Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 225 (2023) 3592–3601

Procedia Computer Science

www.elsevier.com/locate/procedia

27th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2023)

Scalable BDI-based Multi-Agent System for Digital Design Reviews

Stefan Plappert[a,*], Christian Becker[a], Paul Christoph Gembarski[a], Roland Lachmayer[a]

[a]*Institute of Product Development, Leibniz University of Hannover, An der Universität 1, 30823 Garbsen, Germany*

Abstract

The increasing complexity in product development and the lack of knowledge exchange, for example, between development and manufacturing lead to unnecessary iteration loops and high costs. To overcome this situation, an option is the execution of a design review using multi-agent systems to provide designers with a digital assistance system for checking their CAD (computer-aided design) models regarding manufacturability using a milling process. This paper explores how to increase the scalability and robustness of multi-agent systems (MAS) for manufacturability assessment by using runtime-generated BDI (belief-desire-intention) agents and graph-based feature recognition. The applicability and validation of the presented approach is carried out by evaluating different milled part designs.

Keywords: multi-agent systems; digital design review; graph-based feature recognition; computer-aided design

1. Motivation

Conducting a design review in a collaborative environment is a technique used to review the design from different perspectives simultaneously, thus bridging the gap between the development department and other stakeholders involved in the product development process [17]. To be able to use the advantages of design evaluation even for simple components, the process for performing design reviews in a digital and interactive development environment is investigated in the literature. For this purpose, agent-based technology provides a natural means of modeling a problem based on the distributed Artificial Intelligence (AI) paradigm, as real-world entities and their interactions can be directly represented by autonomous problem-solving agents, which additionally make the current knowledge search an active personal assistant [9, 11, 13].

However, existing multi-agent systems (MAS) for digital design reviews lack scalability and robustness, especially when analyzing arbitrary geometries of CAD (computer-aided design) models. This paper intends to automatically

* Corresponding author. Tel.: +49-511-762-14989.
E-mail address: plappert@ipeg.uni-hannover.de

1877-0509
10.1016/j.procs.2023.10.354

identify manufacturing features for different CAD models and to evaluate them regarding their manufacturability, e.g., for 2.5D milling processes, using deliberative BDI (belief-desire-intention) agents generated at runtime.

This paper is organized as follows: Section 2 introduces the theoretical background and related work on MAS. The software architecture, programming of the BDI agents and the graph-based feature recognition approach of the multi-agent system for digital design reviews is explained in Section 3. Then, the application, and validation of the multi-agent system for milling designs takes place in Section 4. Finally, Section 5 summarizes and discusses the paper and outlines further research approaches.

2. Theoretical Background and Related Work

In the product lifecycle, all instances are in constant contact, procedures are explained, errors are discussed, and information is exchanged. To represent this behavior in agents, they have to act independently and make decisions. In general, an agent can be conceived as a knowledge-based system in its own right, as it can perceive, reason, adapt, learn, cooperate, and delegate in a dynamic environment to address specific problems [15]. A MAS is essentially a system structured as a set of autonomous agents that can flexibly adapt their behavior to changing operating conditions [28]. In these systems, no single agent possesses all the knowledge required to solve a problem [18], as this is communicated to each other through message exchange [4]. The information exchange in this case can proceed via *message passing* or via a *shared-memory blackboard system*. The blackboard system is based on a shared memory that all agents can access and deposit their information.

The architecture of a MAS is the abstract definition of the communication and interaction between agents. Architectures range from purely logic-based architectures to reactive or behavioral architectures that operate in a simple stimulus-response fashion, to more planning or deliberative architectures that reason about their actions. Probably the best-known deliberative agent architecture is Rao and Georgeff's BDI architecture [10]. BDI-based agents are characterized by beliefs, desires, and intentions. Beliefs represent the agent's knowledge about itself and the environment. Desires represent states of the world that the agent wishes to bring about. Intentions or plans are the means by which the agent can act to achieve its goals. BDI agents are well suited for unpredictable scenarios that require dynamic decision-making because they can choose the best plan to achieve a goal, given their current beliefs [1, 2, 5]. Furthermore, Dong et al. present self-adaptive agents, which can take on different roles depending on the environment [7]. In this context, BDI agents represent an extension of adaptive agents, which are not only able to adapt to the environment independently, but also incorporate the goals and intentions of the other agents into their decision-making using beliefs.

In related literature on MAS in engineering design, several approaches exist that support the designer in evaluating his or her design and include further aspects of the product life cycle for this purpose [21]. With this in mind, existing approaches were compared for automated analysis and synthesis capabilities with respect to CAD models and their used architectures. Regarding component manufacturability, Sun et al. [26] have developed a system that uses agents to identify the features of an Unigraphics model and evaluate, for example, the accessibility of tools. This system was firstly extended by Jia et al. [12] in terms of a modular architecture so that agents can be easily reused and updated, and secondly by Mahesh et al. [16], who have used WebMAS to allow agents to communicate over the internet and extend the system to include aspects of process planning, scheduling, and production monitoring. Medani and Ratchev [17] have presented a prototype web-enabled system based on a STEP (Standard for the Exchange of Product model data) AP224-compliant product data model for rapid assessment of product manufacturability in the extended enterprise using collaborative autonomous design and manufacturing agents. Wang et al. [27] have developed an agent-based collaborative design framework for aircraft structural parts to support manufacturability analysis and cost evaluation, during the design process. To evaluate the manufacturability of milled parts, Plappert et al. [20] have presented a behavior-based approach that has been extended to include synthesis operations, such as adding tool run-outs.

Based on related work on MAS for digital design reviews, it can be stated that existing features of the CAD program are often relied upon for the manufacturability assessment. Only Plappert et al. [20] use a kind of rule-based approach that uses point, edge, and surface relations related to used shape features in the CAD system. In addition, Wang et al. [27] developed a dynamic feature recognition based on the principle of AAG for local feature recognition. However, it lacks a specific consideration depending on the tools used. Furthermore, it was found that most MAS use reactive agents and surprisingly none of the related work uses the BDI architecture, although it allows a higher degree

of autonomy for the agents. In terms of scalability, agents are typically pre-registered and usually undergo a linear sequence of behaviours, thus insufficiently exploiting the advantages of asynchronous and network-like program flow. To address the presented research gap, the following research questions are investigated in this paper:

- How can BDI agents be built to ensure easy scalability depending on the CAD model?
- How can feature recognition be designed to support robust and tool-centric manufacturability assessment?

3. Multi-Agent System for Digital Design Reviews

Based on related work, two areas of action were identified to increase the scalability and robustness of MAS for digital design reviews. First, it should be possible to create agents on demand at runtime and equip them with communication capabilities. Second, any CAD models shall be split into features and reviewed for their manufacturability. Therefore, in the first step, the used software architecture will be presented.

3.1. Software-Architecture of the Multi-Agent System

For the high-level architecture and the division of the different modules, the presented software architecture builds on the work [22]. Therefore, the focus in the following is on the technical implementation. The most widely used holistic MAS framework in the past two decades is JADE (Java Agent Development Framework) [3], which is compliant to the standards of FIPA (Foundation for Intelligent Physical Agents) and builds a network with agent containers, using the object-oriented language Java [2]. However, the choice of Java as the sole programming language for constructing JADE agents is considered inappropriate in many situations, especially when native support for the abstractions of agent-oriented programming would be valuable [3].

Among the newer representatives of MAS frameworks is the SPADE 3.0 (Smart Python Agent Development Environment) platform, which is programmed in Python and uses XMPP (Extensible Messaging and Presence Protocol), an open protocol for instant messaging and presence notification, to enable distributed communication between humans, agents, and third-party software [19]. SPADE features a fully open, scalable, and extensible development and execution environment, transparent integration of humans and agents, a modern and feature-rich programming language such as Python, which is the leader in most fields today and especially in AI [6], and a set of development mechanisms that facilitate the implementation of MAS applications. With the use of SPADE, the agents are programmed as containers or classes, which are filled by *behaviours* so that the agent can react to changes in its environment.

In addition to the core of the software architecture, the Python development environment, other peripherals are added to the overall development environment. This includes, on the one hand, the setup of a communication server, which supports the exchange of messages via XMPP. In this case, an XMPP server is set up using *Prosŏdy IM* and the Ubuntu operating system, which enables message exchange over the Internet. To enable the integration of existing data, e.g., tables from available tools or machines, interfaces to *Microsoft Excel*, which is a well-known tool for designers, and relational databases, e.g., *MySQL*, are provided. A special feature of this software architecture is the interface to the CAD program *Autodesk Inventor Professional 2023* so that the CAD models can be read out automatically and adapted after the evaluation has been carried out. The CAD program Inventor is used, since this system has in principle all the methods and tools for a knowledge-based CAD [22] and a large part of the functions can be accessed via the API.

3.2. Programming BDI-Agents

In terms of agent standardization, the BDI architecture is used to provide a framework that can be applied to a variety of domains. In this context, for example, the representation of experts from the processes downstream of product development, e.g., manufacturing, assembly, or maintenance, can be regarded as domains, which in turn pursue different goals regarding the product. Here, the classical computational model of a BDI agent is event-driven, i.e., a BDI agent responds to events, e.g., changes in the environment or in its beliefs, then adopts a plan as one of its intentions, and finally decides on its next action from the set of active intentions. In addition, an agent provides a set of services that can be requested by another agent. By defining the agents as classes, an encapsulation of the agents'

functions occurs so that they can be easily copied or called by other programs. For example, access to the 3D model can be made available to all agents so that they can make their own assessments based on it.

To store the beliefs, goals, and plans within the agent, they can be programmed using the programming language *AgentSpeak(L)* (ASL) [23]. The basic idea of programming with AgentSpeak is to define problem-solving knowledge in terms of plans to characterize responses to events. In this way, agents can be created that not only systematically attempt to achieve goals using the knowledge provided to them, but also respond to their changing environment [5]. A special feature of AgentSpeak is the querying and setting of beliefs, as this allows the *belief base* of the agent to be changed. About the knowledge bases used, this approach represents a technology-open solution, since the agents can be equipped with specific interfaces without this having any influence on the rest of the MAS. For this purpose, only the underlying information model has to be evaluated and transferred to the agent so that it can also react to dynamic state changes. For this purpose, the permanent storage in blackboards, which are updated by so-called event triggers, would be suitable. Furthermore, by extending SPADE with the SPADE-BDI plugin, which has an interpreter to read and execute a file written in AgentSpeak, an implementation for BDI agents is possible [19]. This allows mixing procedural, object-oriented and logical programming [24].

3.3. Graph-based Feature Recognition

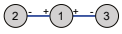
Automatic feature recognition methods are based on the idea of recognizing significant identification characteristics of a feature from the individual elements of the CAD geometry model using software support. Feature recognition views the model from different perspectives and has to read and adopt the information accordingly [8] to interpret a set of manufacturing features from a solid model. In feature recognition mechanisms, features are first extracted based on certain separation rules for further analysis. Numerous techniques, such as rule-based, graph-based, volume decomposition, and artificial neural network-based systems, have been developed in the literature on this topic [25]. Among all these techniques, the graph-based approach is considered the most popular method [25], which was first described by Joshi and Chang [14], as the concept of an attributed adjacency graph (AAG).

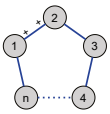
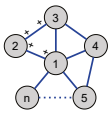
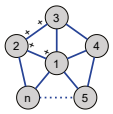
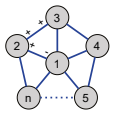
In graph-based representation schemes, such as the AAG, nodes and edges usually represent the faces and edges from the B-Rep (Boundary Representation) structure associated with some attributes, such as convexity and concavity of edges, type of face, perpendicularity, parallelism, or tangency of edges and faces [25]. An AAG can be defined as a graph G=(N,A,T), where N is the set of nodes, A is the set of edges, and T is the set of attributes to edges in A [29]. Zhu et al. [29] present the translation of a 3D model into an AAG, the partitioning into subgraphs, and the classification of these into specific features, using the number of nodes and faces, as well as the relationships of these to each other. Furthermore, they present a method to reunite separated features and to assign detected features to manufacturing processes and machining operations.

The described graph-based approach is adopted in this paper and extended for the analysis of 2.5D milled parts. Therefore, the B-Rep structure of the 3D model is also read out. After the AAG is created for the CAD model, a graph-based feature recognition mechanism is used to search for possible manufacturing features. For this purpose, the AAG is first split at the convex edges. Table 1 shows the different types of features for machining by 2.5D milling process the system is able to detect. Here, on the one hand, the number of faces, the number of edges and their connection type, and the convexity (indicated by a -1) and concavity (indicated by a 1) of the edges play a crucial role in describing the features. Based on the AAG subgraphs, the respective graph type is determined with the number of nodes (faces) and the combination of inner-/outer-loops. This is used for the initial classification of the subgraphs and is then assigned to a feature through further restrictions. If this does not identify a unique feature, the edge types and the connections of faces and edges are also included. As an example, the subgraph with a single cylindrical surface can be directly assigned to the graph type A. However, the cylindrical surface can belong to both a fillet and a through hole. In order to exclude a feature, the edge type is used, which is also indistinguishable, so the connection with adjacent surfaces must be checked. If there is a tangentially connected surface among these surfaces, the subgraph can be clearly assigned to the feature fillet.

The graph-based approach enables robust feature recognition because the individual features in the AAG can be determined independently of each other. Here, feature analysis is independent of the model tree in the CAD program and the type of modeling, allowing different types of filtering of the model structure. Furthermore, negotiation between multiple agents during feature recognition is possible. In addition, the developed system allows the use of a STEP file as an input file, which is the most common form of data exchange and is supported by many CAD systems. This also

Table 1: Graph and feature types for milled parts.

	Graphtyp	Features	Further restrictions		
			Facetype	Edgetype	Connection
A		Face	kPlaneSurface	-	-
		Chamfer	kPlaneSurface	-	Axis -> X,Y,Z ≠ 1
		Fillet	kCylinderSurface	kCircleCurve	Has tangentially connected Faces
		Through Hole	kCylinderSurface	kCircleCurve	-
		Through ID	kConeSurface	kCircleCurve	-
B		Step	kPlaneSurface + kPlaneSurface	-	-
		Blind Hole	kCylinderSurface + kConeSurface	-	-
		Blind ID	kCylinderSurface + kPlaneSurface	-	-
		2 Faces Closed Pocket	-	-	-
C		Cylindrical Step	kCylinderSurface + kPlaneSurface	-	-
		2 Faces Boss	kPlaneSurface + ?	-	-
D		Turning Slot	1x kCylinderSurface	-	-
		Slot	-	-	-

	E	F	G	H	I
Graphtyp					
Features	Slot	Through Pocket	Open Pocket	Closed Pocket	Boss
Connections	-	All Normals in one Plane	Open Edge(-1)	Just OuterLoops	Has InnerLoops

promotes the separation of the approach to modeling and the approach to manufacturing the part. However, contextual information, e.g., about the procedural approach to modeling, is also lost in the process, in which information about holes, for example, must be additionally captured. To compensate for this loss, the approach presented here enables interaction with the designers via a chat client, so that further context information can be requested, e.g., for the function of features.

4. Application and Validation of the Multi-Agent System with Milling Designs

For the application and validation of the MAS with BDI agents and graph-based feature recognition, 3D models of milling designs are evaluated for manufacturability using a digital design review. The models are checked against design guidelines and available tools as well as machines. This is intended to minimize errors in downstream production steps by checking the manufacturability and possible optimization of the component shape. To optimally integrate the system into the designers' workflow and ensure support during the design process, the structured procedure shown in Fig. 1 is implemented using MAS and described in the following.

4.1. Manufacturability Evaluation Process with the Multi-Agent System

First, easy access to the system is provided using a GUI in which the design to be checked is selected and further data on the tools and machines is available. In addition, a contact address (JID) of the user is transferred, via which the user can be reached by the system in case of questions and uncertainties. With this information, the MAS is started so that a manager agent, which monitors the entire system and coordinates the other agents, begins to work. It also collects the processed information and changes, and makes them available for output. The manager agent first starts the directory agent, which functions as an address book and gives an overview of the agents that have been created. All agents that are started must first register with their JID and the assigned role in the MAS. Once the registration is complete, the agents report to the manager agent as ready to work. If an agent needs a contact with a certain role during the process, the directory agent can be contacted, which returns a corresponding JID.

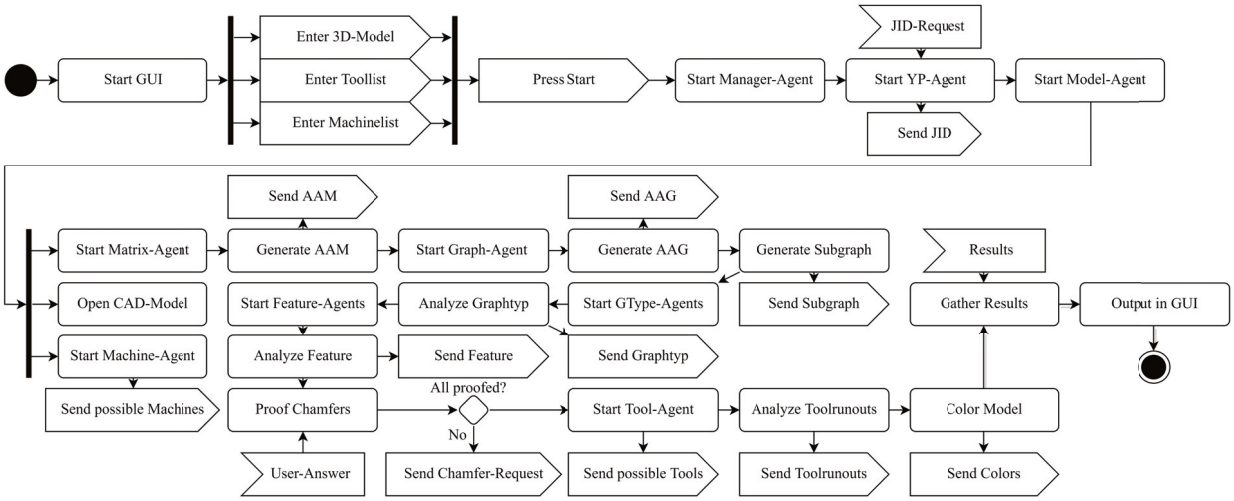


Fig. 1: Activity diagram of the agents for the evaluation of milling designs.

When the directory agent is successfully started, the manager agent calls the model agent. This agent is responsible for reading out and manipulating the CAD model in Inventor. From the information that has been read out, the agents themselves have to generate an image of the model and adjust it if necessary [8]. Reading the model is made possible with the Python library *PyWin32*, which provides an interface to the *application programming interface* (API) and thus to the *API Object Model* stored in Inventor. These interfaces are used to create dictionaries that describe the model in its basic structure. They contain information about properties, points, edges, surfaces, and loops of the model and are supplemented by the recognized features during processing. These dictionaries are passed to the managing agent. The manager agent forwards the dictionaries to the started machine agent and the B-Rep agent.

The B-Rep agent creates an *attributed adjacency matrix* (AAM) from the boundary representation of the model stored in the dictionaries. First, the edges are differentiated into concave, convex and straight edges. Furthermore, these are again subdivided into different curve types and assigned a unique value. If the edge belongs to an outer loop, it is given a negative sign. In the next step, the edges are transferred to an AAM in relation to the associated surfaces. This is in turn passed on to the manager agent. At the same time, the machine agent can compare the machine list with the dimensions contained in the dictionaries and compile the possible machines for processing. After receiving the AAM, the manager agent starts the graph agent, which creates the AAG from the AAM using the Python package *NetworkX*.

Subsequently, this is split at the convex boundary edges, leaving smaller clusters, the subgraphs (Fig. 2). These are forwarded individually via the manager agent to a corresponding number of started graph type agents. The graph type agents consequently assign the subgraph to a graph type. The manager agent is informed of the result and starts a feature agent for each subgraph, which uses the graph type and the information from the dictionaries to assign the subgraph to a specific feature. The feature is passed to the manager, which enters the first entries of the result dictionary with the information.

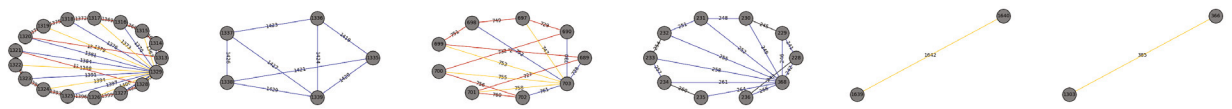


Fig. 2: Subgraphs of the application example of a milled part.

If the features contain chamfers, the model agent checks the necessity of these chamfers in consultation with the users. If necessary, the phases that are not required are removed from the model and noted accordingly in the dictionary. The query is made via the XMPP and the transferred JID. The users can answer the questions via messenger, such as Pidgin. The individual features are now analyzed further. One test relates to the necessity of the included

chamfers. If a convex bevel is detected, the user is asked via the messenger whether it is necessary, as in Fig. 3. If this is negated, the system removes the chamfer directly from the model with the corresponding note in the result dictionary. Since the query also refers to the respective transient key of the chamfer surface and this is not visible from the 3D view, a macro is embedded in Inventor with which the respective surface can be marked. The macro accesses the model and selects the surface that matches the transient key in the API Object Model with a *select command* from Inventor. When the last question is answered, the updated dictionaries are passed to the manager.

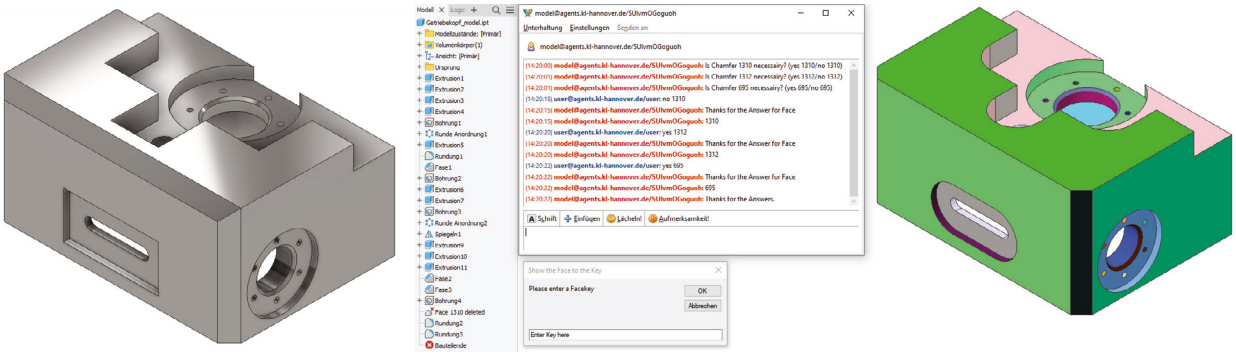


Fig. 3: Gear housing as an application example: Unprocessed Part (left); Revised Part (right).

The manager agent starts a tool agent for each remaining feature, which compiles a possible tool selection from the information and the tool list. At the same time, a recommendation of a tool is made based on the possible cutting volume to shorten the cutting process. For this, the system first needs information such as the machining direction and, if necessary, the base surfaces (main face). These are determined by comparing the normal vectors of the surfaces and the respective extension of the feature in the axial direction. After returning the tool list, the model agent checks the 3D model for missing tool run-outs. If this is missing, it is automatically added to the model with consideration of the tool used. For this purpose, the edges in the tool direction are considered. The changes are noted in the dictionary and transferred to the manager. Next, colors are assigned to the individual features by the model agent, stored in the dictionary and the respective features are colored in the model. This makes it easier for the user to view the results. Finally, the manager compiles all the results and saves them in a folder. This includes an Excel table with the result dictionary and visualization of the assigned colors. The results are also output via the GUI. The results obtained during the process are constantly added to a result dictionary. A section of this dictionary about the subgraphs shown in Fig. 2 is shown in Table 2.

Table 2: Feature table of the application example of a milled part.

No.	Feature	Main face	Axis	Faces	Edges
1	Boss	1329	[X=0, Y=-1, Z=0]	[1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328]	[1359, 1360, 1361, 1364, 1363, 1367, 1366, 1370, 1369, 1373, 1372, 1376, 1375, 1379, 1378, 1381, 1384, 1383, 1385, 1388, 1387, 1391, 1390, 1394, 1393, 1397, 1396, 1400, 1399, 1403, 1402, 1405]
2	Closed Pocket	1339	[X=-1, Y=0, Z=0]	[1335, 1336, 1337, 1338, 1832, 1834]	[1419, 1420, 1421, 1423, 1424, 1427, 1426, 1429]
3	Closed Pocket	703	[X=0, Y=-1, Z=0]	[689, 690, 697, 698, 699, 700, 701, 702]	[728, 724, 727, 730, 729, 732, 747, 749, 752, 751, 753, 755, 756, 758, 760, 761]
4	Open Pocket	368	[X=0, Y=1, Z=0]	[228, 229, 230, 231, 232, 233, 234, 235, 236, 1848, 1850, 1851]	[242, 243, 241, 246, 245, 249, 248, 252, 251, 255, 254, 258, 257, 261, 260, 264, 266]
5	Blind Inside Diameter	1639	[X=-1, Y=0, Z=0]	[1640]	[1642]
6	Blind Inside Diameter	1303	[X=0, Y=1, Z=0]	[366]	[385]
...	...	...	...	...	...

4.2. Validation of the Multi-Agent System

The previous sections have described the structure, procedure, and functions of the MAS. In the following, the MAS is validated on four milling designs and the results are compared based on evaluation criteria. The milling designs differ in their machining scope, the number of design elements and the complexity of the geometries. Furthermore, all components can in principle be produced with a 2.5D milling process and are based on realistic milled components, so that the applicability of the system for practical use can also be demonstrated. The designs used as application examples are shown in Fig. 4. The upper images show the input state of the models, and the lower images show the solutions generated by the system.

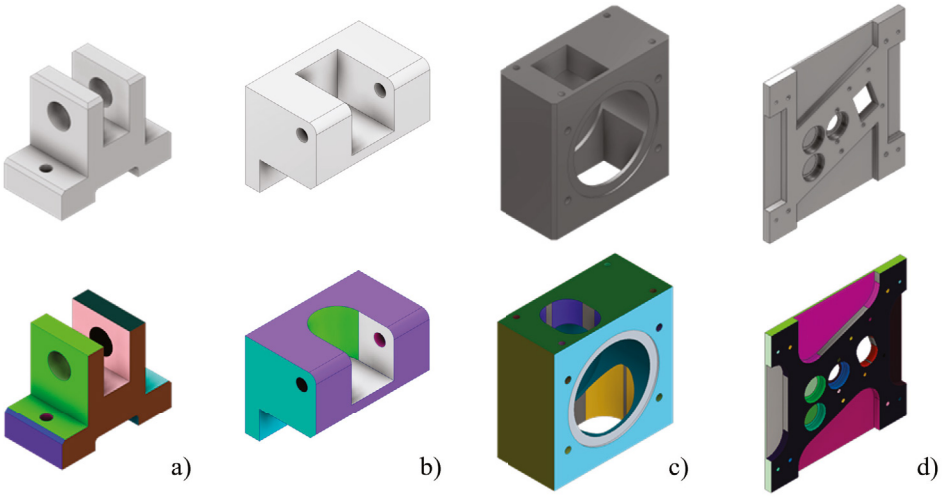


Fig. 4: Application Examples: (a) Bearing Block; (b) Hinge; (c) Cellular wheel housing; (d) Cover Plate.

The digital design review of the various milled components shows that the system correctly detected all errors in the components. In addition, the system automatically detected the external chamfers and contacted the user for this purpose. Furthermore, based on the stored tools, tool run-outs were automatically introduced if they were missing in the designs. In addition, the entire evaluation process was logged, and the results were output as a feature table.

To provide a comparison of the milled parts and their design evaluations, the milling designs are reviewed against each other based on the program flow, analysis, and synthesis. The criteria are user interaction, the number of agents used, and the time the system takes to design. User interaction refers to consulting with the user when chamfers are needed. The number of agents includes the fixed agents and the variable number that depends on the number of features. For the total runtime, only the pure computation time is included, excluding the user interaction, since the response time is highly dependent on the users and thus could distort the result. The analysis includes the number of detected features, the number of different feature types, the number of detected graph types A-C, as well as D-I, and the number of different tools. The graph types were divided into two sections, as graph types A-C have a simple structure with up to two faces and one edge, while graph types D-I have no fixed upper limit of faces involved. The synthesis summarizes the operations that require modification of the model. One is whether an unneeded chamfer is successfully removed after consultation with the user, and the other is the addition of radii due to a missing tool run-out. The results of this investigation are summarized in Table 3.

The comparison shows that the number and variety of features tends to be dependent on the size of the components, but this does not apply to the total running time. The complexity of the individual features is the decisive factor, as shown by the hinge and the cover plate. This is due to the more complex reading of the model. The user interaction turns out to be simple and helpful. The system can reach the user regardless of location and receive quick feedback in case of ambiguities and questions. In addition, the system finds missing tool discontinuations that have a strong impact on production. These are automatically integrated into the model at the appropriate point, optimized for production,

Table 3: Comparison of various milling parts.

	Program sequence			Analysis					Synthesis	
	User Interaction	Number of agents	Total runtime [s]	Number of features	Featuretypes	Graphtypes A-C	Graphtypes D-I	Number of tools	Missing tool run-out	Optical chamfers
Hinge	yes	49	71	14	7	12	2	5	yes	no
Bearing Block	yes	79	73	24	6	22	2	4	no	yes
Cellular Wheel Housing	yes	106	217	33	7	31	2	8	yes	yes
Cover Plate	no	117	300	37	5	32	5	10	yes	no

depending on the tool and the direction. The results are then presented to the user and archived collectively so that changes and information can be traced.

5. Discussion, Conclusion and Further Research

The presented MAS can support a digital design review with multi-agent systems regarding the manufacturability of CAD models. In addition, to address the lack of scalability and robustness in existing approaches, this paper supported the use of runtime-generated BDI agents and graph-based feature recognition. Furthermore, the system can be used to simplify and digitize the approval process by allowing the reviewer to use the feature table for review. Compared to manual review, this can save time as the system automatically reads the machining steps, checks the manufacturability of the tools, and automatically adjusts the CAD model. A check of the design regarding the available machine tools is possible by analyzing the part with different stored tool lists, so that an automatic adjustment of the geometry, in case of a short-term change to another machine tool, is possible without much effort. The combination of a knowledge-based system and a MAS for the analysis and synthesis of a design forms the difference from the related literature, which only suggests changes or offers an adaptation of the CAD model in closed solution spaces.

Nevertheless, the system has some limitations, such as the lack of recognition of interacting and interrelated features, since here, in addition to the tools, the machining strategy also has an influence. The interaction with the designers takes place on the basis of the questions defined in advance so that the additionally required context information must already be taken into account during programming. For this purpose, either yes-no questions or questions with a predefined answer selection are used. Nevertheless, the interface enables additional information to be entered into the system promptly and from any location.

The results shown represent the potential for design aids. Thereby, the extension of the specific domain knowledge represents a promising approach to include further manufacturing processes, such as further machining processes with the use of further axes, casting processes, and additive manufacturing. Since the design processes under consideration are very knowledge-intensive, the investigation of learning algorithms seems promising regarding intelligent agents, in which automated classification of analysis steps is performed using data-based AI algorithms.

References

[1] Alzetta, F., Giorgini, P., Najjar, A., Schumacher, M.I., Calvaresi, D., 2020. In-Time Explainability in Multi-Agent Systems: Challenges, Opportunities, and Roadmap. Springer. volume 12175 LNAI. pp. 39–53. doi:10.1007/978-3-030-51924-7_3.

[2] Bellifemine, F., Caire, G., Greenwood, D., 2007. Developing Multi-Agent Systems with JADE. 1 ed., Wiley. doi:10.1002/9780470058411.

[3] Bergenti, F., Caire, G., Monica, S., Poggi, A., 2020. The first twenty years of agent-based software development with jade. Autonomous Agents and Multi-Agent Systems 34, 36. doi:10.1007/s10458-020-09460-z.

[4] Berriche, F.Z., Zeddini, B., Kadima, H., Riviere, A., 2016. Closed-Loop Product Lifecycle Management Based on a Multi-agent System for Decision Making in Collaborative Design. John wiley & sons. pp. 540–551. doi:10.1007/978-3-319-47650-6_43.

[5] Bordini, R.H., Hübner, J.F., Wooldridge, M., 2007. Programming Multi-Agent Systems in AgentSpeak usingJason. John Wiley & Sons, Ltd. doi:10.1002/9780470061848.
[6] Cass, S., 2020. The top programming languages: Our latest rankings put python on top-again - [careers]. IEEE Spectrum 57, 22–22. doi:10.1109/MSPEC.2020.9150550.
[7] Dong, M., Mao, X., Yin, J., Chang, Z., Qi, Z., 2009. SADE: A Development Environment for Adaptive Multi-Agent Systems. pp. 516–524. doi:10.1007/978-3-642-11161-7_37.
[8] Fougères, A.J., Ostrosi, E., 2018. Corrigendum to "intelligent agents for feature modelling in computer aided design" [j. comput. des. eng. (2018) 19–40]. Journal of Computational Design and Engineering 5, 379–379. doi:10.1016/j.jcde.2018.05.005.
[9] Gembarski, P.C., 2020. On the Conception of a Multi-agent Analysis and Optimization Tool for Mechanical Engineering Parts. John Wiley & Sons. volume 186. pp. 93–102. doi:10.1007/978-981-15-5764-4_9.
[10] Georgeff, M.P., Rao, A., 1992. An abstract architecture for rational agents, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA. pp. 439–449.
[11] Jennings, N.R., Wooldridge, M., 1995. Applying agent technology. Applied Artificial Intelligence 9, 357–369. doi:10.1080/08839519508945480.
[12] Jia, H., Ong, S., Fuh, J., Zhang, Y., Nee, A., 2004. An adaptive and upgradable agent-based system for coordinated product development and manufacture. Robotics and Computer-Integrated Manufacturing 20, 79–90. doi:10.1016/j.rcim.2003.08.001.
[13] Jian, G., Gao, J., Wang, Y., 2010. A multi-agent based knowledge search framework to support the product development process. International Journal of Computer Integrated Manufacturing 23, 237–247. doi:10.1080/09511920903529222.
[14] Joshi, S., Chang, T., 1988. Graph-based heuristics for recognition of machined features from a 3d solid model. Computer-Aided Design 20, 58–66. doi:10.1016/0010-4485(88)90050-4.
[15] Liu, Q., Cui, X., Hu, X., 2008. An Agent-Based Intelligent CAD Platform for Collaborative Design. volume 15. pp. 501–508. doi:10.1007/978-3-540-85930-7_64.
[16] Mahesh, M., Ong, S.K., Nee, A.Y.C., 2007. A web-based multi-agent system for distributed digital manufacturing. International Journal of Computer Integrated Manufacturing 20, 11–27. doi:10.1080/09511920600710927.
[17] Medani, O., Ratchev, S., 2006. A step ap224 agent-based early manufacturability assessment environment using xml. The International Journal of Advanced Manufacturing Technology 27, 854–864. doi:10.1007/s00170-004-2279-0.
[18] Ortiz-Arroyo, D., Larsen, H.L., 2004. A multi-agent system framework for collaborative decision making under uncertainty, IPSI. Null ; Conference date: 19-05-2010.
[19] Palanca, J., Terrasa, A., Julian, V., Carrascosa, C., 2020. Spade 3: Supporting the new generation of multi-agent systems. IEEE Access 8, 182537–182549. doi:10.1109/ACCESS.2020.3027357.
[20] Plappert, S., Becker, C., Gembarski, P.C., Lachmayer, R., 2022a. Feasibility evaluation of milling designs using multi-agent systems. Proceedings of the Design Society 2, 763–772. doi:10.1017/pds.2022.78.
[21] Plappert, S., Gembarski, P.C., Lachmayer, R., 2021. Multi-Agent Systems in Mechanical Engineering: A Review. Springer Singapore. volume 241. pp. 193–203. doi:10.1007/978-981-16-2994-5_16.
[22] Plappert, S., Gembarski, P.C., Lachmayer, R., 2022b. Development of a knowledge-based and collaborative engineering design agent. Procedia Computer Science 207, 946–955. doi:10.1016/j.procs.2022.09.150.
[23] Rao, A.S., 1996. AgentSpeak(L): BDI agents speak out in a logical computable language. pp. 42–55. doi:10.1007/BFb0031845.
[24] Rincon, J., Julian, V., Carrascosa, C., 2022. Flamas: Federated learning based on a spade mas. Applied Sciences 12, 3701. doi:10.3390/app12073701.
[25] Shi, Y., Zhang, Y., Xia, K., Harik, R., 2020. A critical review of feature recognition techniques. Computer-Aided Design and Applications 17, 861–899. doi:10.14733/cadaps.2020.861-899.
[26] Sun, J., Zhang, Y.F., Nee, A.Y.C., 2001. A distributed multi-agent environment for product design and manufacturing planning. International Journal of Production Research 39, 625–645. doi:10.1080/00207540010004340.
[27] Wang, W., Li, Y., Li, H., Liu, C., 2012. An agent-based collaborative design framework for feature-based design of aircraft structural parts. International Journal of Computer Integrated Manufacturing 25, 888–900. doi:10.1080/0951192X.2011.597779.
[28] Weyns, D., 2010. Architecture-Based Design of Multi-Agent Systems. 1 ed., Springer Berlin Heidelberg. doi:10.1007/978-3-642-01064-4.
[29] Zhu, J., Kato, M., Tanaka, T., Yoshioka, H., Saito, Y., 2015. Graph based automatic process planning system for multi-tasking machine. Journal of Advanced Mechanical Design, Systems, and Manufacturing 9, JAMDSM0034–JAMDSM0034. doi:10.1299/jamdsm.2015jamdsm0034.