You've Got a Plan? A Domain Modelling Approach for Collaborative Product Disassembly Planning with PDDL

Dominique Briechle
Institute for Software and Systems Engineering
University of Technology Clausthal
Clausthal-Zellerfeld, Germany
dominique.fabio.briechle@tu-clausthal.de

Andreas Rausch
Institute for Software and Systems Engineering
University of Technology Clausthal
Clausthal-Zellerfeld, Germany
andreas.rausch@tu-clausthal.de

***Abstract*—Product disassembly has become more and more relevant to leverage repair, refurbish and remanufature (3Rs) operations while simultaneously enabling access to spare parts from products whichs lifecycle cannot be extended. Such processes can help tackling global ecological production impact as well as overall resource shortage. Nowadays, those operations are still expensive, time-consuming and error-prone because of the high variety in overall product composition, the cost of manual labour and the limitations of disassembly systems in terms of adaption. Artificial intelligence (AI)-based planning could hereby act as a suitable solution to enable automated systems dealing with decomposition tasks. The Planning Domain Definition Language (PDDL) offers a domain-independent canvas, which is suited to deal with a broad level of compositional variety. However, the lack of a suitable systematic methods to describe hierarchical compositions in PDDL limits its application for adaptive disassembly task planning. This work, therefore, aims to overcome these limitations by proposing a methodology to describe compositions and disassembly systems. We introduce, in the scope of the paper, a formal domain meta-model, capable of depicting such hierarchical structures and therefore enabling a precise disassembly of product compositions. Finally, we conduct two disassembly planning tasks and show the applicability of our method to handle hierarchical compositions and product variety.**

***Keywords*—Collaboration, Disassembly, AI-based planning, PDDL, Circular Economy.**

I. Introduction

With the rise of global demand for products and items, especially electronic and electric products, the number of products per person is at an all-time high.

Simultaneously, product lifecycles are falling short of their actual lifespan because of lack of repair, leading to shorter product lifetime and amplifying the accumulation of electronic and electric waste [1]. The consequences are, therefore, drastically decreasing environmental quality and an increase in harmful emissions [2], which affect the quality of life around the globe.

In order to tackle the generation of new waste, product life cycles must be extended and circular economy operations like repairing, refurbishing and remanufacturing (3Rs) must be leveraged [3]. However, still a huge amount of products currently in use are not treated in a Circular Economy compatible way, which drastically limits their lifetime and at the same time reduces the possibility of repairing, refurbishing and remanufacturing products in order to reuse them [4]. The main barriers are hereby diverse and span from a lack of skilled professionals to economic factors and product-related issues like technical obsolesce and inability to upgrade [5].

In terms of cost reduction and substitution workforce, smart automated systems can compensate to a certain degree those shortcomings with the corresponding soft- and hardware tool sets [6]. In addition, automated systems offer the integration of a huge variety of tools in order to conduct adaptable operations, especially in the field of disassembly of products to prepare them for repairing, refurbishing and remanufacturing, which further elevates the economic feasibility in the long term [7].

However, based on the huge variety of products and their composition, disassembly planning of products is still a difficult process, requiring a high level of domain knowledge and technical skill, especially because of the difficulties in modeling hardware-based, hierarchical products [8]. This must be reflected as well by the automated system and requires therefore a semantically understanding of the product composition to enable adaptable disassembly planning. Additionally, different sub-systems, like multiple robots with different tools, must collaborate with one another to ensure the proper conduction of such operations, which has to be considered in the planning system as well [6].

To ensure such collaborative and adaptable approaches, AI-based planning relies on descriptive tools, like the PDDL, which can support system operators with a sufficient canvas for the generation of a sequenced disassembly plan for specific products. Although PDDL presents the opportunity to describe domain-unrelated problems, it lacks a comprehensive framework suitable for the description of hierarchical structures, which we find in physical products. Products are, therefore, composed of several sub-assemblies, relying on one another and having interconnections in order to form the product. This kind of semantic understanding is however crucial to derive

an adaptive plan with atomic disassembly actions based on the individual features of a product, since different products feature a manifold of different part and connector types, let alone assemblies and modifications.

The paper, therefore, aims to propose a domain meta-model capable of describing hierarchical and complex structures, which allows the adaptive definition of product models. Additionally, the model incorporates a system for illustrating the high variety of given product compositions by extending its core entities with sub-types for the description of physical parts and connectors. Finally, an according domain is formalized in PDDL, mirroring the domain meta-model and therefore enabling the automated generation of disassembly sequence planning for different products.

The paper is structured as follows. Section II presents the related work, consisting of the background of the automated disassembly domain, followed by example use-cases as motivation, in Section III. The overall system's concept is illustrated in Section IV. The PDDL domain description and methodological background are described in Section V. Section VI contains the implementation of the domain meta-model in PDDL and the application of the system on the use-cases described in Section III. The paper closes with a discussion of the findings (Section VII) as well as a conclusion of the paper (Section VIII).

II. Related Work

The currently investigated methodologies and technologies for disassembly planning are manifold. Chang et al. [9] are listing a variety of different approaches, which can be used for this matter. This list includes classical approaches like Graph- and Petri-based planning systems as well as more autonomous methods like intelligent planning tools and algorithms. In general, Lambert et al. [10] differs between two major groups after Heemskerk et al. [11]: disassembly planning and disassembly scheduling. These groups consist of the planning of the detailed level (for sub-compositions) and the sequences necessary to disassemble them and scheduling, defining the planning of the tasks required for the process [10].

Especially for the alignment of different levels of planning into one system the PDDL is a suitable methodology, merging benefits from both the Action Description Language (ADL), developed by Pednault [12] and the Universal Method-Composition Planner (UMCP), proposed by Erol et al. [13][14].

The usage of PDDL for decomposition tasks of assemblies has already been topic in several scientific research works. Hoebert et al. [15], for example, used PDDL for the planning of an unscrewing operation conducted by robots, integrating additionally re-planning to tackle uncertainties in the setup. A similar use-case with emphasis on decision explainability of robotic disassembly was investigated by Zhang et al. [16]. PDDL is further used as adaptive planning foundation for human-robot collaboration cases, which bears resemblance to the use-case described in section four with its multi-tool collaborative aspect [17]. However, these applications are investigating on their behalf different application scenarios and key aspects.

III. Illustrating Examples as Problem Motivation

As already stated, our core motivation is to provide suitable disassembly planning for a variety of different products. As examples, we selected two distinct products, a power tool battery and a smoke detector, which consist both of a hierarchical component setup, while simultaneously consisting of just a few components, making both ideal for the small demonstration use-cases in the scope of our paper.

As can be seen in Fig. 1, both the products feature an

Fig. 1. Images of the power tool battery and smoke detector.

external housing, which encloses the inner life of the product. Because of that, the logical conclusion is that those housings have to be removed in order to reach the inner components of the products. To disassemble the product, we, therefore, need certain steps, which are required to disconnect parts and connections in order. In contrast to human operators, automated systems are way less intuitive and therefore require a clear structure of operations. To disassemble the products in a similar way to Fig. 2 we, therefore, need an accurate plan, consisting of different steps, hence action sequences.

Fig. 2. Image of the disassembled products.

The key is, therefore, the derivation of a common understanding, hence a semantic, by automated entities, which

allows the proper disassembly. Therefore, our domain meta-model approach must not only consider the hierarchical setup of those products but as well the adaptability of the system which is required by the overall product variety. To evaluate our proposed model, we will, therefore, model both products and generate an corresponding disassembly plan.

IV. Overall Concept

In order to reach our goal of an automated, adaptable disassembly system, we derived a concept from our general idea consisting of several sub-systems steps, as shown in Fig. 3.

The disassembly system consists of three overall sections, covering different sub-systems which are responsible for interacting with one another. The system's physical component is the **Disassembly Line**, consisting of the **Scanning device** and the **Disassembly Tools**. The idea of the concept is to capture the external features a product by recording it with an optical device, resulting in a digitized model of the product. This happens through different perceptive sensor units, like 3D cameras, capable of recording a realistic digital depiction of our product.

The product will then be identified by the so-called **Product identifier**, who is responsible for matching the products' system ID with the components of the **Knowledge Base**. This section consists of two subbranches, the **Product Assembly Description Library (PADL)** and the **Disassembly Action Library (DAL)**. The matching of the **Product identifier** is made against the product models contained in the **PADL**. These models consist of a textual description of the product setup, usable as problem definition for the systems' **Planner** and containing the information required for the generation of a sequence plan such as composition of products and sub-assemblies, type of parts and connectors and the relation between these entities. The **DAL** on the other hand contains the counterpart of the planning system to the **PADL** and consists of the set of actions and their descriptions necessary to enable the disassembly planning. They, therefore, have requirements which have to be fulfilled by the systems environment in order to apply them, such as connection types or state conditions. It is, therefore, the component of the system that ensures the adaptability in terms of disassembly plan generation-based on the different incoming models of the **PADL**. Based on the matching of the product model, the **Planner** is using the corresponding set of actions from **DAL** and will generate a sequenced plan containing atomic disassembly actions.

This plan will be taken in by the **Executor**, which is responsible of mapping the action-based plan into command, which can be processed by the **Disassembly Tools'** controllers. Via the respective controllers, the **Disassembly Tools** are able to disassemble the incoming products in a collaborative manner corresponding to the generated plan.

In the scope of this paper, we will focus on the part of our Adaptive Product Disassembly System responsible for the planning of our action sequences, hence our approach consisting of a domain meta-model and, derived from that, our PDDL domain.

V. Domain Meta-Model

The domain meta-model, shown in Fig. 4, builds the systematic foundation of the disassembly planning system. It provides the structure required to describe the components and the setup of our product (described in the green block of the figure) and the corresponding actions necessary to dissolve a given link between the model's entities (red block). The green-marked section of the illustration can therefore be viewed as the product assembly concerning part of the model, the **PADL**, while the red part deals with the representation of the external tools and their effect on the model's structure, hence the actions reflected by the **DAL**.

The general idea of this kind of model description is derived from the block-based software architecture domain [18][19], containing therefore similar elements adjusted to our hardware-based domain. The product model therefore incorporates the four major entities parts, connections, compositions and connectionports, which reflect our domain and enable the construction of product models and their assembly groups on the different levels of the product's assembly hierarchy [18][19].

The model's entities are hereby similar to the once [18] used to describe components for architectural conceptions of software systems. The detailed functionalities of the system's entities and links are described in detail in the following section.

The first sections contains the entities regarding the **PADL**:

- *Composition*: The *Composition* provides the descriptional context of a specific product and the adjunct sub-assemblies. Therefore every hierarchy level and assembly group has an associated *Composition* which has a systemic link to the adjunct *Parts*, *Connections* and *ConnectorPorts*. The *Composition* therefore bridges the different hierarchical levels in a product's structure and enables the domain to dissect the *Connections* from the *Parts*.
- *Part*: The *Part* reflects one of the two physical elements in our domain meta-model. *Parts* are, therefore, the "hardware" of a product and are crucial for the disassembly of the same since they are one of two component types. Based on the type of product, the disassembly planning has to take different compositional hierarchies of specific components into account. *Parts* can therefore have a superior and subordinate *Composition*, which defines the product's structure and the *Part's* setup in the disassembly model. In case of a subordinate *Composition*, the *Part* has a sub-composition which consists of subordinate *Parts* and *Connections*, whereas if the *Part* is part of a superior *Composition*, it is a piece of a governing assembly.
- *ConnectorPort*: *ConnectorPorts* are the instances of the connector-intakes of the assembly's *Connections*. Each *Part* has therefore a minimum of one *ConnectorPort* to

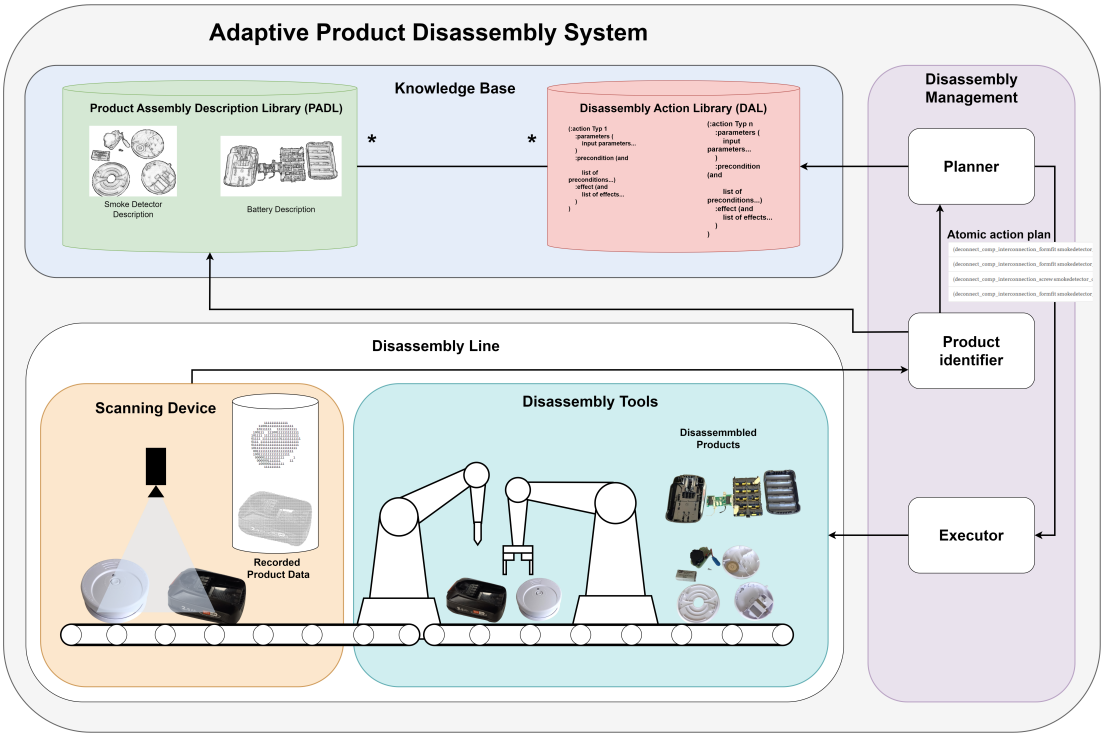

Fig. 3. Illustration of the overall concept.

be connected to an adjunct *Connection* in order to place it into a larger assembly, hence a *Composition*. *ConnectorPorts* can intake on their side multiple *Connections* to be able to design compositions with different setups and *Connection* relations.

- *Connection*: The *Connection* link builds the entities in the meta-model which are responsible for the inter-linkage between the *Parts*. The *Connection* features the attribute ”hierarchy”, which defines the *Connection* to be one of two possible types: *Interconnections*, which mirror physical entities internally in a specific sub-composition and *Transconnections*. These *Transconnections* are the type of *Connections* which enable our meta-model to describe hierarchical *Compositions*. They therefore reflect physical *Connections* which tie different *Compositions* in an assembly to one another. In a physical *Composition*, the *Connections* are reflecting components like screws, solder joints or nails, which are establishing the connection between the *Parts* of an assembly. The *Connections* can be extended, as illustrated in Fig. 4, by different subtypes to match the setups of specific products.

The portions of the meta-model concerning the disassembly actions are contained, as already mentioned, in the **DAL**:

- *DisassemblyAction*: The *DisassemblyAction* is a representation of the actions required to loosen a certain *Connection*. It is therefore directly linked with the corresponding operation to the *Connection*. As the name implies, the action is not an entity of the hierarchical product model and is therefore presented in another color (red). Rather than presenting a component of a product, the *DisassemblyAction* is a representation of the functionality of the disassembly tools required to dismantle certain components. To highlight this aspect, the for the different *Connections* represented extensions are mirrored here by the action extensions.

The different entities of the model are connected to one another via systemic links. These links describe the relation of the entities to one another on a meta-level and are the foundation for the conception of the PDDL actions which we use in order to design our PDDL domain. The following section contains a brief description of the different connectors:

- *has_Comp*: *has_Comp* links a *Part* to a subordinate *Composition*. It therefore describes a relation of a higher level *Part* connected to a lower *Composition*, describing therefore the *Composition* of the sub-assembly of a superior structure.
- *has_Con*: *has_Con* is the interlink between a superior *Composition* and a subordinate *Connection* in the assembly. It is therefore one of two kinds of systematic links to hardware components.
- *has_Part*: The opposite of the aforementioned *has_Con* is the *has_Part* link. It connects in our model a superior *Composition* to a subordinate *Part*.
- *Con_has_CP*: The link *Con_has_CP* describes the relation between a *Connection* and a *ConnectorPort*. Therefore each *Connection* is linked to at least two different *ConnectorPorts* in order to reflect the inlays of a specific *Part*.

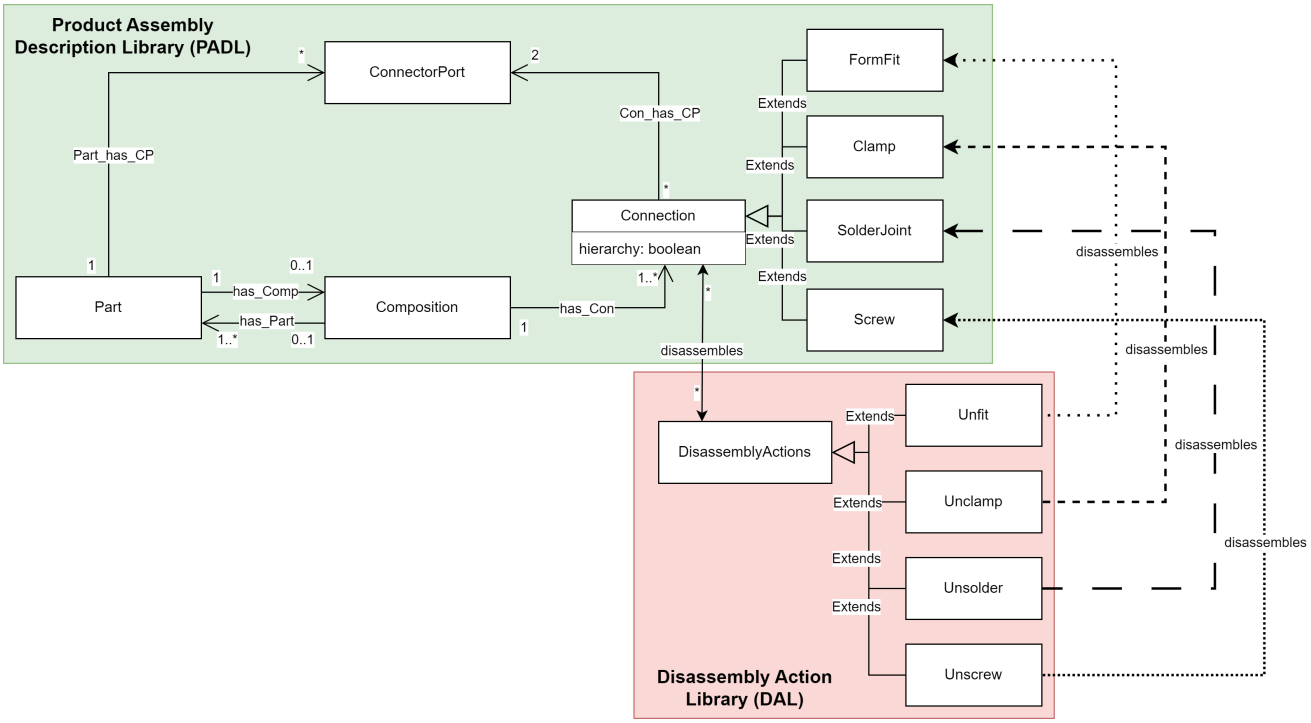

Fig. 4. Domain meta-model for the conception of product models.

- *Part_has_CP*: As the counterpart to the before-mentioned link, the *Part_has_CP* links the *Part* to the specific *ConnectorPorts*. A *Part* must therefore have at least one connected *ConnectorPort* in order to be considerated by an assembly, respectively being disassembled.

VI. Implementation in PDDL

This section describes the foundational idea of PDDL as well as the implementation of the suggested domain model in the definition language.

A. Planning Domain Definition Language

The technical foundation for the used planning system is, as already mentioned, the Planning Domain Definition Language. The Planning Domain Definition Language allows the description of a specific domain. It, therefore, enables the definition of certain constraints and effects, which need to be fulfilled in order to alter the ”world”, meaning the environment of the model, in a certain way. The planning system consists of three distinct elements: the domain, the problem and the planner. The domain and problem are formulated with the help of the PDDL and describe the world’s boundaries in types and conditions as well as its instantiation for a specific scenario. The domain hereby contains the formulation of the world’s restrictions and conditions. These global conditions are represented in the domain as **(:predicates)**, which are at some point in a plan either true or false and are therefore the core aspects, which are defining the interactions in the world.

The abstract entities in the system are described in the PDDL in a hierarchical matter in the form of **(:types)**. These **(:types)** define the inheritance between different entities allowing, therefore, systems with sub-types.

The central element of the domain are however the **(:actions)**. Those **(:actions)** are the ”moving” parts of the system, allowing the planner to create a sequenced row of steps with their corresponding input **(:parameters)**. They therefore have certain **(:preconditions)** defined in order to be ”activated” as well as certain **(:effects)** which will result by conducting an **(:actions)**. As a result of a successfully conducted **(:action)**, the certain **(:effects)** will be triggered, which is an alteration of the world based on the formulated outcome.

While the domain describes the modeling constraints and rules of our world, the problem relates to a concrete instantiation of a certain model. Therefore, the problem contains in our case the description of our product. The components of the product are hereby defined in **(:objects)**, which reflect the instances of the before defined **(:types)** of the domain. The **(:objects)** section is, therefore, composed of all of the entities of an instantiated environment. The core component of the problem is, however, the **(:init)**. This section of the problem contains the initial state of our product, hence the world instantiation. The initialisation of the world’s stage is in PDDL conducted by initializing the formulated **(:predicates)** by setting them as true or false. Finally, the goal stage of the resulting plan is formulated in the problem. This goal is formulated as well out of a certain instantiated **(:predicates)** which have to be either true or not at the end of a planning cycle [14].

B. PADL Meta-Model implementation

With the help of our **PADL** section in the meta-model, the foundation for the PDDL specific **(:types)** and **(:predicates)** is set. The usage of **(:types)** requires in PDDL the setup of a specific requirement, namely the requirement **(:typing)**. As can be seen in the following code Lst. 1, the **(:types)** are directly aligned to the entities in the meta-model. For the base model, the sole attribute is the one for the *Connections* in order to describe the hierarchical correlation between the different layers. The extensions depicted in Fig. 4 are, therefore, the same as in PDDL and will be used in the PDDL domain as well as in the problem of our two example models.

```
(:requirements :typing)
(:types
    part - object
    connectorport - object
    connection - object
    composition - object
    interconnection - connection
    Transconnection - connection

)
```

Listing 1. Typing of the PDDL domain.

The different core elements, typed as objects, can be extended in order to outfit them with fitting subordinates. This can be e.g. specific types of connectors like screws and bolts or types of parts like housings, cells and motors, depending on the targeted product.

The foundation for the later defined **(:actions)** of our PDDL domain are the **(:predicates)**. These **(:predicates)** are directly derived from our meta-model and describe, as already mentioned in the Related Work section, the state functions of our world and can therefore be either true or false for the given objects and their instances in the PDDL problem. As it is the case in our meta-model, the **(:predicates)** are a representation of the links between the system's entities. The formulated **(:predicates)** are, therefore, establishing the link to the respective elements in our PDDL domain, as can be seen in Lst. 2.

```
(:predicates
(comp_has_cp     ?part - part
                 ?connectorport - connectorport)
(has_comp        ?part - part
                 ?composition - composition)
(has_part        ?composition - composition
                 ?part - part)
(has_con         ?composition - composition
                 ?connection - connection )
(con_has_cp      ?connection - connection
                 ?connectorport - connectorport)
)
```

Listing 2. Domain predicates.

C. DAL Meta-Model implementation

As shown above, the **(:actions)** are contained in the **DAL** section of the model. The **(:actions)** are formulated in PDDL on the basis of the **(:predicates)**. Here, the **(:parameters)**, **(:preconditions)** and resulting **(:effects)** are defined, which will be the outcome of a conducted **(:action)**. The **(:parameters)** are, therefore, taking the specified input variables into account, on which the **(:action)** acts on. The PDDL domain, based on the meta-model, consists of four basic **(:actions)**:

- *disconnect_composition-interconnection*: This **(:action)** is designed to cut the systematic link between a superior *Composition* and its subordinated *Interconnections* (Lst. 3). It therefore ensures, that all links, which are part of a certain *Composition* are decoupled, before the disassembly process continues.

```
(:action
disconnect_composition-interconnection
    :parameters (
        ?comp - composition
        ?i1 - interconnection
        ?p1 - part
        ?p2 - part
        ?c1 - connectorport
        ?c2 - connectorport
    )
    :precondition (and
        (has_con ?comp ?i1)

        (forall (?deleg - transconnection)
            (not(has_con ?comp ?deleg))
        )
        (forall (?parts - part)
            (not(has_comp ?parts ?comp))
        )
        (part_has_cp ?p1 ?c1)
        (part_has_cp ?p2 ?c2)
        (has_part ?comp ?p2)
        (has_part ?comp ?p1)

        (con_has_cp ?i1 ?c1)
        (con_has_cp ?i1 ?c2)
        (not(= ?c1 ?c2))
        (not(= ?p1 ?p2))
    )
    :effect (and
        (not(has_con ?comp ?i1))
        (not(con_has_cp ?i1 ?c1))
        (not(con_has_cp ?i1 ?c2))
    )
)
```

Listing 3. Action: disconnect_composition-interconnection.

- *disconnect_composition-Transconnection*: The **(:action)** is functioning in the same manor as the before mentioned *disconnect_composition-interconnection* action, but deals with the sub-type *Transconnection* (Lst. 4). It, therefore, decouples all of the *Transconnections* of a given *Composition* and must be carried out before the *Interconnections* of the *Composition* are disconnected. It is, therefore, responsible for releasing the hierarchy spanning links.

```
(:action
disconnect_composition-transconnection
    :parameters (
        ?comp - composition
        ?t1 - transconnection
        ?p1 - part
        ?p2 - part
        ?c1 - connectorport
        ?c2 - connectorport
    )
    :precondition (and
        (has_con ?comp ?t1)
```

```
        (forall (?allcomp - composition)
            (not(has_part ?allcomp ?p1))
        )
        (has_comp ?p1 ?comp)
        (has_part ?comp ?p2)
        (part_has_cp ?p1 ?c1)
        (part_has_cp ?p2 ?c2)
        (not(con_has_cp ?t1 ?c1))
        (not(con_has_cp ?t1 ?c2))
        (not(= ?c1 ?c2))
        (not(= ?p1 ?p2))
    )
    :effect (and
        (not(has_con ?comp ?t1))
        (not(con_has_cp ?t1 ?c1))
        (not(con_has_cp ?t1 ?c2))
    )
)
```

Listing 4. Action: disconnect_composition-transconnection.

- *disconnect_part-composition*: To continue with the disassembly process, the connection between a *Part* and its subordinate *Composition* must be dissolved (Lst. 5). The action, therefore, checks as **(:preconditions)**, that the specific *Part* isn't bound anymore to a superior *Composition*.

```
(:action
disconnect_part-composition
    :parameters (
        ?part - part
        ?comp - composition
        ?c1 - connectorport
        ?c2 - connectorport
        ?i1 - interconnection
        )
    :precondition (and
        (forall (?over - composition)
        (not(has_part ?over ?part))
        )
        (has_comp ?part ?comp)
        (part_has_cp ?part ?c1)
        (part_has_cp ?part ?c2)
        (not(con_has_cp ?i1 ?c1))
        (not(con_has_cp ?i1 ?c2))
        (not(= ?c1 ?c2))
     )
    :effect (and
        (not(has_comp ?part ?comp))
    )
)
```

Listing 5. Action: disconnect_part-composition.

- *disconnect_composition-part*: This **(:action)** is carried out in order to dissolve the systematic connection between a superior *Composition* and its subordinate *Part* (Lst. 6). The **(:precondition)** is, therefore, that the *Part* does not have any *Connection* to a *Part* of a superior *Composition*.

```
(:action
disconnect_composition-part
    :parameters (
        ?comp - composition
        ?part - part
        ?c1 - connectorport
        ?c2 - connectorport
        ?i1 - connection
        ?i2 - connection
    )
    :precondition (and
        (part_has_cp ?part ?c1)
        (part_has_cp ?part ?c2)
        (has_part ?comp ?part)
        (forall (?links - connection)
            (not(has_con ?comp ?links)
        )
        )
        (not(con_has_cp ?i1 ?c1))
        (not(con_has_cp ?i2 ?c2))
        (not(= ?c1 ?c2))
        (not(= ?i1 ?i2))
    )
    :effect (and
        (not(has_part ?comp ?part))
    )
)
```

Listing 6. Action: disconnect_composition-part.

D. Demonstrator: Power tool battery & Smoke detector

Considering the upper presented domain meta-model and general PDDL domain, we can now define a specified PDDL-domain and problem in order to formulate our disassembly scenario and depict collaborative disassembly approaches for our power tool battery and smoke detector. We therefore have to use the proper extension to describe the parts, connections and tools in a more detailed and domain-specific matter, as shown exemplary in the composition structure diagram Fig. 5. This is necessary in order to assign the different **(:actions)** to the specific tools, which is required to disassemble a specific sub-type of *Connection*, e.g. screw with a screwdriver, and therefore, to enable disassembly in a collaborative matter by dividing tasks between the different tools.

We extend the types *Interconnection* and *Transconnection* with the four different sub-types solder-, screw-, clamp- and formfit- trans/interconnection. The extension of the hierarchy offers the ability to reduce the potential scope of an **(:action)** to a specific sub-type. The **(:actions)** are, therefore, building the framework necessary for our collaborative and automated disassembly environment. It enables our system to plan and assign tasks based on different requirements, e.g. **(:preconditions)**, to different tools and, therefore, using the unique features of the specific collaborator.

The PDDL problem, as mentioned before, reflects the instantiated model and contains all of the components of the product. This is of course highly individual and dependent on the product type and sometimes even the individual product. The **(:objects)** are, therefore, containing all the described components of the product, while the **(:init)** contains their initial condition (instantiated **(:predicates)**, hence the product setup (illustrated in Fig. 5 for one of our example cases)). The **(:goal)** of this demonstrations is formulated as a dissection of the battery's cells and the smoke detectors connection between the battery and the microcontroller, which is respectively placed on the lowest hierarchy level of both of the product's composition. This results in two plans, as shown in Fig. 6 and Fig. 7, consisting of 25 actions for the power tool battery and 15 actions for the smoke detector. For solving the plan,

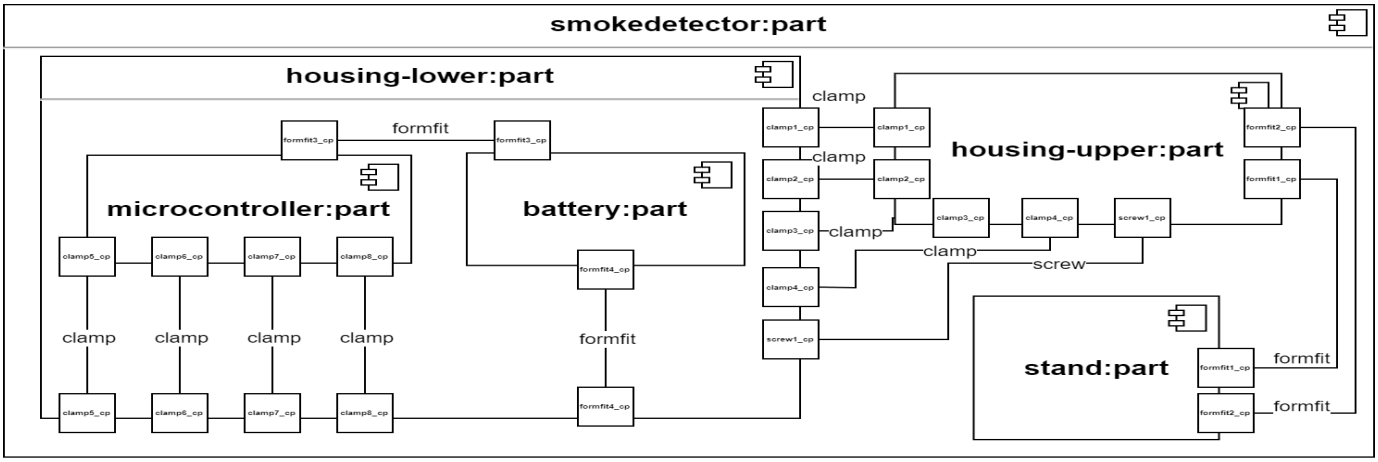

Fig. 5. Composition Structure Diagram of our modeled smoke detector.

```
Total time: 1.05921
Nodes generated during search: 332
Nodes expanded during search: 312
Plan found with cost: 15
Fast-BFS search completed in 1.05921 secs
```

Fig. 6. Planner Output of the smoke detector disassembly resulting in 15 actions until reaching the goal state.

```
Total time: 1.24214
Nodes generated during search: 429
Nodes expanded during search: 169
Plan found with cost: 25
Fast-BFS search completed in 1.24214 secs
```

Fig. 7. Planner Output of the power tool battery disassembly resulting in 25 actions until reaching the goal state.

a Best First Width Search (BFWS) solver was used together with a Fast Forward (ff) parser, which proved to be the best performant solver for both our problems.

The PDDL domain proves to be applicable for both our defined use-cases and shows, how domain meta-modeling can be used as a foundation to derive a suitable domain definition for hierarchical decomposition planning. The extensions of the sub-types have been used in both use-cases, enabling the coupling of certain operations not only to a specific **(:action)**, but also enabling further specification of a task through the used parts and connectors. This results in a more universal PDDL domain, applicable for different disassembly planning tasks.

VII. Discussion

The proposed meta-model shows how a unified structure on the one hand supports disassembly planning by providing a common ground for specific disassembly problems, while on the other showing the applicability of the Planning Domain Definition Language for describing component-based assembly structures. Additionally, it provides an easy-to-use foundation, which allows the integration of different tools based on systematic links of the model's entities.

A. Theoretical Contribution

The theoretical contribution of the paper consists of the domain meta-model and its framework for the domain extensions of the entities based on the instantiation for a specific product. The meta-model provides the theoretical foundation for the formulation of component-based assembly structures. The hierarchical concept depicted by the *Connections* and its two base attributes allows the formulation of more complex *Compositions* to be solved. The model can, therefore, be used as a starting point for further research regarding product disassembly planning via PDDL.

B. Practical Use-Case

The practical implementation shows the application of the formerly described domain meta-model in the context of actual product use-cases in order to demonstrate its effectiveness in disassembly planning scenarios. Especially the extensions of the entities with adjunct sub-types opens up the possibility to design planning models for collaborative scenarios. The applied extensions are, therefore, shown in two different use-cases which feature both the basic expandability of the presented meta-model. The representation of different physical connectors formulated as sub-types mimics the actual composition of a power tool battery and a smoke detector, as shown in Fig. 5. However, the current disassembly tasks have only been simulated on a planning level. The system's overall applicability has yet to be determined in real-world, hardware-driven scenarios with actual automated tools.

C. Model Limitations and Future Research

The current model is suitable to describe and, therefore, plan hierarchical composition scenarios. However, the model has in its current state a major limitation, which affects its application for real-world applications, since it does not take the state of products into account. The state of the physical components

affects the decomposition planning drastically, hence given uncertainties for planning. Defect screws can lead for example to the inability to use a screwdriver as a primary tool and, therefore, leverages a huge impact on an adjunct robotic tool. Currently, the depiction of the state is not fully integrated into the base model, which limits its application to only ”factory new” composition disassembly plans and the states of ”present/not present” for certain components. However, it is planned by the authors to extend the model to allow the proper depiction of the state of different components and to design a methodology to take the same into account during planning.

VIII. Conclusion

The paper describes a system concept, which is capable of disassembling products based on the items individual setup. The main contribution is hereby the meta-model with its instantiated PDDL domain, which enables the generation of highly adaptable disassembly action plans based on the defined product model.

It provides a common ground for depicting disassembly problem modeling and offers an extendable core, usable for a variety of problems. This extendable core further allows the representation of different disassembly tools and enables the automated and adaptive conduction of disassembly steps by the hardware components.

The model, therefore, offers insight into the general mechanisms of the cross-entity interaction and enables the application for a large variety of use-cases. Future research should focus on the aforementioned limitation, especially on the extension of the parameters to depict disassembly problems on a more detailed level and the capability of the system to deal with uncertainties and derivations regarding the product state. Furthermore, the base model can be used in different contexts in order to display its generality for cross-domain applications.

Acknowledgement

This work was conducted in the scope of the project ”Life_TWIN” by the Federal Ministry for Economic Affairs and Climate Action (Research Grant: 03EI5014A).

References

[1] J. Bachér, Y. Dams, T. Duhoux, Y. Deng, T. Teittinen, and L. F. Mortensen, “Electronic products and obsolescence in a circular economy,” *Eionet Report - ETC/WMGE*, no. 3, 2020.

[2] Y. Sitaramaiah and M. K. Kumari, “Impact of electronic waste leading to environmental pollution,” *Journal of Chemical and Pharmaceutical Sciences*, no. JCHPS Special Issue 3, 2014.

[3] M. A. Khan, S. Mittal, S. West, and T. Wuest, “Review on upgradability - A product lifetime extension strategy in the context of product service systems,” *Journal of Cleaner Production*, no. 204, pp. 1154–1168, 2018.

[4] X. Zhang, L. Zhang, K. Y. Fung, B. R. Bakshi, and K. M. Ng, “Sustainable product design: A life-cycle approach,” *Elsevier Chemical Engineering Science*, no. 217, 2020.

[5] N. Roskladka, A. Jaegler, and G. Miragliotta, “From “right to repair” to “willingness to repair”: Exploring consumer’s perspective to product lifecycle extension,” *Journal of Cleaner Production*, no. 432, 2023.

[6] G. Foo, S. Kara, and M. Pagnucco, “Challenges of robotic disassembly in practice,” *29th CIRP Life Cycle Engineering Conference*, no. 29, pp. 513–518, 2022.

[7] K. Wegener, W. H. Chen, F. Dietrich, K. Dröder, and S. Kara, “Robot Assisted Disassembly for the Recycling of Electric Vehicle Batteries,” *The 22nd CIRP conference on Life Cycle Engineering*, no. 22, pp. 716–721, 2015.

[8] S. L. Soh, S. K. Ong, and A. Y. C. Nee, “Design for Disassembly for Remanufacturing: Methodology and Technology,” *Procedia CIRP 15*, no. 15, pp. 407–412, 2014.

[9] M. M. L. Chang, S. K. Ong, and A. Y. C. Nee, “Approaches and Challenges in Product Disassembly Planning for Sustainability,” *The 27th CIRP Design*, no. 27, pp. 506–511, 2017.

[10] A. J. D. Lambert, “Disassembly sequencing: a survey,” *International Journal of Production Research*, no. 41, pp. 3721–3759, 2003.

[11] C. Heemskerk, L. Reijers, and H. Kals, “A Concept for Computer-Aided Process Planning of Flexible Assembly,” *CIRP Annals*, vol. 39, no. 1, pp. 25–28, 1990, ISSN: 0007-8506.

[12] E. Pednault, “ADL: Exploring the middle ground between STRIPS and the situation calculus,” *International Conference on Principles of Knowledge Representation and Reasoning*, no. 1, pp. 324–332, 1989.

[13] K. Erol, J. A. Hendler, and D. Nau, “UMCP: A Sound and Complete Procedure for Hierarchical Task-network Planning,” *Conference on Artificial Intelligence Planning Systems (AIPS)*, no. 2, pp. 249–254, 1994.

[14] D. McDermott, M. Ghallab, A. Howe, *et al.*, “PDDL-The Planning Domain Definition Language,” *Tech Report CVC TR-98-003/DCS TR-1165*, 1998.

[15] T. Hoebert, D. Neubauer, M. Merdan, W. Lepuschitz, Thalhammer, and M. Vincze, “ROS-driven Disassembly Planning Framework incorporating Screw Detection,” *International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, no. 20, 2023.

[16] Y. Zhang, H. Zhang, W. Zhigang, S. Zhang, H. Li, and M. Chen, “Development of an Autonomous, Explainable, Robust Robotic System for Electric Vehicle Battery Disassembly,” *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 409–414, 2023.

[17] S. U. Lee, A. Hofmann, and B. Williams, “A Model-Based Human Activity Recognition for Human–Robot Collaboration,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 736–743, 2019.

[18] C. Deiters, “Description and consistent composition of building blocks for the architectural design of software systems,” Ph.D. dissertation, University of Technology Clausthal, Clausthal-Zellerfeld, GER, 2015.

[19] S. Herold, “Architectural Compliance in Component-Based Systems - Foundations, Specification, and Checking of Architectural Rules,” Ph.D. dissertation, University of Technology Clausthal, Clausthal-Zellerfeld, GER, 2011.