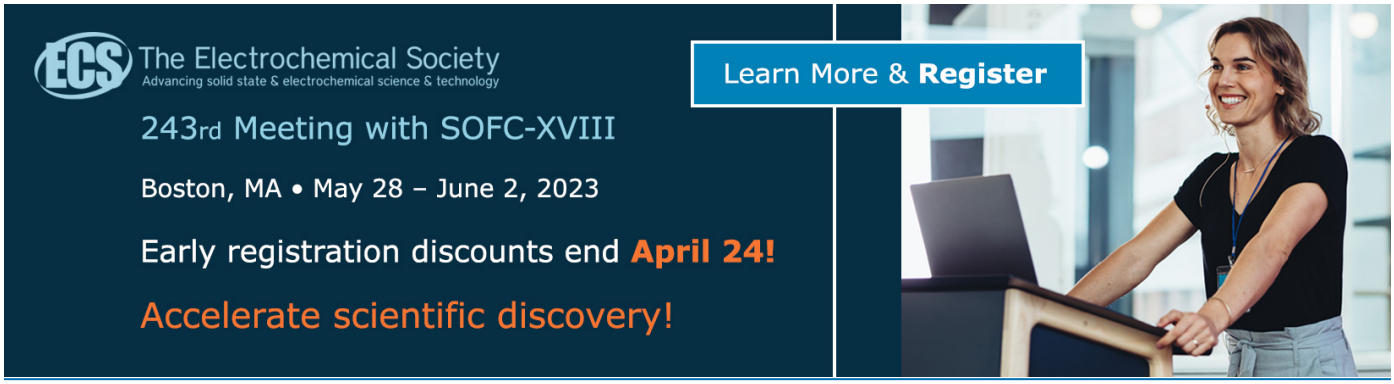
An Intelligent Agent-Controlled and Robot-Based Disassembly Assistant



IOP Conf. Series: Materials Science and Engineering **235** (2017) 012005 doi:10.1088/1757-899X/235/1/012005

An Intelligent Agent-Controlled and Robot-Based Disassembly Assistant

Jan Jungbluth[1,2,3,4], Wolfgang Gerke[2,5] and Peter Plapper[3,6]

[1] Technology Research Group, SEW EURODRIVE, Bruchsal, Germany
[2] Robotic and Control Group, Trier University of Applied Sciences (TUAS), Environmental Campus Birkenfeld, Birkenfeld, Germany
[3] Faculty of Science, Technology and Communication, University of Luxembourg, Luxembourg, Luxembourg

{[4]jan.jungbluth | [5]w.gerke}@umwelt-campus.de, [6]peter.plapper@uni.lu

Abstract. One key for successful and fluent human-robot-collaboration in disassembly processes is equipping the robot system with higher autonomy and intelligence. In this paper, we present an informed software agent that controls the robot behavior to form an intelligent robot assistant for disassembly purposes. While the disassembly process first depends on the product structure, we inform the agent using a generic approach through product models. The product model is then transformed to a directed graph and used to build, share and define a coarse disassembly plan. To refine the workflow, we formulate "the problem of loosening a connection and the distribution of the work" as a search problem. The created detailed plan consists of a sequence of actions that are used to call, parametrize and execute robot programs for the fulfillment of the assistance. The aim of this research is to equip robot systems with knowledge and skills to allow them to be autonomous in the performance of their assistance to finally improve the ergonomics of disassembly workstations.

1. Introduction

Disassembling is involved in many processes, for example remanufacturing, corrective maintenance, proper disposal and manufacturing. However, fully automated disassembly lines are, compared to product assembly lines, rare. One reason is that disassembling at the end of a product lifetime is much harder to automatize than assembling. In disassembly we have to cope with fouling, wear, damaged or only absent parts. Furthermore, we have to deal with product manipulations, such as individual extensions or improvised fixes, which are not obviously visible. It is the unpredictable condition of a product that prohibits further automation. Even if better sensor technology could identify inappropriate product conditions, it would be impossible, or highly expensive, to treat all possibilities in a fully automated manner. Other challenges are the small lot sizes or individualized products (lot size 1). Today's fully automated processes are not flexible enough for the treatment of different products or variants. In corrective maintenance, it is a normal case to have an unsteady, unpredictable flow of different products. Dismantling processes in corrective maintenance may also have different target stages, depending on what part has to be replaced. Furthermore, there is, especially in central waste recycling plants, a lack of information about the product structure. For example, types of materials in the product. These are reasons why disassembly workplaces stayed unautomated, thus resulting in a

bad situation for the workforce, who remain exposed to health problems due to heavy workload. In addition, the economic fitness of disassembly processes is highly reduced by the substantial amount of manual labor in the destructive disassembly process. Consequently, important concepts for environmental protection, such as remanufacturing, cannot spread wider into industry.

A solution for this problem would be intelligent robot-based assistants, which would be a compromise between automation and manual labor with great advantages in the disassembly domain. One advantage is that only humans have the cognitive abilities to identify and handle the aforementioned unexpected situations. Thus, humans are able to ensure the overall success of the process by contributing their awareness to adapting the process to a situation [1]. The robot instead can provide assistance with power and endurance over the complete disassembly process and thereby improve the ergonomics of the workplace. Therefore, multi-skilled robots may take over automatable tasks, such as unscrewing, or support the execution of tasks, e.g., handling of heavy parts. Especially over the 10 last years, with the development of lightweight and force-sensitive robots, new robot skills have extended the robots' fields of application. Those new skills provide realizable "assistance opportunities" in disassembly procedures, and some of them could be integrated into one robotic system. However, once integrated, the problem arises of controlling the multi-skilled robot and its behavior to assist the user in a situation- and goal-oriented manner. Even providing the system with needed precise positions information is big issue [2]. Manually programming the whole process would not be meaningful [3], especially if we consider many different products. Only intelligent systems with higher autonomy could support and interact fluently with humans in such a dynamic environment [4, 5]. Therefore, the system needs to identify the disassembly process and the work contend by itself. Then, the robot must know how to assist the user in a certain task; it also needs necessary information and skills to perform this task. To meet these challenges, we take the approach of an informed software agent, the architecture of which we present and explain in Section 2. In Section 3, we explain the kind of information that is provided to the agent and in section 4 we clarify how we decompose the complex disassembly problem into individual subproblems. Coarse disassembly planning is used to identify the task sequence that strips down the assembly. In Section 5, we present our approach to assigning the work content of each task to the participants through a planning search in the possible situation space. Explained in Section 6 is how we execute the determined detailed plan by invoking and parametrizing robot programs to finally control the robot assistance behavior.

2. The Agent-Based Robot Control Architecture

Today's industrial robot controllers are not suitable for human -robot interaction in complex environments such as disassembly workstations [1]. In such environments, the objective and the boundary condition change from task to task and require some planning, higher skills, and knowledge to succeed efficiently. Our approach to empower industrial robots to this application field is to overlay the robot controller with an intelligent agent-based control system, which does high-level planning, defines and controls the robot behavior, e.g., the type of assistance, and guarantees the necessary information to the underlying robot controller. The developed software agent, which we present in this section, consists of different software modules (see figure 1). In the agent's "Knowledge Base" module, we store a symbolic representation of the product structure, which we call the product model. The model contains information about the parts and how the parts are connected with each other in the assembly. Therefore, we model different classes of parts and connections. We also model the actions that the actors (the user and the robot) provide to the overall system, to act in the environment or on the product structure. We explain all the stored information in further detail at the time of its use, following the systems information flow and processing. In the "Coarse Planning" module we create, depending on the disassembly objective and the product model, a coarse plan that consists of a sequence of disassembly steps. Each disassembly step consists of one or more tasks. A task defines an independent subproblem, which is the removal of a specified connection and the referenced parts. Moreover, we use the coarse planning to assign the means of production, such as tools, robot effectors, and carriage cases, to a task. This is done through the "Stock Manager" module. Depending on the

connection type, different assistance behaviors of the robot are available and selectable in a task by the user. Furthermore, the user can rearrange the task order and manipulate parts' condition states, for example, to mark a damaged part, over the human-machine-interface (HMI) module to adapt the process to the circumstances and his/her will. The coarse plan is then processed task by task by the "Detail Planning" module, in which we first create a discrete state space representation of the current and the target situations. A search through the possible state space, which considers the actions provided by the user and the robot, leads to a set of possible action sequences from which the fittest sequence is selected for execution. The selected action sequence is then performed, monitored and synchronized by the "Control" module through advising the user over the "HMI" module or invoking and parameterizing programs on the underlying robot controller.

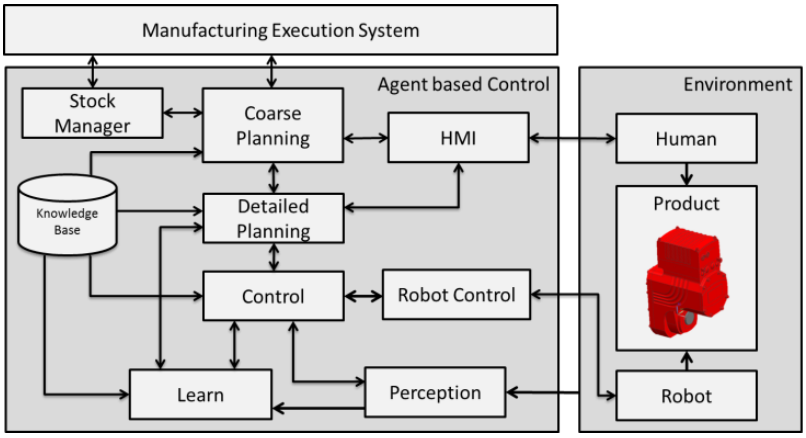


Figure 1. The agent's architecture and the information flow between the different software modules.

Currently under construction is the "Perception" module. In the first place, it is foreseen to track the human hands to synchronize the process without action-executed confirmation-button pressing. We use the "Perception" module also to recognize and interpret gestures and voice commands to alter the robot behavior in the execution phase. The "Learn" module is considered in the agent architecture to adapt the assistance behavior to an individual user. A possible purpose of machine learning is to predict the user's desire for an assistance behavior by comparing the current situation with similar situations from recorded older interactions. Also, the agent could improve the coarse planning, particularly the ordering of parallel executable tasks, by learning from manually adapted plans. Furthermore, recording and analyzing such disassembly processes could produce valuable data and lead to deeper insights into the disassembly process. Next, we explain the system in more detail, starting in the knowledge base and considering the product model.

3. The Product Model

A technical product is an assembly of parts that are linked together through connections. The step-by-step removal of these connections is the process of disassembling. The sequence of removing the connections is not arbitrary but partially defined through the product structure. Finding this disassembly sequence is known as disassembly planning, which is the topic of Section 4. So to generate the disassembly sequence later, we first have to present the structure in an appropriate and machine readable manner. Disassembly planning is an ongoing research field which also gains interest, through the assembly-by-disassembly approach, from the well-studied area of assembly planning. The majority of reviewed publications in this area use undirected and directed graphs as well as hypergraphs to represent the product structure [6–10]. Other approaches are based on petri nets [11], description logic and object-oriented models [12], and more recent approaches are based on ontologies [13–16]. Most works focus on sequence generation and thus lack in providing metadata for the human-robot interaction. For our usage, we have decided to develop an easy-to-use, light but comprehensive model to represent the product structure. After careful consideration, we have chosen the object-orientated approach based on its advantages of information encapsulation, easy implementation, and

extension. In addition, we can create other structures from the object-orientated model. We now explain the two fundamental classes of which our product model is composed.

3.1. Part Classes

In contrast to the most works which are based on [7], we do not tie functional parts to connections; instead, we clearly separate parts and connections into our fundamental classes. To distinguish between different classes of parts that have different needs of information in the sense of disassembly, we use a flat taxonomy-like hierarchy of classes (see figure 2). Each class inherited from the super class carries common and more specialized information in its attributes. A common attribute, for example, is the position and orientation of a part in reference to the main coordinate system of the assembly. More specialized information could be the thread length or the driver style of a screw class instance. An overview of the attributes we use in the super class is listed in the table of figure 2. Whenever a component indicates some special needs of information in the disassembly process, we can create a new class or extend a similar one. Chiefly, this is necessary for fasteners or connection techniques, but it could also be used to model fluids or gases. While the removal processes of a joint can have parameters that depend on the attributes of a part, we uncouple this information from the process. For example, the thread length and the thread pitch of a screw define the number of twists until the screw is loose. In this sense, it is a great advantage to use variable process parameters and to uncouple part-dependent facts from processes. Furthermore, we implemented state descriptions and qualifiers for each part class to represent each class's possible and current condition. This could be used to mark a part that it is damaged or missing. Beyond representing several parts, it is essential to model subassemblies. Mostly, it is not advisable to strip down a subassembly in another assembly. Typically, it is better to fractionalize subassemblies after their removal from the main assembly. So treating a subassembly as a special part class is beneficial in the manner that subassemblies gets completely removed from the main assembly and then further dismantled.

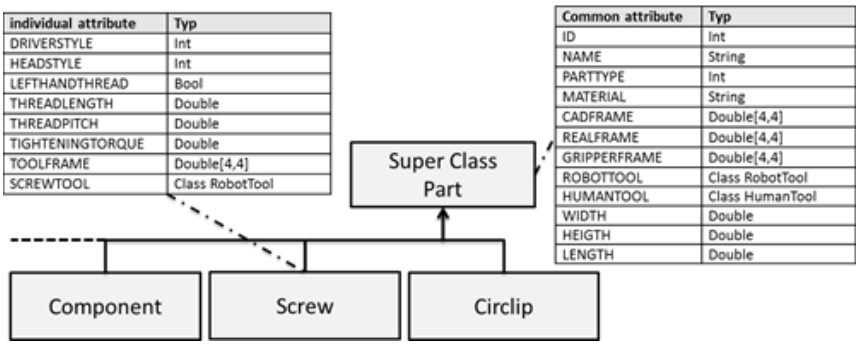


Figure 2. The inheritance hierarchy of part classes with details of common and individual attributes.

Also considered is information for the further treatment of the parts. In a remanufacturing process, we could be interested in adding information to the parts about a test and a rework process a part has to pass to get back into production. For appropriate recycling, the material is an important attribute. This approach to equip the robot with the necessary information looks plausible, but it also means a substantial amount of knowledge engineering. Some of this information is already stored in today's CAD systems in a form of standard part libraries and could be used as a source of information for the product data model. Some information needs to be consolidated and made available in a clearly structured manner that is readable for humans and machines.

3.2. Connection Classes

Similarly to the part classes, we build hierarchies with connection classes (see figure 3). Some common attributes are also illustrated in the table of figure 3. A connection defines what kind of liaison is between two or among more parts and could be of any type. From a face-to-face contact, a weld joint or a magnetic attraction, any kind of liaison can be designed and implemented. The real

power of the different connection classes is that they describe a formal process. If we can describe a certain state and objects in such a process, then we have a situation. Knowing the situation gives the assistant the ability to link commands dynamically with context information. In Section 5, we describe a formal process definition that makes use of a state representation and a set of actions, which could be performed by the acting agents (e.g., the user and the robot). We use this formal process description in detailed planning (Section 5) to find the best sequence of actions to remove the bound. As we see more clearly in Section 4.1, we can represent the product structure in a symbolical (machine-readable) manner by describing the parts and connections of an assembly.

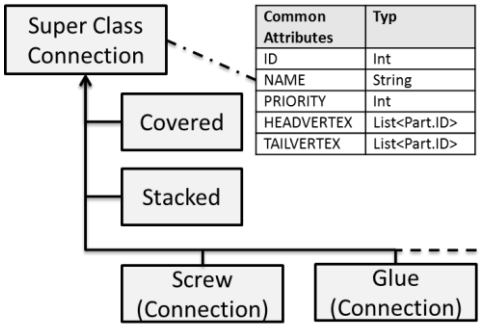


Figure 3. Inheritance hierarchy of the connection class. The ordering implied the disassembly priority of each class from low to high. Also illustrated are the common attributes of the super class.

For the efficient manual design of these connections, we integrated functions into the NX 10 CAD system, which also generates the instances of the part classes during the connection creation. In future work, the product model will be automatically created by the product configurator. The format used to store the part and connection instances for a product model will be based on the XML format and be transferred via RFID directly from the assembly or via web services to the software agent.

4. Coarse Disassembly Planning

Coarse disassembly planning reduces the amount of process definition done by the user to a significant level. While the disassembly task can vary among corrective maintenance, remanufacturing and recycling, we have to be able to create substructures of the complete disassembly for the exchange of wear and tear parts and target-oriented plans for the removal of valuable components. We also have to generate plans for the full dismantling of the product for recycling purposes. To achieve this, the disassembly task is communicated to the agent through the product model and the part to be removed. In this form, we are able to determine a plan for the removal of valuable or broken parts and, through selecting the root component of the assembly, the complete disassembly of a product. In the reviewed works [6–16] on assembly or disassembly planning, the main focus was on finding a sequence in which the parts could be added or removed to create a complete assembly or disassembly. It was not considered, to build substructures of assemblies, or to do further task and process planning. The used techniques vary depending on how the product structure was modeled. Most techniques used on graphs are mathematical and based on the adjacency matrix [6-9]. The inference is used by ontologies and description logic approaches [13–16], and rule-based systems use forward or backward chaining. Also found in the reviewed papers are applications of fuzzy logic and genetic algorithms [11]. We have decided to use the simple but powerful approach of topological sorting and do further task and process planning. We build the coarse plan in five steps. The first step is to create the product graph from the product model. In the second step, we create the minimal graph with the selected part as “root”. Then we create a sequence of disassembly steps and tasks through topological sorting in the third step. In the fourth step, we assign the means of production to each task. In the last step, the user has to define the type of assistance we want in a certain task. Each step is shortly discussed as follows:

4.1. Creating the Product Graph

We represented the product structure internally through a directed graph **G**(**V**, **E**), which consists of a set of vertices **V** and edges **E**. A vertex **v** represents a part instance, and an edge **e**($\mathbf{v}_i$, $\mathbf{v}_j$) represents a link between two parts with the direction from the head $\mathbf{v}_i$ to the tail-vertex $\mathbf{v}_j$.

A connection instance may have several edges depending on the connection type to better depict the interactions between parts. The graph is automatically created by the parts and connections described in the product model. For a better understanding of this topic, we will explain the graph creation and the part and connection models from the previous topic. For further explanation, kindly compare the sectional view of an electrical drive (in figure 4) and its product graph (in figure 5).

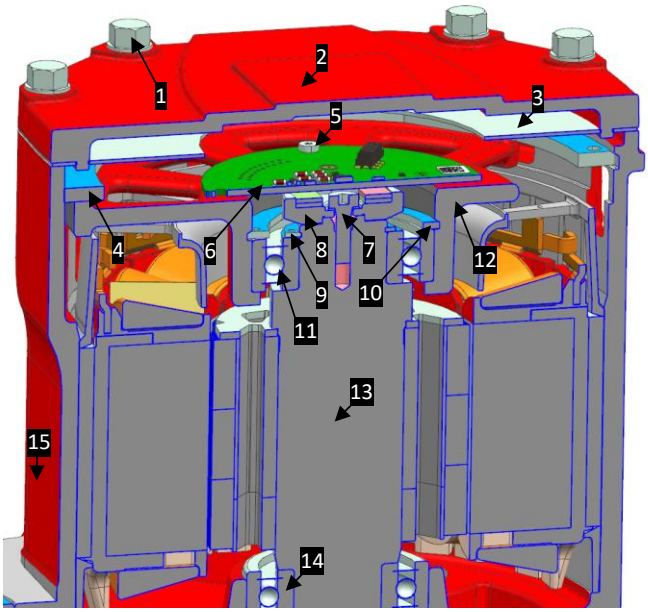


Figure 4. A cross-sectional view of an electrical. drive with numbered parts.

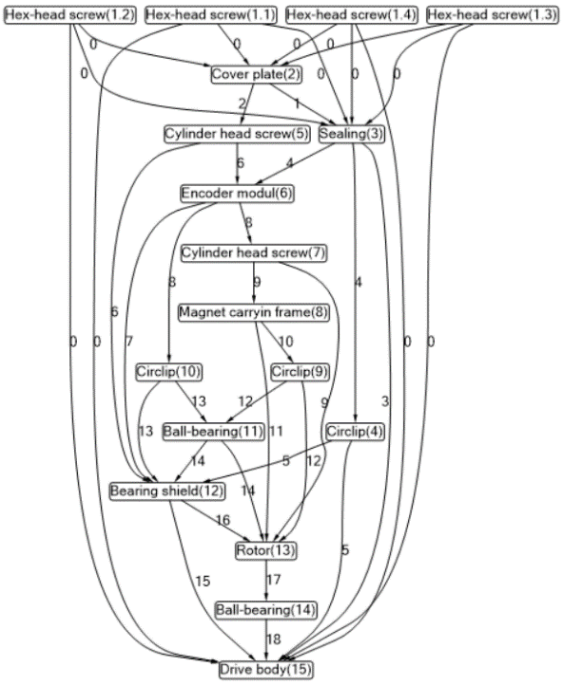


Figure 5. A product graph of the electrical drive of figure 4. The numbers on the edges are the IDs of connection instances.

We now explain three different connection types, two part types, and how we represent them in the graph. First, we consider the bolted joint of the four hex-head screws (Number 1.1–1.4 in figure 5) that link together the cover plate (2), the sealing (3) and the drive body (15). The four screws are all instances of the "screw" part class. The sealing, the cover plate, and the drive body are instances of the "component" part class.

The bolted joint is symbolically described by only one instance from the "screw" connection class because the screws are identical, and they link the same parts. In the screw connection instance, we define the screws as head- vertices and the other components as tail-vertices.

This connection is represented in the graph with sixteen edges, from each screw to any of the other three connected parts. The next connection we have to specify is the cover plate, which is lying on the sealing. We represent it by an instance of the "stacked" connection class. This connection is represented by one edge from the cover plate to the sealing. An instance of the "covered" connection is used to describe the situation that one part avoids access to another part without any physical interaction. This case applies to the cover plate and the cylinder head screw (5). The covered instance is also illustrated on the graph by one edge from the cover plate to the screw.

4.2. Disassembly Planning

The part to be disassembled defines the start node of a breadth-first algorithm, which adds all parent nodes and corresponding edges to a new subgraph. Through topological sorting of the subgraph, we then create the disassembly sequence. In each step of the sequence, we identify part nodes that have no ingoing edges and could be removed. Through the outgoing edges of each part node, we get related connection instances. Then, a new task is created with the connection and part instances thereby taking care that parts that belong to the same connection instance are grouped together in the same task (in the current disassembly step). Moreover, if a part is attached to more than one connection, then we sort the connections by their disassembly priority in the task. The disassembly priority describes which connection of a set of liaisons has to threaten first. For example, if part A lies on part B and between A and B is a glue, then it is better to warm up the parts until the glue loses its strength and then remove part A. Because multiple parts and connections could be removed independently in a disassembly step, we can have multiple parallel executable tasks. If all removable parts are handled in one disassembly step, we delete the nodes and edges and create a new step. This process repeats until the subgraph has no more nodes.

4.3. Assigning the Means of Production

Especially in remanufacturing processes, we have to separate unequal, and usable from unusable, parts for their further treatment. To do so, we have to store the parts in different transport boxes and keep track of which part is in which box. Assigning the right box to a part, and vice versa, is one duty of the local "stock manager" module. The stock manager uses the width, height and length attributes of a part to find a suitable box in the local stock. The box is then assigned to the task and part. If the box is not available in the local stock, then the stock manager produces an order. While we model all process-involved objects in our environment, we have a class that represents boxes. A box object has attributes, such as its storage dimensions or their position and orientation in reference to the robot work frame, and qualifiers to represent their current state, e.g., if there is a box available, or it is ordered. Furthermore, the local stock manager determines the needed human and robot tools by accessing the attributes of a part with respect to the connection type. For example, a screw instance has three different tools: the tool for human use, such as a screwdriver, a robot tool for manipulating the part, usually a gripper, and a robot screw tool to loosen the joint. Like the box object, each tool is represented by an instance of a tool or robot tool class and has attributes, such as their positions, and state qualifiers. Furthermore, we assign the actors, e.g., the user and the robot, to the tasks to have all process relevant entities together.

4.4. Plan Manipulation and Assistance Definition

The sequence generated through the agent is a partially ordered plan. So there are different possible processes and the created one does not have to match a user's expectation. Furthermore, there could be other reasons that the user wants to rearrange or manipulate the found order. To rearrange the process, the user can move tasks back and forth in the disassembly sequence and manipulate the process in other methods by modifying the state qualifiers of the parts and objects to adapt the process to the current product state. More input is needed regarding the fact that the user has to define how he expects to be supported. Therefore, the user has to choose for each connection one type of assistance from a set of recommended assistance behaviors. At the moment, this is done by right clicking on the link with the computer mouse, which is a very unnatural method to communicate and flow-break. Using multimodal communication, through gestures and voice commands, to define or change the form of assistance would be a great advantage and is currently under investigation. To reduce the amount of assignment in this kind, future work will have to investigate if machine learning or case-based reasoning is able to predict a user's wish for assistance with tolerable accuracy.

To summarize the steps to this point, we have used a decomposition strategy [17] to split the complex disassembly problem into smaller, independent subproblems. The subproblem we consider is the removal of a certain type of connection, which is a process that we can generally describe formally and solve efficiently because of its smaller size. Furthermore, we can think of the coarse planning in a

manner that it determines work that has to be done, without assigning it to the user or robot. The detailed planning in the next step solves the subproblem and assigns the work to the actors.

5. The Detailed Disassembly Planning

In the ideal case, the manual removal of a connection takes place through a fixed sequence of actions. In disassembly workstations, we regularly find cases in which the sequence has to be adapted, probably with other actions, to produce the wanted output. Collaborative work also means that actions might be executed by one or another agent, in a dynamic manner that produces many variants of the process. The actors can also provide assistance to each other, for example, the user can change a robot effector so that there are even more possibilities. With this combinatory problem in mind, we have decided that the process should not be explicitly defined through the use of fixed finite state machines or petri nets. Instead, we use a search approach to find a suitable sequence of actions to solve our subproblem. To represent our issue as a search problem, we have first to decide on a vocabulary of conditions, objects, and actions. Then, we have to encode actions from our domain and define a problem instance by defining the initial and the target conditions. We now explain how we generate the initial state and target state, represent the actions and what algorithms we used to search for a solution by using an easy example task. We consider the loosening of a "stacked" connection, in which part A is simply lying on part B.

5.1. The Task State Description

The state we want to represent depends on the connection type and the involved objects, such as part(s), tool(s), boxes and the acting agents. Since we have already collected all these objects in a task description, we can create the initial state by merging all objects' state descriptions. Some objects' state values are predefined, for example, the *PartState*, *PartPos*, and *ConnectionState* (see Table 1), but could be manipulated by the user. The value of the *BoxState* and *BoxPos* are defined by the stock manager, depending on whether, if the box is ordered, it is full or ready to use. The robot states are defined by sending queries to the robot controller. The users *HandPos* state will be tracked, and the *HandState* value is estimated by the perception module. These state values represent literals, which formally represent the condition of the task at the beginning, the end, and in between through a state vector. In a state, we also save the parent's ID, the costs and the action type that created the state. For the initial state, these attributes are zero. The end condition of the task state is partially defined, through the default values of the *ConnectionState* and *PartPos* (see Table 1). It is the users' choice to add other state dimensions and values to the end condition to define it more precisely.

Table 1. Observed objects in the task. Each object can have several dimensions and state values. The underlined values indicate predefined states of the initial task state. The literals in cursive are default values for the target state of the task.

Object type	State dimension	State values
Stacked Connection	Connection-State	isStacked, isDetached, *isRemoved*
Human	HandState	isEmpty, isNotEmpty
	HandPos	atUnknownPos, atPartPos, atBoxMagazinPos, atRobotGuidePos
Robot	RobotState	isUnknown, isIdle, isRunning, isGuided
	RobotPos	atUnknownPos, atHomePos, atPartPos, atBoxMagazinPos, atRobotToolMagazinPos,
RobotTool	RobotToolState	isUnknown, isOpen, isClosed
	RobotToolPos	atUnknownPos, atRobotFlange, atRobotToolMagazinPos
Part	PartState	isOK, isNOK,
	PartPos	atUnkownPos, atPartPos, atGripper, atHand, *atBox*
Box	BoxState	isAvailable, isOrdered, isFull
	BoxPos	atUnknownPos, atBoxMagazinPos

5.2. Describing actions

Methods to describe or analyze manual and robot-automated tasks are well known, for example, Method-Time-Measurement (MTM) and Robot-Time-And-Motion (RTM). For the new type of collaborative work, there is quite a lack of methods. An adapted Method-Time-Measurement as process logic for cognitive automated assembly was mentioned in [18], which did not mention a human-robot interaction. Other studies such as [19] only link the basic Methods-Time-Measurement (MTM-1) system to equivalent robot actions without any collaborative actions. We have decided to describe actions only with the motion and end-effector elements of RTM, thus treating the human as a robot, and added a new element: collaborative actions. The removal of a stacked connection is described by the actions listed in Table 2. Each action has a precondition and transmission vector, which describes when the action is executable and how it affects the state. Each action defines which program on the robot controller is called when the action gets executed. The attribute list of an action is used to parametrize the robot program with the needed parameters. For example, the actions *GotoPartPos* and *GotoBoxMagazinPos* in Table 2 have the position of the part and box in the actions parameter list. Both actions call the same robot program, which only moves from its current position to the "send" part or box position. Furthermore, actions have a cost value that is dynamically assigned with respect to the user's choice of assistance. If an action is part of the assistance the user wants, it gets a lower cost assigned. In this example, we have mentioned three different robot behaviors. The *Manual* behavior defines only human actions and, therefore, manual work. *Automatic* means that the robot works autonomously. The *Collaborative* form describes that the user guides the robot to the unknown part position, and the robot can then grasp the part and put it in the transport box. We use this approach to provide to the users a set of known and predictable robot behaviors that stay adaptive to allow necessary modifications.

Table 2. All actions that could take place in the process of removing the stacked connection. The different assistance opportunities and the corresponding weighting of the action are indicated by L for low and H for high action costs.

Actor	Type	Action	Manual	Automatic	Collaborative
Human	Motion Element	GotoPartPos	L	H	H
		GotoBoxMagazinPos	L	H	H
		GotoRobotGuidePos	H	H	L
	End-effector Element	GrabPart	L	H	H
		ReleasePart	L	H	H
	Collaborative Element	GuideRobotToPartPos	H	H	L
Robot	Motion Element	GotoHomePos	H	H	H
		GotoPartPos	H	L	H
		GotoBoxMagazinPos	H	L	L
	End-effector Element	OpenGripper	H	H	H
		CloseGripper	H	H	H
		GrabPart	H	L	L
		ReleasePart	H	L	L
	Collaborative Element	GetGuidedToPartPos	H	H	L

5.3. Searching for solutions

Although the problem of removing part A, which is lying on part B, does not look very complicated, we have to be aware of the combinative size of the problem. In a naive breadth-first search approach, with a branching factor of 8 and the minimal process depth of 4, we have 8^4 (4,096) different paths to compute. A method of improving the algorithm is the use of an extended state list that stops us from extending paths on nodes we have already extended. Furthermore, we have implemented a policy that forbids the algorithm to use two motion elements in a row. This already gives us (for the problem mentioned) a good computational state space. To find the best path of actions, we have used a branch and bound search with extended list and the policy (see figure 6). If the initial state could be a starting point of all assistance behaviors, we would get without dynamic weighting only one form of assistance. With dynamic weighting, we get the user selected form of assistance. This approach of assigning the work content to the agents might seem overloaded on the problem we mentioned, but it uses a generic

approach that we can use for all other disassembly subproblems, such as screwing and so on. Also, our technique gives the assistance system a highly adaptable behavior. If the user leaves the workplace, his/her actions are not usable, and the only behavior is the *Automatic* one. If the part position is unknown in the *Automatic* mode, the next found assistance form is the *Collaborative* behavior (see figure 7). In the case that the search algorithm might not find a suitable sequence of actions (solution), it can explain its reasoning process through the symbolic structure of the situation state space. Also, a great advantage of the search approach is that when we build these detail plans, task after task, we do not have to worry about plan merging, because at each task we create a new plan on the actual situation.

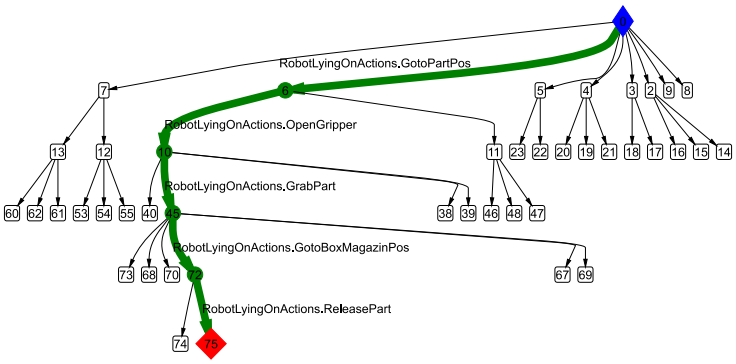


Figure 6. A state space created by the search algorithm. From the initial state (blue diamond) and with the *Automatic* assistance behavior, a solution state (red diamond) was found, and the robot actions on the green path could be executed.

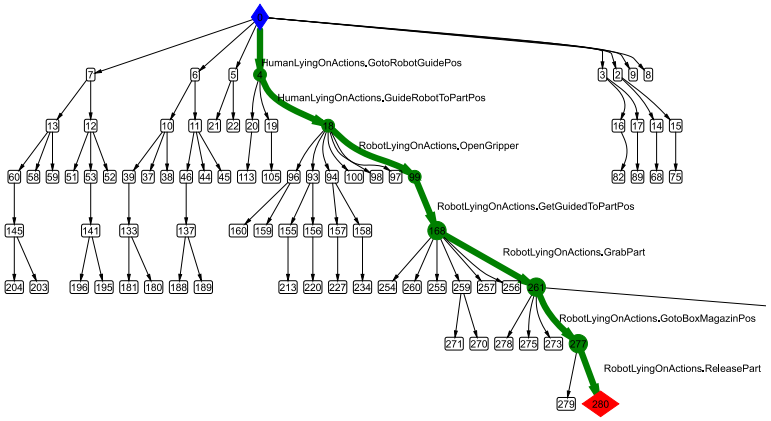


Figure 7. This state space was created by the search algorithm also with the *Automatic* assistance selected, but through the fact that the part position is unknown, the solution consists of the actions of the *Collaborative* assistance behavior.

6. Execution of the Detailed Plans

The "Control" module of the agent is responsible for the execution of the robot programs, for the guidance of the user and for the synchronization of both. The robot programs are invoked and started through a TCP/IP socket connection from the Robot Controller. Then, the robot program starts and connects as a client to our agent to receive the actions parameters list items. At the moment, the disassembly is processed task by task and action after action. The synchronization of the user and robot actions rely on the user's pressing of the confirmation button. To improve the synchronization, we want to track the human hand movements to estimate if an action was carried out. This lets us keep track of the user progress. For example, if we observe that the user moves his hand to the part position and then to the box position, we can assume that he has removed the part. This would be a very efficient way to synchronize the users' and robots' actions. Another important goal is to enable the

parallel execution of tasks. The reduced robot motion in collaborative workplaces and the step-by-step workflow wastes much time, and the user gets bored waiting for his/her turn. So this is a very crucial skill for the assistant. Another ambition is to enable the user to give commands through a gesture or voice in full operation. This could be used to give simple commands, such as “stop,” “open gripper” or “close gripper,” or to change the assistance behavior completely and force detailed replanning.

7. Conclusion

In this paper, we have represented an approach of a robot-based disassembly assistant controlled by an informed software agent. We have discussed the need for a common workflow for a fluent, safe and purposeful assistance in collaborative disassembly. We have further described our approach to inform the agent with product models and our developed two-stage process of workflow planning. Therefore, we have explained the first planning step, which is based on the product graph, topological sorting, and task planning algorithms. We have also illustrated the second task-based planning step, which has focused on the refining of the workflow depending on the situation and the user-chosen assistance through a branch and bound search algorithm. In Section 6, we have discussed executing the robot assistant behavior. Finally, we can summarize that the information and methods we have provided to the robot assistance system enable higher autonomy to perform valuable assistance.

Acknowledgments

This research is conducted in a cooperation among the Company SEW EURODRIVE, the University of Luxembourg and the University of Applied Sciences Trier - Environmental Campus Birkenfeld.

References

[1] Wolfgang Gerke, *Technische Assistenzsysteme: Vom Industrieroboter zum Roboterassistenten*. Berlin: De Gruyter Oldenbourg, 2015.
[2] S. L. Kendal and M. Creen, *An Introduction to Knowledge Engineering*. London: Springer-Verlag London Limited, 2007.
[3] A. Perzylo *et al.*, “Intuitive instruction of industrial robots: Semantic process descriptions for small lot production,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2293–2300.
[4] B. Ludwig, *Planbasierte Mensch-Maschine-Interaktion in multimodalen Assistenzsystemen*. Berlin: Springer Vieweg, 2015.
[5] R. R. Murphy, *Introduction to AI robotics*. New Delhi: Prentice-Hall of India, 2004, c2000.
[6] G. Dini, F. Failli, and M. Santochi, “A disassembly planning software system for the optimization of recycling processes,” (en), *Production Planning & Control*, vol. 12, no. 1, pp. 2–12, 2001.
[7] L. S. Homem de Mello and A. C. Sanderson, “A correct and complete algorithm for the generation of mechanical assembly sequences,” in *Proceedings, 1989 International Conference on Robotics and Automation: A correct and complete algorithm for the generation of mechanical assembly sequences*, May. 1989, pp. 56–61.
[8] L. S. Homem de Mello and A. C. Sanderson, “AND/OR graph representation of assembly plans,” (en), *IEEE Trans. Robot. Automat.*, vol. 6, no. 2, pp. 188–199, 1990.
[9] Y. Jyh-Cheng and L. Yi-Ming, “The Structure Representation for the Concurrent Analysis of Product Assembly and Disassembly,” Department of Mechanical and Automation Engineering, 2005.
[10] H. J. Kim, S. Kernbaum, and G. Seliger, “Emulation-based control of a disassembly system for LCD monitors,” *The International Journal of Advanced Manufacturing Technology*, vol. 40, pp. 383–392, 2009
[11] G. D. GRADI, “PETRI-NET Modelling of an Assembly Process System,” Institute of Information Theory and Automation - Department of Adaptive Systems, Prague, Czech Republic

[12] Xu, C. Wang, Z. Bi, and J. Yu, "Object-Oriented Templates for Automated Assembly Planning of Complex Products," (en), *IEEE Trans. Automat. Sci. Eng.*, vol. 11, no. 2, pp. 492–503, 2014.
[13] M. Merdan, W. Lepuschitz, T. Meurer, and M. Vincze, "Towards ontology-based automated disassembly systems," in *IECON 2010 - 36th Annual Conference of IEEE Industrial Electronics*, pp. 1392–1397.
[14] N. Lohse, H. Hirani, S. Ratchev, and M. Turitto, "An ontology for the definition and validation of assembly processes for evolvable assembly systems," in *(ISATP 2005). The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, 2005*, Jul. 2005, pp. 242–247.
[15] S. Chen, J. Yi, H. Jiang, and X. Zhu, "Ontology and CBR based automated decision-making method for the disassembly of mechanical products," (en), *Advanced Engineering Informatics*, vol. 30, no. 3, pp. 564–584, 2016.
[16] S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, and S. Gupta, "An industrial robotic knowledge representation for kit building applications," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pp. 1365–1370.
[17] Q. Yang, *Intelligent planning: A decomposition and abstraction based approach*. Berlin: Springer, 1997.
[18] B. Britzke, *MTM in einer globalisierten Wirtschaft: Arbeitsprozesse systematisch gestalten und optimieren*. s.l.: mi-Wirtschaftsbuch, 2013.
[19] D. Schröter, P. Kuhlang, T. Finsterbusch, B. Kuhrke, and A. Verl, "Introducing Process Building Blocks for Designing Human Robot Interaction Work Systems and Calculating Accurate Cycle Times," (en), *Procedia CIRP*, vol. 44, pp. 216–221, 2016.