the development of the support system. An example is the mutual dependency between shaft, hub, bearing, locking ring etc. In specific cases, this level of complexity could be too high for design engineers. Thus, the support system has to be developed generically.
A possibility to meet these challenges is to use an agent-based system. Such systems are able to deal with the given complexity. An agent-based system also provides an active behaviour. Furthermore, these systems are very flexible so that they can be easily extended [8]. According to the presented functionality of the agent-based system, the following problems have to be solved:

- Several agent-based systems exist in engineering design [9], automation technology, e-commerce etc. However, it is not clarified in which way agent-based systems have to be built-up to fulfil the suggested functionality in the domain of engineering design and CAD systems.
- Agent-based systems are also knowledge-based systems and therefore *knowledge-intensive* and *knowledge-sensitive*. It is not yet clarified in which way agent-based systems process knowledge in the domain of engineering design and which knowledge representation mechanisms are necessary to represent engineering design knowledge.
- Due to the last two problems, an overall procedure for developing MADS has to be developed.

Certain objectives for the actual paper could be derived from these problems:

- First, a generic procedure for developing this MADS is necessary to include all boundary conditions. This procedure should contain an agent model in which the agents are modelled and a knowledge model, in which the knowledge integration process is described methodically.
- Second, it has to be investigated how a complex system in engineering design (interaction of several machine elements) has to be represented with agents. These agents could perform certain tasks and strive for results which are channelled within the agent model.
- Thirdly, the previously mentioned knowledge model should contain a structured approach for integrating engineering design knowledge into MADSs. This model should be based on common structuring approaches in engineering design science and fundamentals of knowledge engineering.

This paper gives solutions for the presented objectives. Therefore, it is structured as follows: The second section describes the fundamentals of software agents, because this topic is not widespread enough in the community of engineering design science. Afterwards, the overall development of the MADS as well as the specific models is presented in Section 3. The existing prototype is described in Section 4. A conclusion and an outlook can be found at the end.

2 FUNDAMENTALS OF SOFTWARE AGENTS

Due to a closer relationship to software engineering and informatics, the fundamentals of software agents are described in this section. Software agents are encapsulated (software) entities with defined objectives [8]. An agent tries to reach its objective by acting autonomously. It interacts with its environment and with other agents, while keeping a persistent state.
It is able to plan and execute activities by itself and react on unpredictable situations by changing its plan. The internal structure of an agent is explained as follows [10]. As shown in Figure 1, an agent mainly consists of three knowledge elements: an internal model of its environment, its goals and its abilities.

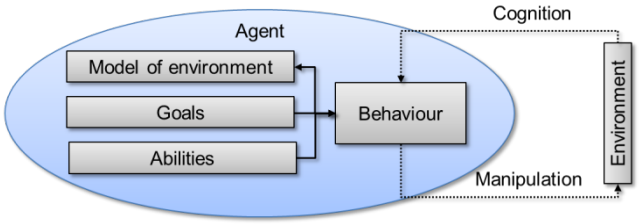

Figure 1. Internal structure of an agent

The first element, the internal model of the environment, is the view of the agent to its environment. It contains information the agent gets from its environment. The internal model is updated every time the agent gets new information. The goals, as a second element, are the intentions of the agent, which they try to reach by using their abilities. These abilities, which are the third element of an agent, represent an agent's possibilities to act. All these elements influence the behaviour of an agent or, more precisely, the agent determines its behaviour according to these elements. The agent itself is embedded within an environment. For example, this environment consists of the surrounding software system, including interfaces to other systems on the one hand and containing other agents on the other hand. An agent is able to interact with its environment by e.g. noticing changes or by manipulating objects of the environment. An agent monitors its environment, compares the environment with the goals and tries to manipulate it according to them. The possible interactions are limited to the agent's abilities.

3 RESULTS

Solutions for the introduced objectives are shown in this section. After presenting the overall procedure for developing the agent-based design support system (MADS), the two key topics, the *agent model* and the *knowledge model*, are indicated. By modelling these two models, it is absolutely necessary to be aware of mutual dependencies between them. They are not only influencing each other, but also intertwined with each other.

3.1 Overall procedure for developing MADS

At the beginning of this project, it was very important to specify a methodical procedure for developing the whole system. In this case, specifying means doing a literature research and considering all related aspects like requirements, agent properties, properties of engineering design knowledge etc. There are many approaches for developing knowledge-based systems on the one hand [11-15] and agent-based systems regarding the knowledge engineering paradigm on the other hand [16], [17]. A fundamental idea is to combine these two approaches within one basic procedure. Furthermore, since the late eighties, the community distinguished between two aspects of knowledge engineering [7], [15]: the transfer view and the modelling view. In this project, the modelling view was used, because it was established for large knowledge bases and a hybrid representation mechanism (rule-based, frame-based, ontology-based). Thus, the transfer view is not efficient and effective enough. The overall procedure is shown in Figure 2. Basically, it is subdivided into the parts *modelling* and *operating*. Four models were identified. At first, the organisational structure of the company, in which the system has to be installed, will be considered in the requirement and organisation model which is leaned on the first model of CommonKADS [13]. It has to be clarified if there is a real need for a MADS and if so, which tasks should be assumed by the MADS. Afterwards, requirements have to be established. The agent model is dealing with the agents themselves (roles, goals, behaviours etc.), which is the micro-concept, and the cooperation/communication of the agents (macro-concept). Afterwards, the knowledge will be modelled within the knowledge model.

Regarding Figure 2, it is important to consider what has been done in the other model. At last, the system as a whole should be modelled. This includes aggregating all previous models, setting up the interface to a CAD system and determining which hard- and software should be used etc.

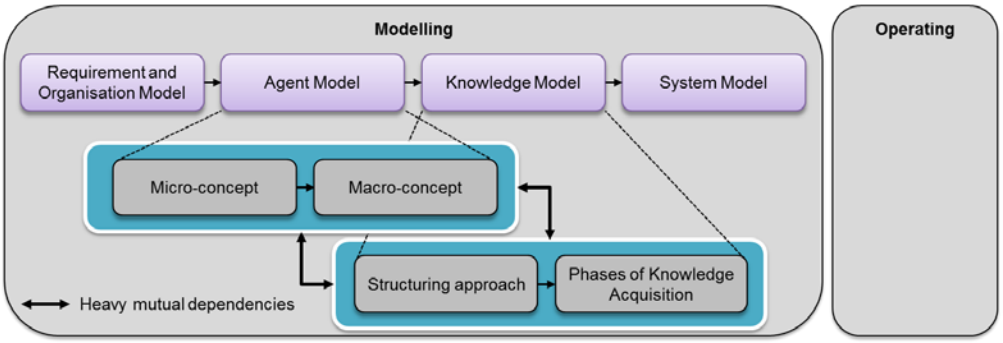

Figure 2. The overall procedure for developing agent-based design support systems

After modelling of the system and elimination of all errors, the system can be operationalised with techniques from software engineering, expertise of CAD systems, respectively fundamentals of CAD and programming, of course. Operationalisation is not only implementing as it is more practical as modelling where the knowledge engineer has to take care of many interrelated and interdisciplinary aspects. In the next section, the two most important models, the *agent model* and the *knowledge model*, will be discussed.

3.2 Agent model

Multi-agent systems consist of several agents which are normally of different types and which follow certain objectives. The following sections describe tasks identified in the project and different agent types fulfilling them.

3.2.1 Agent tasks

The main goal of the MADS is to preserve the consistency of the CAD model. Assuming a consistent CAD model, inconsistencies may only occur after a change by the engineer. Therefore, the first task is to detect changes, such as changes of the requirements list or the CAD model itself. To detect differences, the corresponding objects are permanently compared to a reference. Having detected a change, other objects, which are constrained with the changed object, must be informed as their consistency might be affected too.

Each time a change is detected, the CAD model must be checked. This task can be divided into several sub-tasks. The first sub-task is to coordinate the whole process of the check. This means, to initiate the check, delegate the execution and decide about the result. The second task is the execution of the check itself. This task is distributed to several agents as described in the next section. For the execution of the check, information about the CAD model, e.g. geometrical data, is needed. Therefore, this information must be provided by the agents.

If the result of a check is negative, respectively, if inconsistencies exist, a solution for the detected problems must be found. This solution consists of adjustments of the CAD model, which have to be implemented to recover the consistency. One task is to coordinate the solution-finding process, which means delegating the execution as well as managing and evaluating the different solutions. The execution is distributed to several agents in the same way as the check of the CAD model. Due to the fact that a solution consists of several adjustments, they must be managed for each object of the CAD model and implemented in the CAD system after the approval of a certain solution.

3.2.2 Agent types

The agents of the system can be arranged in two main groups. The first group contains agents which are responsible for project-specific objects, like machine elements. The second group contains agents which follow common tasks and which are independent from a certain project, like checking the CAD model according to different rules. The agent types of these two groups are presented in detail in the following paragraphs.

Object Agents (several types and instances):
Every agent of the first group represents an object in the CAD model. These objects are single parts, connections between parts or assemblies, which are modelled in the CAD system. Each object is monitored by a corresponding agent. Therefore, *Part Agents* are responsible for parts, *Connection Agents* are responsible for connections and *Assembly Agents* are responsible for assemblies. Besides the geometric model in the CAD system, another project-specific object exists to monitor the requirement specification list. This list is represented by a *Requirements List Agent*. Figure 3 shows the responsibility of the agents, regarding the previously presented application scenario.

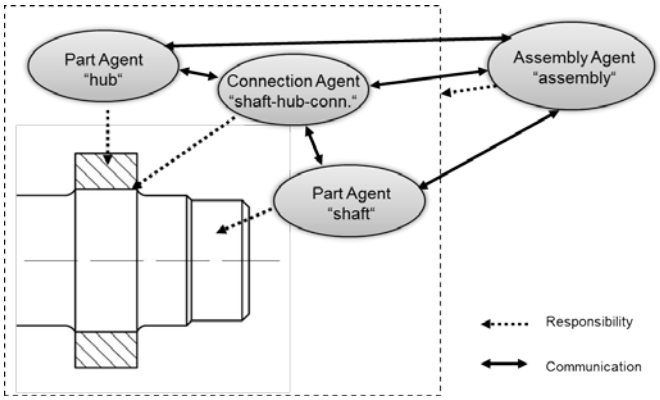

Figure 3. Example for the responsibilities of the Object Agents

All agents of this group have similar tasks and just differ slightly in the type of object they are responsible for. Hence, in the following sections, these agents are simply called object agents. The goals of these agents are detecting changes of the objects, providing information about the object, collecting adaptions of the object and implementing them into the CAD system.
The second group consists of agents which have common tasks and represent the core of the system. An overview of these agents can be found in Figure 4. These agents apply the knowledge integrated into the system, consisting for example of rules and formulas. Hence, the agents of this group perform the check of the CAD model and retrieve solutions to problems they have found. The different types of agents are presented in the following.
Management Agent (one instance):
The first agent of this group is the *Management Agent* which has the task to coordinate the check as well as the solution-retrieving process. It is informed if a change in the CAD model is detected and it initiates the check of the CAD model. It delegates other agents of the group, called *Aspect Agents*, to perform the check of the whole CAD model or only parts of it. When those have finished, the Management Agent interprets the results. If problems in the geometric model exist, like parts that do not fit together, it starts the problem-solving process by delegating the Aspect Agents to solve the different problems. During the process, it collects and manages the retrieved adjustments and summarises them to solutions. If a solution is found, the Management Agent presents this solution to the design engineer and delegates the implementation of the adjustments to the corresponding agents, after having received the acknowledgement from the engineer. Otherwise the solution is discarded and the change by the engineer is rejected.
Aspect Agent (several instances):
Beside the Management Agent, this group contains Aspect Agents. Each of these agents is responsible for an aspect or a specific view on the CAD model. An aspect might be the functionality of the design, the used materials or the planned manufacturing technologies (i. e. Design for X-guidelines [6]). An Aspect Agent performs the check of the CAD model according to its aspect. Therefore, this type of agent possesses knowledge about its aspect in the form of rules and formulas. Having finished a check, the agent might be instructed to solve the problems it has found. It analyses the problem by retrieving the corresponding rules and formulas. Then it identifies the adjustable values and tries to adjust them in the right manner. Therefore, it requests the Object Agents, which are responsible for the corresponding value, to change it to the desired value. Additionally, they have to inform the Management Agent about the change.

Specialist Agent (several instances):
As already mentioned, the Aspect Agents possess knowledge about their aspect which could be very universal. Hence, they are supported by S*pecialist Agents*. Specialist Agents are experts in certain cases, e.g. for the check of certain types of parts or certain types of connections between parts. They possess the specialised rules and formulas from standards or other sources. Therefore, they are able to provide detailed information for the Aspect Agents. In the same way, they have to support the Aspect Agents to adjust certain detailed values. Moreover, Specialist Agents have an approach similar to that of the Aspect Agents. A Specialist Agent analyses its rules and formulas and identifies adjustable values. After that, it requests the correspondent Object Agents to adjust the values and informs the Management Agent about the change.

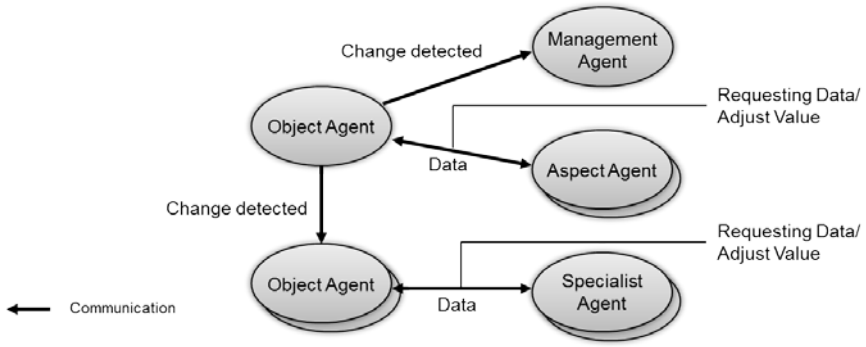

Figure 4. Communication between Object Agents and common agents

The next section describes how the agents detect problems in the CAD model and how they retrieve solutions.

3.2.3 Problem detection and solution process

The detection of changes in the geometric model is done by the group of Object Agents. Taking the example of a single part, these are connections to other parts and the surrounding assembly. After having informed the involved agents, the original agent informs the Management Agent about the change as well (1, see Figure 5).

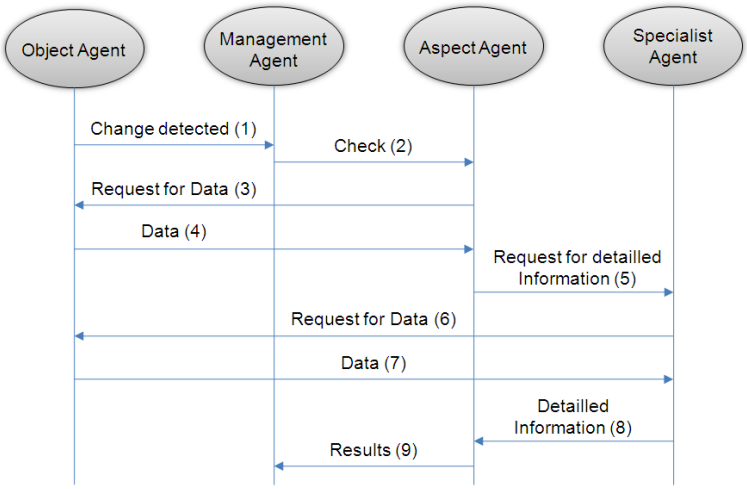

Figure 5. Problem Detection Process

The Management Agent determines all Object Agents whose objects have been changed or might be affected by a change and requests all Aspect Agents to check the changed objects (2, see Figure 5).
The Aspect Agents determine the type of the object, e.g. single part, connection or assembly, and apply their corresponding rules by supplying them with data from the object. They request the needed data, e.g. geometrical information or the used material, from the Object Agents (3). The Object Agent determines the desired information in its internal model of the object and sends it back to the requesting Aspect Agent (4). Since the Aspect Agents only possess common rules, the information they get from the Object Agents is not sufficient in most cases. They have to request support from the

Specialist Agents. Therefore, they retrieve the corresponding specialist and request the specialised information from the responsible Specialist Agent (5).
The Specialist Agent is able to generate the desired information by using its specialised rules and formulas, applying the data it requests from the Object Agents (6, 7).
Having collected all information, the Specialist Agent is able to generate the desired information for the requesting Aspect Agent and answers to its request (8). After this support, the Aspect Agent is able to proceed with the check and informs the Management Agent about the results of the check (9). Therefore, it attests the correctness of the CAD model according to its aspect or provides a list of problems, respectively rules, which are not fulfilled. The Management Agent collects the results of all aspects and of every changed object to interpret them.
Having all results, the Management Agent has to evaluate the existing problems in the CAD model and starts the problem-solving process (see Figure 6). Therefore, it instructs the Aspect Agents to solve their reported problems (10).

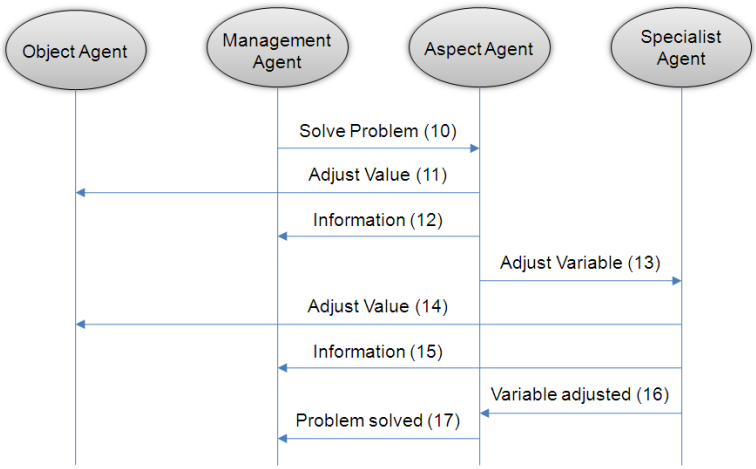

Figure 6. Problem Solution Process

The Aspect Agent analyses the problems and links them with the rules which were not fulfilled. For each rule, it determines why it was not fulfilled. Normally a rule is not fulfilled if a variable has not the desired value. In this case the agent has to determine what could be done in order to change the value. Therefore, it has to know where the value comes from. If it was calculated by using the aspects formulas, it analyses the corresponding formula for adjustable variables. These adjustable variables, respectively their values, might be the result of further formulas which have to be analysed recursively. At the end of this analysis, the Aspect Agent has a list of all variables that can be adjusted in order to change the resulting value and to fulfil the broken rule. Having this list, it will change the variables to solve the problem. Some of these variables may be linked directly with a value of the CAD model. In this case the Aspect Agent requests the corresponding Object Agent to adjust the value (11, see Figure 6). If the variable stems from a Specialist Agent, the Aspect Agent requests that agent to adjust the value (13) and waits for the confirmation (16).
An Object Agent is able to change the value, e.g. a geometrical dimension, directly by editing the object it is responsible for. A Specialist Agent may have calculated the value using one of its formulas and has to analyse the formula the same way an Aspect Agent does. It identifies the adjustable variables and requests the corresponding agents to adjust them (14). No matter how many steps are necessary to retrieve the adjustable variables, at the end, the Object Agents must take the responsibility because they possess the project-specific data which is the only data that can be changed. If one of these agents is requested to adjust such a variable, it changes the value of that variable in its internal model concerning the CAD model. The model in the CAD system stays unchanged at that moment. Using this proceeding, the agents are able to try different solutions without affecting the real CAD model.
It is important to report each adjustment of a variable to the Management Agent (12, 15) so that this agent is able to manage different solutions. If all problems collected by the Management Agent are processed (17), the first iteration has finished and the agent launches another check of the CAD model. This check, which is needed to verify the problem solution of the agents, is carried out as described

above. If problems are still found, the solution process is repeated until there are no more problems. If some problems cannot be solved by the agents on their own and if the process will result in an infinite loop, the Management Agent aborts the process and informs the design engineer about the situation. Provided with the current problems in the CAD model, he is able to fix them with less effort than without the support of the agents. If the agents have found a solution and are able to fix the CAD model, the Management Agent presents the solution, respectively the adjustments, to the design engineer. He can have a look at the solution and make further changes or accept it without changes. If he accepts the solution, the Management Agent instructs all Object Agents, to implement the adjustments into the CAD system. By automating the implementation of the adjustments, the effort for the design engineer is reduced. The result of the described process is a CAD model which is checked according to different aspects and requirements. A further change by the design engineer will restart the process.

This section was dealing with agent tasks and types and how they detect problems. To detect problems, agents need knowledge in form of formulas and rules etc. At the end of this section, the question is, how can this knowledge be implemented in a structured and methodical way? The answer is given in the next section.

3.3 The knowledge model

Based on the agent model, which was discussed in the previous section, this section describes how engineering design knowledge has to be modelled and operationalised in a methodical way to get a functional MADS. In detail, it serves as a guideline for knowledge engineers and has a prescriptive character. Every step is predetermined and has to be adapted to the given facts in each case. As described in Section 3.1, the modelling view is used and therefore, the knowledge has to be modelled holistically. If there are no redundancies, overlaps and inconsistencies within this first phase, the whole MADS can be operationalised (knowledge + agents + communication + architecture). An important aspect is to build-up this knowledge model in a generic way so that it can be used in combination with every agent-based approach in engineering design and not only in this particular project.

In detail, the knowledge model is based on the structuring approach by Roth [18] and on the common steps of knowledge acquisition (knowledge elicitation, knowledge analysis, knowledge representation). Figure 7 presents the final knowledge model within this project. The agent model appeared to be the only input with certain information: micro-concept (classes of agents with roles, goals, behaviours etc., cf. Section 3.2.1 and Section 3.2.2) and the macro-concept (communication between agents, collaboration and cooperation, cf. Section 3.2.3).

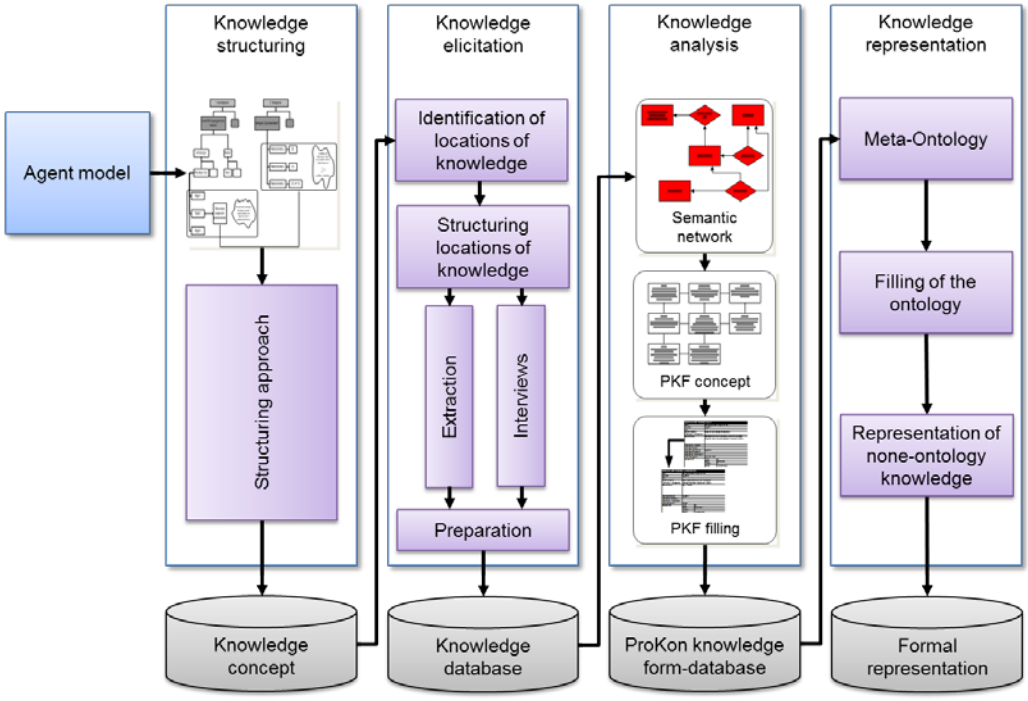

Figure 7. The final knowledge model within the ProKon-project

After that, four different phases have to be processed by the knowledge engineer. The outcome of every phase is a set of results. These results are symbolised by a database symbol below every phase (see Figure 7). The overall result of this knowledge model is a formal representation of the engineering design knowledge. This formalised knowledge has to be operationalised within the next steps. Subsequently, a brief description of the each phase follows.

3.3.1 Knowledge structuring

In this phase, a knowledge concept has to be processed. At first, the knowledge engineer turns the results of the agent model into a knowledge allocation which shows what knowledge is suitable for a particular type of agent. For example, aspect agents need knowledge about certain aspects in design, e. g. guidelines in embodiment design. After that, the structuring approach follows. The goal of this step is to consider the roles of a particular agent and to assign a set of knowledge characteristics to them, which are described by the structuring approach. In this way, it is possible to combine an agent-role with one knowledge type, several knowledge forms, several locations of knowledge, one knowledge character and a probabilistic quality of knowledge. For example, the aspect agent for manufacturing design needs detailed knowledge about the production of certain parts (knowledge type: *goal knowledge*). This knowledge is built up through rules, facts and information in tables (knowledge forms: *rules, attributes, relations, values and tables*). Furthermore, this knowledge can be found in a general manner in standards and books (location of knowledge: *standards and books* → character of knowledge: *explicit*).
The quality of the knowledge is probably correct. In total, four knowledge types, twenty knowledge forms and eight locations of knowledge were identified. In turn, the four general knowledge types can be subdivided into several subtypes, e. g. goal knowledge → expert knowledge, experience knowledge, product knowledge etc.

3.3.2 Knowledge elicitation

At first, based on the results of the previous phase, the knowledge engineer has to identify the locations of knowledge in detail. So, he has to consider the structuring approach with the characteristics of the knowledge and find out certain explicit and implicit knowledge. A structuring of the single locations follows. After that, depending on the character of knowledge, it is important to make a distinction between explicit and implicit knowledge in the remaining process. Explicit knowledge (that means, only information which is written down) could be externalised by extraction. Apart from that, implicit knowledge could only be externalised by interviews. So, experts have to be interviewed by knowledge engineers in certain ways: open interviews, guided interviews and structured interviews [19]. The difficulty is to choose carefully between these three options, use one or more options correctly regarding available experts, respectively regarding order, and being sensitive when dealing with experts. One issue is that experts mostly do not know what they know and, furthermore, they cannot externalise the knowledge in the way the knowledge engineer wants it. Executing interviews is much harder than extracting explicit knowledge, but it is necessary to get a holistic view on the knowledge domain. The elicited knowledge has to be prepared and filled into a database. In this project, a tool for the management of literature was used. This solution works very well in this project and more than thousand knowledge elements about shaft-hub connection, engineering design in general and basics in mechanics, respectively mathematics, were elicited.

3.3.3 Knowledge analysis

Considering the knowledge database, first the knowledge engineer arranges terms and concepts by using a semantic network. The advantage of this is a general overview of the knowledge domain. All terms have to be connected with standardised relations which are typical for semantic networks, e. g. *isA, isPartOf, consistsOf, hasFunction, hasProperty, hasBehaviour* etc. Important is that the knowledge engineers have to choose between these relations and do not create a huge set on his/her own. This fact leads to a more standardised view which is much easier at the beginning. Several good experiences with this procedure have been made within the project. Second, the knowledge is analysed, arranged and represented informally using the so called *ProKon-knowledge-forms* (PKF). These are leaned on the ICARE-forms by Stokes [10] and analyse, arrange and represent knowledge with the help of the structuring approach. A total of seven different forms were established: entity,

name, table, formula, rule, condition and image. With these seven forms, it is possible to analyse every knowledge element in engineering design.

3.3.4 Knowledge representation

The last phase is about representing the analysed engineering design knowledge formally. First, based on the PKF *name* and *entity*, a meta-ontology has to be established. This step is very important, because an ontology might be very efficient and effective if the foundation is defined well. In addition, an agent-based system needs an ontology for communication and cooperation aspects. In this project, two main super classes were defined: *concept* and *synonym*. The reason behind that distinction is that most concepts (e. g. arise) have a synonym (e. g. occur). Below the super class concept, several classes were identified: design element, property, function etc. It is only now that instances occur instead of classes. With predefined relations in the entity-form, it is possible to connect two instances; e. g. Inference fit - *hasFunction* - transfer of torque and axial force. Afterwards, the whole ontology has to be filled with knowledge from the PKF. At last, the non-ontological knowledge should be represented formally. At this point, it is important to push the limit regarding the level of formalisation, because the higher the level, the easier it is for the operationalisation.

After describing the overall procedure of how to develop a MADS with the agent model and the knowledge model, the next section shows how the prototype was built (operationalisation of the agent model and knowledge model) and which results were achieved.

4 PROTOTYPE

The ideas presented in this paper were realized in a prototype. The prototype is based on Java by means of the multi-agent platform JADE. It consists of 19 functional agents which represent three aspects (design for functionality, design for economical use of materials, design to cost) and six specialists (interference fit, key fit etc.). The simplified CAD system contains an assembly with an interference fit, another with a key fit and a requirements list. At the current state, the prototype comprises about 20 rules and 30 formulas. Furthermore, it contains a database for material information. With this implemented knowledge, the MADS is able to detect geometrical and material changes as well as changes concerning applied loads in the CAD model and check it for inconsistencies. It is also able to find a solution for detected inconsistencies, present them to the engineer and implement them. In Figure 8, the screenshot on the left hand side shows the simplified CAD system including the resulting adjustments for a change. On the right, an extract of the agents' communication is shown, logged by a sniffer agent (cf. Figure 5 and Figure 6). Assuming that the designer changes the geometry of the shaft of the included interference fit, the agents detect that change and check the CAD model. The aspect agents report some emerged problems to the management agent. This agent instructs them to solve the problems and then initiates a check again. After a few iterations no more inconsistencies exist and the Management Agent reports the needed adjustments of the CAD model to the designer as shown in Figure 8. By accepting these adjustments, the designer delegates certain Object Agents to implement them. After that, the consistency of the CAD model is recovered.

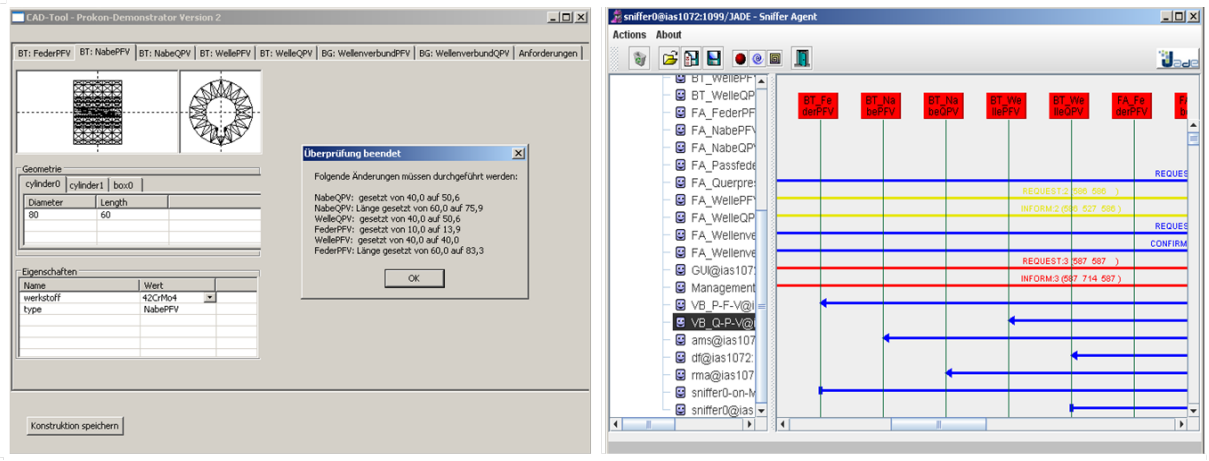

Figure 8. Screenshot of the prototype and the agent sniffer

Regarding the previously discussed application scenario, the prototype is in the position to detect that the geometry and/or the material is not suitable for transferring the torque from the shaft to the hub or vice versa. In Figure 8, this application scenario was processed. For this inconsistent interference fit, solutions were presented to the design engineer. In parallel, solutions for an equally possible key fit are presented in Figure 8 as well so that the design engineer is able to choose between these two eventualities. In this application scenario, several basic and special rules for interference fits and key fits were processed. A valid solution should fulfil every rule. For example, a rule for guaranteeing that the hub does not drift from the shaft is included. This involves also the fact that this rule is only applicable in combination with rotary and alternating bending moments.
At the moment, this prototype is using a hybrid problem-solving method (PSM). It combines a *random walk approach* with a *gradient algorithm*. Concerning the groove length of an interference fit, which has an impact on the calculation of the transferred torque, this length will at first be changed by a random value in a prescribed direction. Second, if this direction is correct, which could be verified by the gradient-algorithm, the groove length will be changed again until the prescribed torque is reached. This PSM works well with a small number of rules to be checked. But, if this number increases, solutions are not guaranteed anymore. Probably, the use of a more efficient PSM is better for these specifications.

5 CONCLUSION AND FUTURE WORK

This paper presents an approach for supporting design engineers within the embodiment design with an agent-based design support system (MADS). All three introductorily mentioned problems were tackled and discussed in this paper. Concerning the core of the agent model, it was important to align the structure of the agents on design guidelines. That means that every aspect agent is responsible for the compliance of a specific design guideline, e. g. agent for design for manufacturing. Apart from that, the specialist agents are dealing with delimited fields of interest, e. g. knowledge about designing an interference fit. This is the reason why the structure is modular and could be enhanced for new design guidelines and machine elements. Thus, it is possible to transfer this agent approach to further applications in mechanical and electrical engineering, because design guidelines and system elements occur in any field. Therefore, it is not always necessary to connect the MADS to a CAD system. Furthermore, the knowledge model, which is based on the fundamentals of knowledge engineering and on the fundamentals in engineering design knowledge, was presented. After a structuring approach, the three typical stages of knowledge acquisition followed. In the stage of knowledge elicitation, it is possible to deal with implicit and explicit knowledge as well. This opens up the applicability to all imaginable knowledge domains, not only the domain of interference fits. The stage of knowledge analysis is the most important stage. Knowledge engineers can get a holistic view on the knowledge domain, on the one hand, and are able to structure the knowledge in an informal way to an easy integration on the other hand. Furthermore, characteristics of agent-based systems are integrated as well. In the last stage, an ontology is predefined with a meta-model and filled within the next step. In summary, this knowledge model is agent-oriented, but flexible enough for applying single stages by itself, and transferred to other problems. To sum up, it can be said that all postulated problems are tackled. Only the connection to the CAD system is challenging. In the future, beside this connection problem, a user-friendly and user-defined knowledge acquisition component will be developed. Therefore, design engineers will be able to integrate missing knowledge on their own. This is an important step to make this MADS applicable in industry and research.

ACKNOWLEDGEMENT

The authors are grateful for the support of their work by the Deutsche Forschungsgemeinschaft (DFG) within the research project ProKon (BI 746/4-1 and GO 810/20-1).

REFERENCES

[1] Lindemann U., Maurer M. and Braun T. *Structural Complexity Management*, 2009 (Springer Press, Berlin, Heidelberg)

[2] Ameri F., Summers J., Mocko G., Porter M. Engineering design complexity: an investigation of methods and measures. *Research in Engineering Design*, 2008, 19, pp. 161-179

[3] Eckert C., Clarkson P. J., Zanker W. Change and customisation in complex engineering domains. *Research in Engineering Design*, 2001, 15, pp. 1-21

[4] Broy M. Die Zukunft liegt im Software und Systems Engineering. *ATZelektronik*, 2009, 4, pp. 9
[5] Gehrke M. *Entwurf mechatronischer Systeme auf Basis von Funktionshierarchien und Systemstrukturen*, 2005 (University of Paderborn, Heinz Nixdorf Institute)
[6] Meerkamm H., Koch, M. Design for X. Clarkson, J. Eckert, C. (editors): *Design process improvement: A review of current practice*, 2005, pp. 306-323 (Springer Press, London)
[7] Studer R. Knowledge Engineering: Principles and Methods. *Data & Knowledge Engineering*, 1998, 25, pp.161-197
[8] Jennings N. R. On agent-based software engineering. *Artificial Intelligence*, 2000, 117, pp. 277-296
[9] Mori T., Cutkosky M. R.: Agent-based collaborative design of parts in assembly. *Proceedings of the DETC'98 ASME Design Engineering Technical Conferences*, 1998, pp. 1-8
[10] Maurmaier M., Wagner T. Development of Embedded Software Systems with Structured Components and Active Composition Support. OMER 3 - 3rd *Workshop on Object-oriented Modeling of Embedded Real-Time Systems*, 2005, pp. 37-42
[11] Stokes M. *Managing engineering knowledge. MOKA: methodology for knowledge based engineering applications*, 2001 (Professional Engineering Publishing, London)
[12] Angele J., Fensel D., Studer R. Vorgehensmodelle für die Entwicklung wissensbasierter Systeme. *Vorgehensmodelle für die betriebliche Anwendungsentwicklung*, 1998 (Teubner, Stuttgart)
[13] Schreiber G., Akkermans H. *Knowledge engineering and management: The CommonKADS methodology*, 2002 (MIT Press, Cambridge)
[14] Milton N. R. *Knowledge acquisition in practice: A step-by-step guide*, 2007 (Springer Press, London)
[15] Angele J., Fensel D., Landes D., Studer, R. Developing Knowledge-Based Systems with MIKE. *Automated Software Engineering*, 1998, 5, pp. 389-418
[16] Iglesias C. A., Garijo M., Gonzales J. C., Velasco J. R. Analysis and Design of Multiagent Systems Using MAS-Common KADS. Singh M. P., Rao A. S., Wooldridge M. (eds.): *Intelligent Agents IV, Agent Theories, Architectures, and Languages: 4th International Workshop*, ATAL '97, Providence, Rhode Island, USA, 1997, pp. 313-327
[17] Glaser N. The CoMoMAS methodology and environment for multi-agent system development. *Lecture Notes in Computer Science: Multi-Agent Systems Methodologies and Applications*, 1997, pp. 1-16 (Springer Press, Berlin, Heidelberg)
[18] Roth D., Binz H., Watty R. Generic structure of knowledge within the product development process. *International Design Conference* (Design 2010), 2010, pp. 1681-1690
[19] Hua, J. Study on Knowledge Acquisition Techniques. Zhou, Q. (editor): *Second International Symposium on Intelligent Information Technology Application*, 2008: IITA '08; 20 - 22 Dec. 2008, Shanghai, China; proceedings. Piscataway, NJ: IEEE Computer Society, 2008, pp. 181-185

Martin Kratzer
University of Stuttgart
Institute for Engineering Design and Industrial Design (IKTD)
Pfaffenwaldring 9
70569 Stuttgart, Germany
Phone: ++49 (0)711 685-68059
Fax: ++49 (0)711 685-66219
Email: martin.kratzer@iktd.uni-stuttgart.de

Martin Kratzer is an academic assistant at the institute of Hansgeorg Binz. Kratzer holds a Dipl.-Ing. in Mechanical Engineering from University of Stuttgart. He is researching about the structural integration of engineering design knowledge into agent-based design support systems (MADS) under a grant of Deutsche Forschungsgemeinschaft (DFG).

Michael Rauscher is an academic assistant at the institute of Peter Goehner. Rauscher holds a Dipl.-Ing. in Electrical Engineering and Information Technology from University of Stuttgart. He is researching about assuring the consistency between different models of mechatronic developments applying software agents under a grant of Deutsche Forschungsgemeinschaft (DFG).