Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-023-02211-3

Concept for the automated adaption of abstract planning domains for specific application cases in skills-based industrial robotics

Lisa Heuss[1] · **Daniel Gebauer**[1] · **Gunther Reinhart**[1]

Received: 27 April 2023 / Accepted: 1 September 2023

Abstract
High product diversity, dynamic market conditions, and a lack of skilled workers are current challenges in manufacturing. Industrial robots autonomously planning and completing upcoming production tasks can help companies address these challenges. In this publication, we focus on autonomous task planning within industrial robotics and investigate how to facilitate the use of automated planning techniques from the field of artificial intelligence for this purpose. First, we present a novel methodology to automatically adapt abstractly modeled planning domains to the characteristics of individual application cases a user intends to implement. A planning domain is a formalized representation of the robot's working environment that builds the basis for automated planning. Second, we integrate this approach into the procedure for developing skills-based industrial robotic applications to enable them to perform autonomous task planning. Finally, we demonstrate the use of the methodology within the application field kitting in two reference scenarios with a mobile robot and a stationary robot cell. Using our methodology, persons without expertise in automated planning can enable a robot for autonomous task planning without much extra effort.

Keywords Industrial robot · Task planning · Artificial intelligence · Automated planning · Planning domain definition language (PDDL) · Skills

Introduction

Due to current challenges, such as high product variety, unpredictable market dynamics, or the shortage of skilled workers, manufacturing companies must establish and integrate new strategies based on technological innovations into their procedures (Koren et al., 2018; Peichl et al., 2022). In this context, industrial robots are considered enablers: They can enable flexible production processes, help companies become resilient to variations in demand, and increase productivity by supporting workers with tiresome tasks (International Federation of Robotics, 2020).

However, programming industrial robots with conventional procedures (e.g., teach-in) still requires specialized knowledge and is time-consuming (Hägele et al., 2016). New programming paradigms, for instance, skills-based procedures, aim to simplify programming industrial robots. The approach of skills-based robot programming typically applies at least a tripartite structure (Pantano et al., 2022; Pedersen et al., 2016; Thomas et al., 2013): The general idea is to encapsulate the robot's functionalities into parameterizable function blocks, so-called skills, that can be composed over different abstraction levels. In this way, users have a set of primitives (e.g., moving a robot or opening/closing a gripper) at their disposal, which they can use to create manifold higher-value skills (e.g., picking or assembling a part). Based on these skills, users instruct the robot to complete various tasks (e.g., pick a part and assemble it or place it at another location). As skills are reusable in diverse tasks and the abstraction level for programming increases, users require less expertise and time to program a robot (Heuss et al., 2022; Pedersen et al., 2016; Thomas et al., 2013). In this way, using industrial robots for high product variety and smaller batch sizes is becoming increasingly profitable. Drag&bot and FRANKA EMIKA already offer commercially available solutions for programming robots similar

✉ Lisa Heuss
Lisa.Heuss@iwb.tum.de

[1] TUM School of Engineering and Design, Institute for Machine Tools and Industrial Management (iwb), Technical University of Munich, Boltzmannstr. 15, 85748 Garching Near Munich, Germany

Published online: 12 October 2023

to a skills-based approach and, thus, demonstrate its industrial applicability and potential (drag & bot GmbH, 2023; FRANKA EMIKA GmbH, 2023). Nevertheless, for the robot to perform a task, users still have to build the complete skills sequence and parameterize each individual skill. This leads to a huge workload when tasks become more complex, increasing the number of skills required to complete them, or when tasks change frequently (see also Huckaby et al. (2013)).

In that case, the extension of skills-based robots by **automated planning** from the field of artificial intelligence has great potential for simplifying task instruction (Huckaby et al., 2013; Pedersen et al., 2016; Rovida et al., 2017). Then, users only need to specify the goal to be achieved by the robot, while the robot autonomously plans the necessary skills sequence to get there. In this way, robots can also be used in applications where each task differs from the previous one (e.g., batch size one production). One example here, which we will continue to address in this publication, is a mobile robot for kitting tasks: The robot is handed over a list of parts that it has to deliver to different workplaces. Based on this information, the robot autonomously plans and performs the necessary actions for collecting requested parts from a storage site and delivering these to workplaces.

In order to use automated planning for robot task planning within industrial robotics, it is necessary to understand its basic functionality. This can be broken down into three main steps (Ghallab et al., 2016): Firstly, the application and its characteristics (e.g., the kitting application) are described in a descriptive model called the **planning domain**. The planning domain relates to a state-transition system. It defines the possible states of a system, the actions that can be performed, and a related state-transition function. Secondly, based on the planning domain, a **planning problem** can be described by the current initial state and the previously mentioned goal to be accomplished. Thirdly, an **automated planning system** takes the specified planning domain and problem as input and solves the planning problem by finding an action sequence for proceeding from the initial state to the goal. A robot can then execute the found action plan with its skills.

The just-introduced planning formalism is domain-independent (Ghallab et al., 2016). Thus, it is suitable for various application fields. However, modeling planning domains is challenging (Kootbally et al., 2015; Ortiz et al., 2013; Rovida et al., 2017; Vaquero et al., 2013): It is complex and has to be conducted by a person with expertise in automated planning and the application. In addition, this mainly manual specification process is error-prone and time-consuming. In the future, widespread and flexible use of industrial robots is predicted. If we want to widely deploy robots that autonomously plan their tasks here, underlying planning domains must be modeled or adapted individually for the various robot applications. However, this is not feasible and profitable for manufacturing companies. First, manufacturing companies lack the required expertise in automated planning. Second, recurring need for modeling and adapting planning domains would cause frequent and rapidly increasing workloads.

The overall objective of our work presented in this publication is to reduce manual effort and the expertise required for describing planning domains when initially setting up an industrial robot for a new application case or reconfiguring it for another one. Our novel idea is to automatically generate a suitable planning domain for an individual application case to be implemented by adapting an abstractly modeled planning domain to the requirements of the specific application case. For this, we present a new methodology and demonstrate its integration into the procedure for developing skills-based industrial robot applications. In this publication, we investigate this approach on a conceptual basis and conduct a feasibility study based on two reference scenarios. The presented work contributes to the area of applied science by introducing a new concept of how to use automated planning from the field of artificial intelligence within industrial robotic scenarios. By utilizing our approach, we support manufacturing companies to enable their robots for autonomous task planning more easily and efficiently. In this way, companies will benefit from more flexible application possibilities of industrial robots that will help them to increase their productivity and competitiveness in the long term.

In the next section, we review the state of the art. Then, we present our methodology in "Methodology" section and its application within two reference scenarios in "Reference scenarios" section. Afterward, we discuss both in "Discussion" section. Finally, we conclude our work in "Conclusion and outlook" section.

State of the art

Potentials of automated planning for task planning of skills-based robots

We like to start the analysis of state of the art with a summary of the potential of automated planning for skills-based robots based on previous works in this field:

- *Wide applicability in diverse applications*: The subject area of planning distinguishes between domain-specific and domain-independent procedures (Ghallab et al., 2016): Domain-specific approaches address topic-specific problems and are tailored to solving these. This field includes, for example, motion planning or simulation-based process planning. Domain-independent planning abstracts these topic-specific characteristics and provides algorithms that build on the commonalities of diverse planning problems. This domain-independent character makes

automated planning applicable far beyond the boundaries of individual thematic application areas. This is especially promising for industrial robotics, as we assume to use future robots in various manufacturing applications that we may even be aware of today.

- *Suitability for planning action sequences based on the robot's current set of skills*: When it comes to task planning for skills-based robots, we aim to find a sequence of skills to complete a task specified at an abstract level. This is precisely the planning problem that automated planning methods address. Previous works demonstrated the suitability of automated planning for this purpose and showed that planned actions can be directly mapped to robot skills (e.g., Pedersen et al. (2016)). In this case, skills implement a similar interface as the actions used for planning based on defined input parameters and a set of preconditions and effects for describing the state change they perform.
- *Modular structure and exchangeability of planning models and algorithms*: As introduced in "Introduction" section, the general structure of automated planning systems is simple and modular: Domain-specific content is described in the planning domain and problem. Domain-independent planning algorithms take this information as input and output a plan. Through the introduction of the **Planning Domain Definition Language (PDDL)** (Ghallab et al., 1998) also exists a commonly used quasi-standard for describing planning domains and problems. This allows easy exchange of planning models and algorithms (Ghallab et al., 1998). Over time, PDDL and related planning algorithms have constantly evolved (e.g., see Ghallab et al. (2016) and Green et al. (2023)) for a more thorough overview). The field of industrial skills-based robotics can benefit from this easy set-up and defined interfaces of automated planners and the variety of existing PDDL versions and planning algorithms. Planning models can be shared between various robot applications, and different planning algorithms can be tested against each other to find the best-fitting approach for a specific scenario. For instance, in the skills-based robot architecture from Rovida et al. (2017), users can easily integrate the planning algorithm of their preference.
- *Checking and optimizing planned action sequences*: Manual specifying skill sequences is often characterized by time-consuming trial and error to find consistent and optimized plans. This can be counteracted using automated planning because the previously mentioned formal specification of action's state changes prevents inconsistencies in found plans, and there exist planners supporting metrics and optimization (Huckaby et al., 2013).

To sum up, automated planning in combination with PDDL holds excellent potential for task planning of skills-based robots in manufacturing applications. For this reason, we put it in the focus of this work.

Introducing the planning domain definition language

After the short introduction of PDDL in the previous section, we describe its basic syntax based on Ghallab et al. (1998) in this section: Within a planning domain in PDDL, states are represented by types and predicates. **Types** are similar to classes in object-oriented programming. The type *object* is a built-in type. New custom types can be subordinated to it and may, in turn, have subtypes. **Predicates** are used to describe the characteristics of a state, taking the previously defined types as arguments. Predicates can be either true or false. **Actions** can be executed to change the state of a system. Each action takes a set of **parameters** as input that refer to the previously defined types. In addition, an action definition consists of a list of preconditions and effects. **Preconditions** describe the state that must be met for an action to be executable. **Effects** specify the state change after completing an action. Based on the formal definition of a planning domain, a planning problem can be described. For this, actual **objects** of the types defined in the planning domain are initialized. Based on these, the **initial state** and **goal** to be reached are described by means of a list of declared predicates. The planning domain and problem are described in two separate files and are handed over to an automated planning system for solving the planning problem.

For an easier understanding, we exemplify this theoretical introduction by means of a robotic pick-and-place application. Figure 1a shows a simple exemplary planning domain in PDDL. We define the robot's *gripper*, considered *parts,* and *areas* where parts can be located as types (lines 2–4). To model possible states, we specify predicates to define the area a part is located in (*part-in-area*), to show if a part is *gripped* by the robot, and to indicate if the robot's gripper is *free* (lines 5–9). With the action *pick*, the robot can pick up a specified *part* from the *area* it is currently located in with its *gripper* (lines 10–22). Based on this planning domain, Fig. 1b illustrates a planning problem in PDDL. In the initial state, the robot's gripper is free, and there is one part in the area in front of it (lines 8–11). In the goal state, the part should be gripped by the robot (lines 16–18). The resulting plan to complete this planning problem consists of just one action. The robot needs to pick up the part.

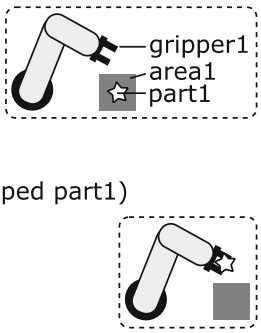
Applying automated planning for task planning of industrial robots

Automated planning has to be incorporated into the robot's system architecture if it is to be used for task planning by industrial robots. For this, we present three exemplary works.

```
1  (define (domain example)
2     (:types
3        gripper part area - object
4     )
5     (:predicates
6        (part-in-area ?part - part ?area - area)
7        (gripped ?part - part)
8        (free ?gripper - gripper)
9     )
10    (:action pick
11       :parameters (?gripper - gripper ?part - part
12       ?area - area)
13       :precondition (and
14          (part-in-area ?part ?area)
15          (free ?gripper)
16       )
17       :effect (and
18          (not(part-in-area ?part ?area))
19          (not(free ?gripper))
20          (gripped ?part)
21       )
22    )
23 )
```

(a)

```
1  (define (problem p1)
2     (:domain example)
3     (:objects
4        gripper1 - gripper
5        part1 - part
6        area1 - area
7     )
8     (:init
9        (free gripper1)
10       (part-in-area part1 area1)
11    )
12
13
14
15
16    (:goal
17       (gripped part1)
18    )
19 )
```



(b)

Fig. 1 Simple exemplary planning domain (**a**) and planning problem (**b**) written in PDDL (Ghallab et al., 1998). In PDDL planning domains, arguments of predicates and parameters of actions are described by the variable name indicated by a preceding question mark and a type. In the planning problem, we additionally visualized the initial state and goal

The SkiROS architecture focuses on the requirements of production industry and provides a software framework for organizing robot knowledge and skills (Rovida et al., 2017). It simplifies software integration and allows one to develop robots with improved reasoning functionalities. In previous work, we presented the REpac framework, a skills-based architecture for developing intelligent and easily configurable industrial robot applications (Heuss et al., 2022). Here, we integrated modules from ROSPlan for task planning. ROSPlan is a more general architecture for integrating task planning into various robotic applications in the Robot Operating System (ROS) (Cashmore et al., 2015).

Generally, the underlying functionality of all three system architectures is similar: In order to maintain the knowledge relevant for planning, the robot has an internal knowledge base (also called world model). The task planning system takes as input a goal to be reached by the robot, extracts the initial state from the robot's internal knowledge base, and writes both information into a planning problem. The task planning system then calls an automated planner to generate an action sequence based on this planning problem and a planning domain. Both the planning problem and domain are written in PDDL. The actions, modeled within the planning domain, match the robot's skills. Thus, the robot executes the planned action sequence using its skills. Simultaneously, it constantly updates its internal knowledge base. This general approach for the system structure of a robot is usable in various application fields. Also, all approaches allow one to easily exchange the used automated planning system as long as it supports the PDDL version used for modeling the planning domain. To build a robot to work in a specific application case, developers mainly need to formalize the knowledge required therein by the robot for planning or other reasoning purposes and implement the skills the robot needs to execute planned actions and update its internal knowledge base.

Creating a planning domain for an application case

The previous section shows that formalizing relevant knowledge of an application case within a PDDL planning domain is essential for utilizing automated planning for autonomous task planning by robots. However, manual modeling planning domains in PDDL is challenging (see also Kootbally et al., 2015; Ortiz et al., 2013; Rovida et al., 2017; Vaquero et al., 2013)): Firstly, modeling planning domains is error-prone and work-intensive. Secondly, this requires specialized knowledge about automated planning, PDDL, and the industrial application case. In this section, we review related works

to simplify the modeling and adjustment of PDDL planning domains for various robotic application cases. Thereby, we focus on works in the field of industrial robotics and in those that use PDDL. We identified three research directions and use these as a classification to present related works.

The first research direction addresses the general modeling of planning domains in PDDL for industrial robot applications (A1). Kootbally (2016) shows how to model robot capabilities for classifying industrial robots in order to select the most suitable robot for a certain task in the application field of kitting. Pane et al. (2021) examine the modeling of and planning with actions to be performed simultaneously (e.g., to avoid collisions while moving to a grasp pose when a disturbance occurs). With a wider scope, Huckaby et al. (2013) consider the application field of assembly. They present a task modeling framework that they use with automated planning. Based on a taxonomy of skills that a robot requires to operate in the application field, they provide a suitable planning domain. In a similar way, Bezrucav and Corves (2022) present a generic planning domain for industrial scenarios with mobile manipulators which they can manually configure for different instances of the generic scenario.

The second research direction considers the learning of action models within a planning domain (A2). Hofmann et al. (2017) record a set of plans within which they determine recurring action sequences. From this, they learn macro actions with preconditions and effects that represent the previously identified action sequences. Subsequently, planning times can be reduced by applying the macro actions in the planning domain. However, the planning system takes the planning domain as input, so it still needs to be modeled a priori. Ortiz et al. (2013) and Liang et al. (2019) apply a learning-by-demonstration approach. Ortiz et al. (2013) use a sensor network to recognize subordinate actions and states of a task demonstrated by a human. Based on that, they learn the action's preconditions and effects. In the context of simplifying robot programming, Liang et al. (2019) use a programming-by-demonstration approach to simultaneously learn low-level actions by kinesthetic teaching and the parameters, preconditions, and effects of high-level actions through visual perception.

The third research direction deals with automatically translating PDDL planning domains from other knowledge formats and models (A3). Here, the most widespread approach is to store necessary knowledge for a robot application in an OWL ontology and derive PDDL files for task planning from this. This approach is applied in the previously presented SkiROS framework (Crosby et al., 2016; Rovida et al., 2017) or a robotic assembly pipeline based on the Factory of the Future (FoF) ontology (Schäfer et al., 2021). Kootbally et al. (2015) store relevant knowledge for robotic kitting applications in OWL and XDSL/XML files from which they can automatically generate planning domains and problems in PDDL. In addition, individual alternative approaches were also examined: The project itSIMPLE provides a design environment that supports users in specifying planning domains and problems using UML diagrams that are then automatically translated into PDDL (Vaquero et al., 2013). SmartPM is a process management system that uses automated planning to find recovery plans in case of execution failure (Marrella et al., 2018). SmartPM assists users in specifying domain knowledge via graphical interfaces, uses situation calculus and IndiGolog for internal knowledge management and translates between these formalisms and PDDL files to integrate automated planning.

Evaluation and contribution

Based on the presentation of related works in the previous section, we identified three approaches for creating PDDL planning domains that are possible for the scenario we are considering. We evaluate these with regard to our work and with a focus on workload and required expertise:

A1 *Modeling planning domains in PDDL*: As already stated, manual modeling and adjusting planning domains in PDDL is time-consuming and requires specialized expertise about the application case, automated planning, and PDDL. Huckaby et al. (2013) and Bezrucav and Corves (2022) discussed the idea of modeling a generic planning domain based on general production skills and using this in multiple diverse application cases. However, Huckaby et al. (2013) mention that connecting planned high-level actions to lower-level executing skills is challenging. We agree with this: Modeling just one planning domain for an application field (e.g., kitting or assembly) and using this in different application cases is not practicable. In reality, the use cases to be implemented differ from each other (e.g., consider diverse objects, skills, or subordinate characteristics) and, consequently, will always necessitate recurring manual and work-intensive adjustments of the planning domain. Bezrucav and Corves (2022) address the manual modification of a generic planning domain for different application instances. However, their focus is on modeling the generic scenario and its planning domain in PDDL. In contrast to this work, we will focus on the automated adaption of generic planning domains to individual robotic application cases in this publication.

A2 *Learning action models by demonstration*: This approach, as shown by Ortiz et al. (2013) and Liang et al. (2019), simplifies creating and modifying a planning domain in a known operating environment. New actions can be easily added to the planning domain by demonstrating these to the learning system. However, if

the structure and characteristics of the operating environment change (e.g., other objects or characteristics of these should be considered), major efforts arise. First, types and predicates for representing the new objects and characteristics must be defined in advance. Second, depending on the objects and related characteristics one wants to capture, different sensors and computation algorithms (e. g., for image processing) are suitable. These must conscientiously be selected and designed. This can be extremely laborious and requires appropriate expertise.

A3 *Translating from other knowledge representations*: All works presented under this research direction have in common that knowledge required for planning is specified in some format that is not PDDL (e.g., OWL, XML, UML) and then automatically translated into a PDDL planning domain. This reduces the required expertise and effort compared to manual modeling planning domains directly in PDDL. Users might be more familiar with the respective knowledge format, be supported by graphical interfaces, and formalized knowledge might be used for different purposes. Nevertheless, all necessary information needs to be stated explicitly. Thus, workloads remain that are not to be underestimated. In addition, one must be proficient with the used alternative knowledge representations (e.g., OWL, XML, UML) and interfaces.

To conclude, related works have mainly investigated how to support users in modeling PDDL planning domains. The widespread and diverse use of industrial robots is predicted in the future. To establish automated planning for task planning of robots in manufacturing, we need to consider the requirements of diverse and evolving application cases in different variants of the underlying planning domains. For this, adjustments must be frequently integrated into the planning domains. However, the support of existing approaches is not sufficient for this. First, existing approaches require an expert of the application case itself with additional specialized knowledge in a minimum of one discipline (e.g., automated planning and PDDL in A1, sensors and computing algorithms in A2 when new objects or characteristics must be captured, or another knowledge representation in A3). Persons who set up robotic applications in manufacturing companies often do not have this additional specialized knowledge. Second, the expected need for continuous evolution and maintenance of planning domains causes recurring and accumulating workloads that counteract the efficiency gains of automated task planning by robots. If the planning domain is modeled manually (A1) or translated from another knowledge representation (A3), the related knowledge must be explicitly specified at some point. This results in workloads that constantly increase with the number of adjustments. When action models are learned from demonstration (A2), setting up sensors and programming computing algorithms may cause high workloads.

To overcome these drawbacks, we present a new and alternative approach to the discussed problem. Our novel idea is to share and reuse PDDL planning domains in different robotic application cases by automatically adapting these to the individual requirements of an application case. Previous works mentioned that specified planning domains can be reused for different application cases. However, none has focused considered how to efficiently share planning domains between diverse application cases. In this context, the automatic adaption represents the novelty of our work. We assume that therefore required information is already specified during the general development procedure of robotic applications, can be extracted from this process, and used for automatically adapting an abstract planning domain for the specific application case. In this way, we expect that we can provide an expert in the application case a planning domain with further decreased work effort. The presented work aimed to investigate this novel idea in the context of skills-based industrial robotics. For this purpose, we elaborated a concept for this and conducted a feasibility study. Our resulting scientific contribution can be summarized as follows:

- We introduce a novel methodology for the automated adaption of abstract planning domains for specific application cases, and
- demonstrate its integration into the development procedure of skills-based robot applications.

Methodology

We begin the presentation of the novel methodology by defining its detailed objectives. Then, we introduce our terminology for this publication, give an overview of the methodology and describe it in more detail in the succeeding sections.

Preliminary considerations, objectives, and industrial example of use

Preliminary considerations

The objectives of our methodology are based on three observations that we would like to present in advance based on preliminary considerations:

C1 *Application fields of industrial robots*: The application fields of robots in production can be classified and specified. Today's top three application fields for

industrial robots are handling, welding, and assembly (International Federation of Robotics, 2022). The tasks occurring in them are well studied and described in classic production literature. We see handling and assembly as promising application fields for the work we present in this publication.

C2 *Required skills within an application field*: Different works have studied and described the skills that occur in diverse application fields, for example, for assembly (Hammerstingl & Reinhart, 2018) or logistics and assistive robot applications (Bøgh et al., 2012). These works show that the skills required by a robot to work within a certain application field can be abstracted into a defined set. However, we think that due to differences in the implementation, there will be different instances of a skill in real-world robotic applications (e.g., depending on an object's characteristics, a suitable gripping principle has to be selected; see also Heuss et al. (2018)).

C3 *Sharing of robot skills and related knowledge*: A common idea in skills-based industrial robotics is to reduce development efforts by sharing skills and related knowledge between different robots via a central marketplace/database (Stenmark & Malec, 2015; Wenger et al., 2016). Users can select appropriate skills for an application case from the central skills database to modularly combine these within the robot system or use offered knowledge services.

Objectives

Based on our conclusion from the state of the art, our overall objective is to reduce manual efforts and required expertise for specifying planning domains when configuring industrial robots for diverse application cases. Compared to the approaches in the state of the art, our novel idea is to improve the sharing of planning domains between different robotic application cases by automatically adapting abstractly modeled planning domains to the characteristics of specific individual application cases. For this, we present a new methodology. We structure our approach based on four technical sub-objectives:

O1 *Modeling of abstract planning knowledge*: Based on C1 and C2, we aim to model planning knowledge about typical application fields of industrial robots in an abstract way so that we can adapt it to the requirements of specific application cases later on. This abstract planning knowledge should be stored in a central database analogously to C3.

O2 *Expansion of abstract planning knowledge by relevant information from specific skill instances*: Based on the considered abstract skills, developers can implement different specific instances of these to address application-specific requirements (see C2) and share these via a central skills database (see C3). During this process, we aim to automatically acquire information relevant to planning and expand the abstract knowledge models of O1 by this specific planning knowledge.

O3 *Automated generation of planning domains for specific application cases*: As described in C3, users download relevant skills onto a robot to implement an intended application case. Based on this selection of specific skill instances and the central knowledge modeling build based on O1 and O2, we aim to simultaneously and automatically generate a suitable planning domain for the specific application case.

O4 *Enabling industrial robots for autonomous task planning*: By providing users with a specific planning domain, we aim to support users who are not experts in automated planning to enable their robot to autonomously plan its task in individual application cases.

Industrial example of use based on the FORobotics project

During the presentation of the methodology, we utilize an example of use to clarify our explanations. In this section, we introduce the example of use to which we will refer in the rest of this publication. The research project FORobotics considered the kitting of parts as a suitable application field for autonomous mobile robots (Berger et al., 2021). Figure 2a illustrates the application field *kitting* in a generalized manner as we derived it from the FORobotics project and consider it for the presented work. The mobile robot consists of a mobile platform, a manipulator equipped with a *gripper*, and various sensors for capturing its environment. With the aim of its mobile platform, the robot can *drive* between *locations*. At these locations, as well as on the robot's mobile platform, are *areas* for storing *parts*. The robot can *pick* and *place* parts from these areas with its gripper. Thus, it can deliver parts between different locations and areas.

The FORobotics project involved several industrial partners who have this application case in different forms in their production. We base our example of use on the application case from Mey Maschinenbau Prien GmbH & Co. KG (Berger et al., 2021). Here, the robot is tasked with delivering boxes between shelves in a storage site and workplaces. Based on this, we describe the slightly adapted application case we assume for this publication. Figure 2b schematically illustrates the setup. For this application case, the robot is equipped with a specialized *boxgripper*. On the shelves and at the workplaces are defined *surfaces* to store *boxes*. On its

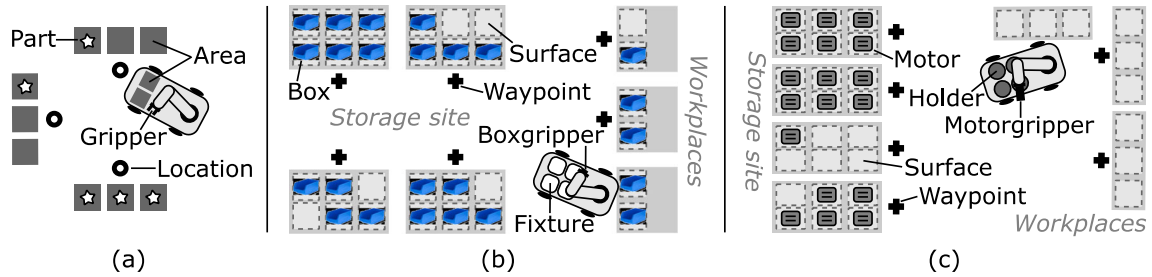


Fig. 2 Schematic illustration of the generalized application field of kitting (**a**) and subordinately derived application cases of box kitting (**b**) and motor kitting (**c**)

mobile platform, the robot has a set of *fixtures* for transporting the boxes. In the robot's navigation software, *waypoints* can be defined, which the robot can drive to afterward.

Terminology

In this section, we clarify the terminology we use in this publication. First, we introduce our general definition of tasks and skills. Then, we present the terminology used for describing planning domains in the context of this publication.

Tasks and skills

We use the terminology for tasks and skills which was derived in a previous work of the authors (Heuss et al., 2022):

- A **mission** specifies the overall task for the robot in the form of the desired goal to be achieved. (e.g., a list of parts with goal areas to which these should be delivered)
- An **action plan** is a sequence of actions to be performed to achieve a mission. Each action in the plan is linked to a robot skill that can execute it.
- A **skill** is a parameterizable functional capability offered by the robot system. Skills can be orchestrated over multiple levels. **Primitive skills** are the functional building blocks at the lowest level and directly communicate with and control the robotic system components (e.g., *move* with a manipulator to a specified pose). **Composite skills** are higher-level complex process flows that result from the combination of lower-level skills (e.g., *pick* an object).

In this publication, we use composite skills to execute the planned actions of an action plan. Thus, and for an easier understanding, in the following, we refer to a composite skill when we use the term skill. Only if the distinction is relevant do we differ between primitive and composite skills.

Planning domains for robotic applications

To enable the robot to autonomously determine an action plan to complete a mission handed over to it, we use automated planning as introduced in "Introduction" section. For this, the automated planner needs a planning domain as a basis. Referring to our considerations and stated objectives in the previous section, we aim to model a planning domain for an application field in a generalized manner so that it can be automatically adapted to specific application cases in the real world. In that way, we consider robotic applications at different abstraction levels:

- An **application field** describes a suitable task range for a robot in a generalized manner (e.g., *kitting*, *assembly*). At this **abstract level**, we abstract the specific setup of a robot platform, the implementation details of skills, or the characteristics of the objects in the robot's surroundings. Thus, we can model generalized knowledge for an application field valid for different application cases.
- An **application case** relates to a real implementation of a robotic system within an application field (e.g., *box kitting*). At this **specific level**, we focus on one actual application case within an application field. For this, we take the previously mentioned details into account as these are relevant for the correct execution of actual application cases.

We transfer these abstraction levels to the modeling of planning domains and skills. Figure 3 illustrates the related meta-model on the left side and exemplifies it on the right side. We start with the terminology for planning domains:

- The **abstract planning domain** formalizes the required planning knowledge for a robot to complete typical missions in a defined application field in a generalized manner. For example, the *kitting domain* describes the knowledge for the application field of kitting in an abstract manner.
- The **specific planning domain** extends the abstract planning domain to represent additional knowledge required by the robot in an actual application case. Related to our

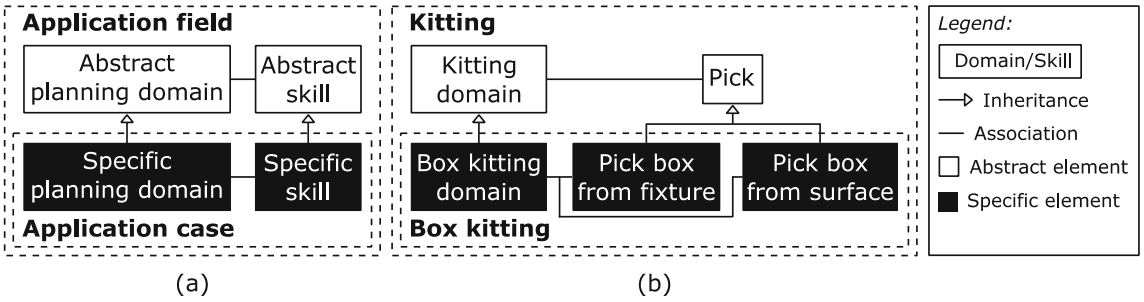


Fig. 3 Meta-model for describing planning domains and skills (**a**) and its application in our example of use (**b**)

example of use, we aim to automatically derive the specific *box kitting domain* from the abstract kitting domain for the application case of kitting boxes.

The robot executes derived action plans by means of its skills. Thus, each planning domain can be linked to a defined set of skills to execute the actions modeled in this planning domain. We have already considered the distinction between abstract and specific skills in previous work (Heuss et al., 2018) and are now integrating this into the view of application fields and lower-level application cases:

- **Abstract skills** are the functional capabilities generally required by a robot to operate in a specified application field. *Pick* is an exemplary abstract skill the robot needs to operate in the application field of kitting.
- **Specific skills** refer to the actual implementations of abstract skills that a robot needs to complete application cases in reality. Thus, specific skills take characteristics such as the setup of a robot system, related kinematics, or functional principles into account. For example, when handling boxes, the robot must always hold them the right way up to prevent the box's content from falling out. In comparison, this is not necessary for the handling of solid objects. Further, the motion sequence and skill parameters differ when picking up a box from a fixture or a surface. This is because the fixtures on the robot's mobile platform are permanently fixed in relation to its manipulator. Thus, the robot can request the box's pose from its internal knowledge base and, based on that, directly pick it up. In contrast, when the box is located at a surface not on the robot's mobile platform, the robot needs to capture the box's pose with its camera before it can pick up the box. Thus, in our example of use, the robot requires the two individual specific skills *pick box from surface* and *pick box from fixture*.

Methodology overview

The overall objective of the introduced methodology is to simplify the use of automated planning in skills-based robotic applications by automatically providing specific planning domains for individual application cases. In the past, we presented the first concept for integrating autonomous task planning based on automated planning into skills-based robotics (Heuss & Reinhart, 2020). In the present publication, we work out this approach. For this purpose, we present the **PDDLAutoGen tool**, a tool for supporting users in creating specific **PDDL** planning domains for their individual robotics application cases based on an **Auto**mated **Gen**eration of these. Furthermore, we presented a software framework for developing skills-based robot applications in previous works (Heuss et al., 2019; 2022). This framework provides a skills control platform and can be extended by a task planning system based on automated planning. Building on this and the life cycle of skills-based robotic applications therein, we demonstrate the use of the PDDLAutoGen tool. Figure 4 gives an overview. To introduce our methodology, we split the life cycle of a skills-based robotic application into three phases skill development, application case configuration, and robot operation. These three phases are illustrated at the top of the figure.

Skill development: First and foremost are **skill developers** who design and implement specific skills (1). During this activity, they describe a specific skill and all its parameters based on the template of a superordinate abstract skill. In addition, they implement the specific skill's process flow. We took over this procedure for developing skills from our previous work (Heuss et al., 2022) and extended it by using the template of superordinate abstract skills. The PDDLAutoGen tool takes the description of a new specific skill as input (2). From this, it automatically extracts knowledge relevant to task planning and integrates it into the **planning knowledge base** (3). All developed specific skills are uploaded to the **specific skills database** by skill developers and so are provided for the later implementation of various robotic application cases (4).

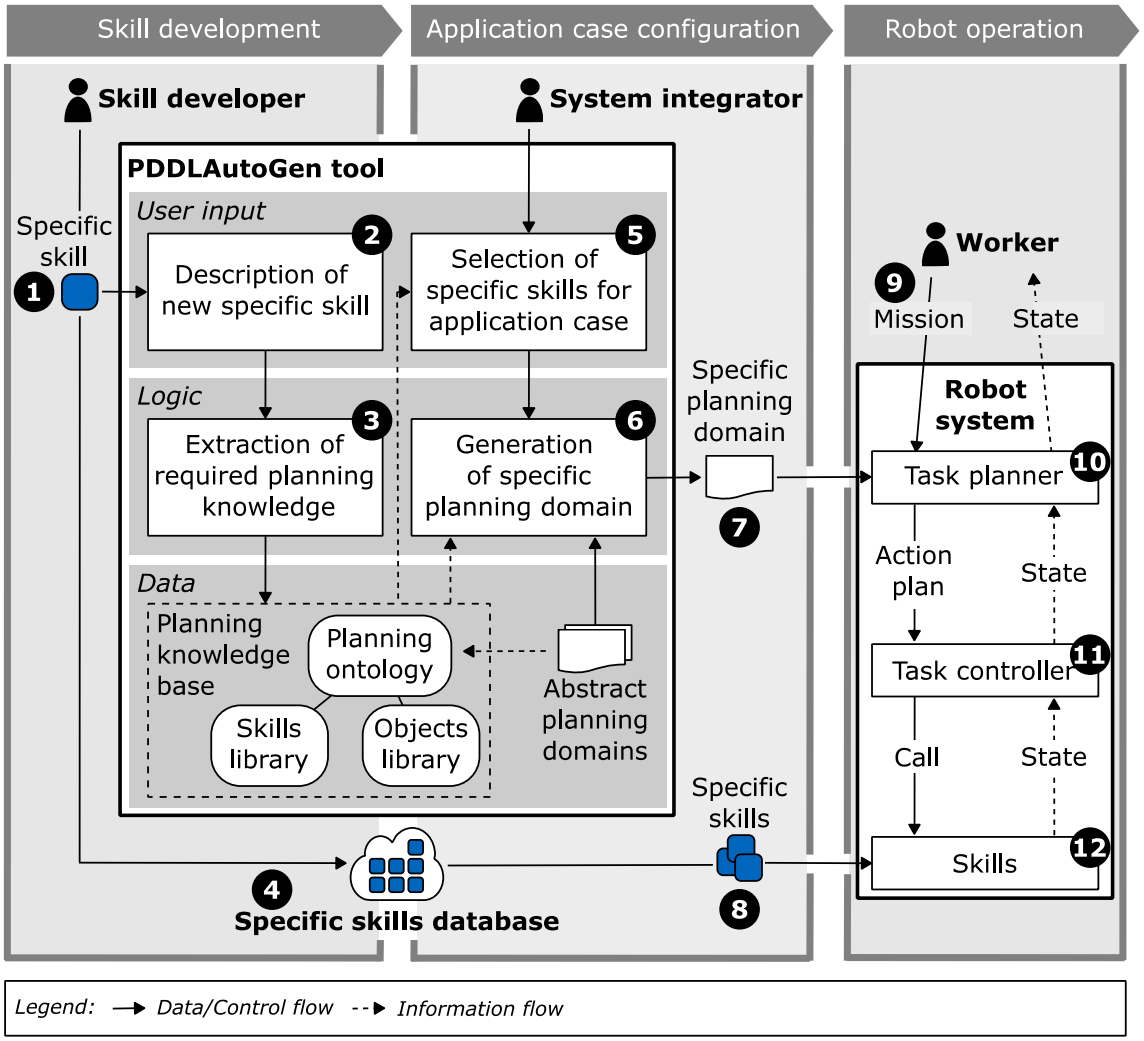


Fig. 4 Overview of the methodology

Application case configuration: In the second step, **system integrators** are engaged to configure a specific application case. At this point, we assume that the **robot system** has already been physically built and software core components (e.g., 3D model of the system setup), as well as our skills-based software framework (Heuss et al., 2022) have been set up. Now, the PDDLAutoGen tool supports system integrators to enable their robot for autonomous task planning. Based on the intended application field, they select the specific skills the robot requires to operate in the application case (5). Based on this input, the PDDLAutoGen tool automatically generates a specific planning domain (6). For this, it uses the information stored in its internal database, which consists of predefined abstract planning domains for defined application fields and the planning knowledge base. The planning knowledge base links the extracted information from previously integrated specific skills to the planning knowledge modeled in the abstract planning domains. System integrators can then follow the procedure for configuring an application case based on our skills-based robot software framework (Heuss et al., 2022). System integrators can directly use the output specific planning domain to configure the robot's **task planner** (7). In addition, system integrators download required specific skills and related software components from the specific skills database onto the robot and configure these for the intended application case (8). In this context, the skills are also extended by preconditions and effects as modeled in the specific planning domain. Moreover, initial knowledge of the application case for the robot can be described in the robot's internal knowledge base if required.

Robot operation: Finally, the robot can flexibly complete various tasks during operation. Our skills-based robot framework (Heuss et al., 2022) offers the functionalities needed for this and works as generally illustrated in "Applying

automated planning for task planning of industrial robots" section. **Workers** instruct the robot at a high abstraction level by specifying the goal to be achieved by the robot in a mission (9). The robot's task planner takes this as input and creates a planning problem by combining the current state stored in its internal knowledge base with the previously specified goal in the mission (10). Subsequently and with the aim of the specific planning domain, the task planner derives an action plan to solve the generated planning problem by internally calling an automated planning system. The action plan is forwarded to the **task controller** that subordinately controls its execution by calling the robot's skills (11). In that way, the robot executes each individual action using the related specific skill (12).

In this publication, we introduce the PDDLAutoGen tool as a stand-alone system. This way, we can better describe its system boundaries and interfaces to other components. However, it could also be integrated into an overall software environment for developing specific skills and configuring application cases coupled with a skills database. In addition, the three considered user roles, skill developer, system integrator, and worker, can be performed by different persons or just one person depending on the persons qualifications and personnel resources in a company. Furthermore, the PDDLAutoGen tool could be used with another skills-based robot architecture that supports automated planning or just a stand-alone automated planning system. In the following sections, we describe the structure and functioning of the PDDLAutoGen tool in detail. For this, we start with the illustration of how to use the PDDLAutoGen tool from the perspective of skill developers and system integrators (user input layer) in "Using the PDDLAutoGen tool" section. Then, we describe the internal functionality and database structure of the PDDLAutoGen tool (logic and data layers) in "Internal functionality and database structure of the PDDLAutoGen tool" section.

Using the PDDLAutoGen tool

Based on the methodology overview in the previous section, we describe in this section how to utilize the PDDLAutoGen tool from users' perspectives. Thereby, we go through the phases of skill development and application case configuration and describe in more detail how skill developers and system integrators interact with the PDDLAutoGen tool.

Developing a specific skill and integrating it into the PDDLAutoGen tool

Figure 5 illustrates the procedure for developing specific skills. We base it on the procedure we previously introduced in the context of our skills-based software framework for robotic applications (Heuss et al., 2022) and extend it so that specific skills are derived from abstract skills. We have marked the steps and information that also represent the inputs of the PDDLAutoGen tool (see step 2 in Fig. 4). In the following, we present the procedure step-by-step:

1. *Initialization of the specific skill*: To initialize a new specific skill, skill developers select the abstract skill to which a specific skill is to be subordinated and name it uniquely. All abstract skills are predefined within the planning knowledge base of the PDDLAutoGen tool. For example, the specific skill *pick box from surface* is derived from the abstract skill *pick*.
2. *Description of the specific skill parameters*: We describe skills in an object-centered manner. We assume that skills are applied to physical objects and, thus, can be parameterized related to these. For example, the abstract skill *pick* can be used to pick a *part* from an *area* with the robot's *gripper*. In this context, each skill parameter is defined by an object class (e.g., *part*) and a unique parameter name (e.g., *?part*). Within the PDDLAutoGen tool, we apply the convention to name skill parameters based on their object class and a prefix. In addition, within the scope of this publication, we indicate these, similarly to PDDL, by a preceding question mark. Now, during the description of the skill, skill developers specify the classes of the specific objects that a specific skill should be applicable to. Hereby, the specific object class is always a subclass of an abstract object class. For example, the skill *pick box from surface* can be utilized to pick up a *box* from a *surface* using a *boxgripper*. Based on that, the specific skill parameters become uniquely named analogously to the related abstract parameters (e.g., *?box*). Furthermore, users can specify individual input and output object parameters representing additional characteristics relevant to skill execution. For example, the skill *pick box from surface* needs a *viewpose* of the *surface* as input. From this pose, it can overview the surface where the box is located to visually determine the box's actual pose with the aim of a camera.
3. *Implementation of the specific skill process flow*: Based on the specified parameters of the specific skill, skill developers implement its internal process flow. Our skills-based software framework offers two options for this purpose. The process flow can either be modeled using a state machine or scripted using Python. The state of the art also applies further approaches (e.g., behavior trees (Paxton et al., 2016)). For the scope of this publication, we decided to use state machines as these are widely known and used. However, other procedures can, in principle, also be combined with the presented methodology. As mentioned in "Tasks and skills" section, composite skills combine subordinate, mostly primitive

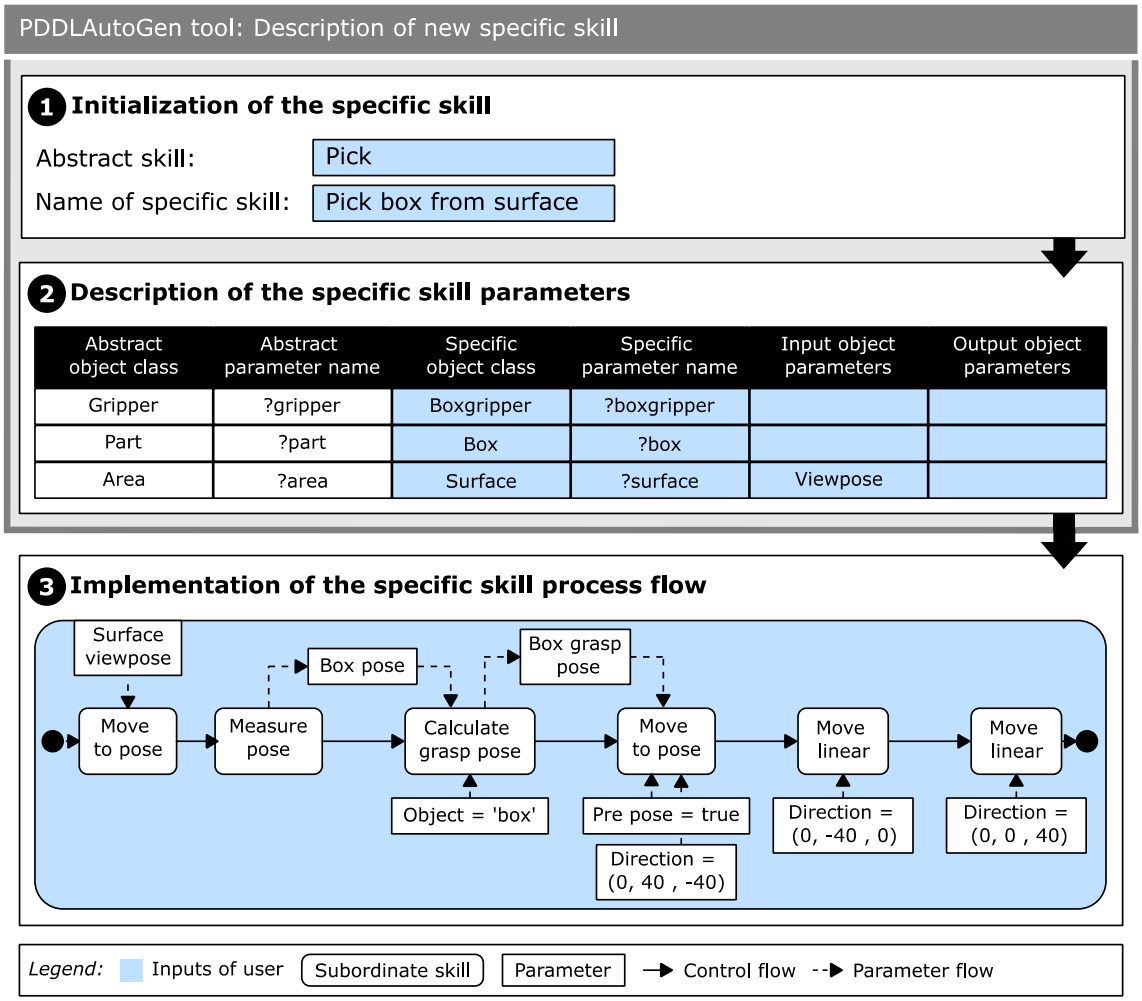


Fig. 5 Procedure for developing specific skills illustrated based on the exemplary specific skill for picking up a box from a surface

skills to execute a complex behavior. In this context, the nodes of the state machine describe the actions to be achieved by subordinate skills, and the transitions define the action's order. Additionally, skill developers have to specify the handling of the subordinate skill's parameters. In this way, internal process flows of skills can be easily designed in an intuitive manner. Subordinate skills for modeling the process flow are also made available via the skills database. Depending on the system's components used, these can be downloaded and integrated to build composite skills as needed. Figure 5 illustrates the process flow of the skill to pick up a box from a surface. Firstly, the robot moves to the surface viewpose to measure the box pose and calculate the related grasp pose. Then, the robot moves into a prepose from which it can thread into the box handle with two linear movements to lift it. Since the boxgripper works similarly to a hook, no explicit closing function is necessary. In Fig. 5, parameters adjacent to the bounding box (e.g., surface viewpose) relate to the skill parameters described in the previous step and are passed to the skill externally when called. The other parameters are either predefined (e.g., direction) or determined by one skill at runtime and passed to a succeeding one (e.g., box pose).

When implementing specific skills, skill developers should consider two aspects. First, abstract planning domains and related abstract skills have a defined validity range (see also "Abstract planning domains" section). This validity range ensures compatibility between abstract planning domains/skills with derived specific planning domains/skills. In this context and based on the related validity range, abstract skills provide requirements to be fulfilled by the implementation of subordinate specific skill instances (e.g., to use a robot with one gripper). These implementation requirements should be considered and fulfilled by skill developers. Second, specific skills should provide self-contained and safe functional behaviors. This means that skills must check their executability in advance and control their correct execution to detect errors when necessary. In this way, skills can be

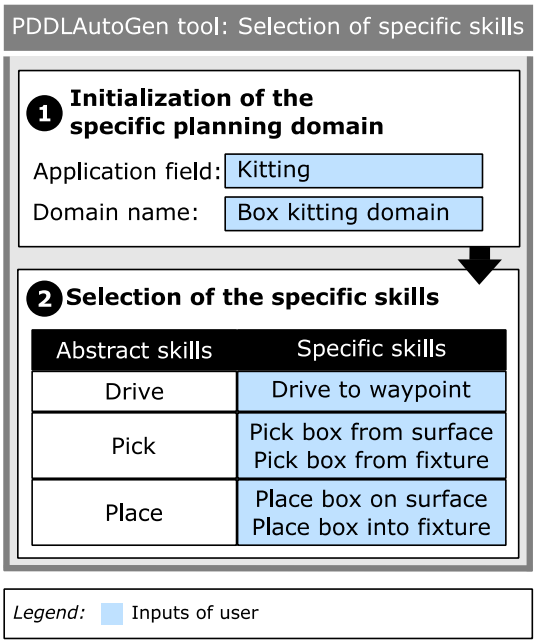


Fig. 6 Procedure for generating a specific planning domain for an application case illustrated based on the exemplary application case of box kitting

called independently from a higher level. If a skill fails to execute a requested action in a plan, it proceeds to a safe failure state and returns an error message. The higher level can then initiate error handling or replanning.

Generating a specific planning domain for an application case

In this section, we detail the procedure for generating a specific planning domain with the PDDLAutoGen tool during the application case configuration phase from the perspective of system integrators (see step 5 in Fig. 4). We superficially presented the procedure for the holistic implementation of an application case using our skills-based robot software framework in "Methodology overview" section. For a more detailed view on this topic, we refer the interested reader to Heuss et al. (2022). In the following and based on Fig. 6, we describe the procedure within the PDDLAutoGen tool step-by-step:

1. *Initialization of the specific skill*: To initialize a new specific skill, skill developers select the abstract skill to which a specific skill is to be subordinated and name it uniquely. All abstract skills are predefined within the planning knowledge base of the PDDLAutoGen tool. For example, the specific skill *pick box from surface* is derived from the abstract skill *pick*.
2. *Selection of the specific skills*: The required abstract skills are defined in the planning knowledge base for each application field. Subordinate to abstract skills, all previously integrated specific skills are further described in the planning knowledge base. Thus, based on the application field selected, in the first step, the PDDLAutoGen tool shows system integrators which abstract skills are required and offers them all subordinate, available specific skills. From these, system integrators select all the specific skills they need to implement the intended application case. For our example, the robot requires skills for driving between different waypoints as well as picking and placing boxes. Figure 6 shows the user's subordinate selection of specific skills.

Based on this input information, the PDDLAutoGen tool automatically generates a specific planning domain for the intended application case and outputs it to system integrators. We present the detailed procedure for this in the next section. In the further course of implementing the application case, users can configure any automated planning system with the specific planning domain that supports the PDDL version and subordinate requirements of the specific planning domain.

Internal functionality and database structure of the PDDLAutoGen tool

In this section, we describe the internal functionalities and structure of the PDDLAutoGen tool. We start with an insight into the modeling of abstract planning domains. Based on that, we analyze necessary adaptions between abstract and specific planning domains. We then explain how necessary information from the implementation of specific skills is stored in the internal planning knowledge base. Finally, we describe the algorithm for generating a specific planning domain for an application case.

Abstract planning domains

We introduced our definition for abstract planning domains in "Planning domains for robotic applications" section. The basic idea is to generalize planning knowledge that a robot requires to operate in an application field into an abstract planning domain. Thus, we aim to specify one abstract planning domain for each intended application field (e.g., kitting, assembly, machine loading). When modeling domain-specific application fields, assumptions are usually made (e.g., a robot with one gripper system is considered). These assumptions should be specified in the form of a validity range to be considered when implementing subordinate specific skills and generating specific planning domains. At this point, we would like to mention that the adaption of abstract planning domains to specific application cases is the novelty of this publication. This requires modeling planning domains, but this is not the focus. In order to demonstrate

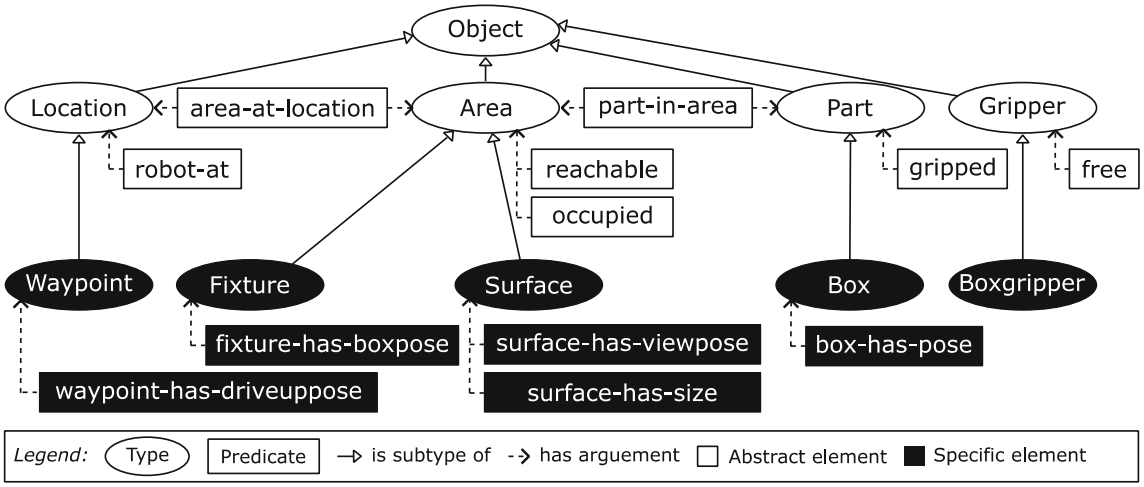


Fig. 7 Types and predicates of the abstract planning domain for the kitting application field (upper part) and the extended specific planning domain for the box-kitting application case (lower part). Subtypes inherit the predicates of superordinate types

our methodology's structure and functionality in the further course of this publication, we modeled an exemplary abstract planning domain for the application field of kitting based on our example of use from "Industrial example of use based on the FORobotics project" section. We introduce it based on Fig. 7.

In its upper part, the figure shows the object types and predicates of the abstract planning domain for the application field *kitting*. The lower part of the figure illustrates the extended specific planning domain for the application case *box kitting* and will be described later on in "Extension of an abstract planning domain to a specific planning domain" section. We created four types, *location*, *area*, *part*, and *gripper*, subordinate to the general type *object*. The object types in the abstract planning domain represent the previously mentioned object classes. By means of predicates, we model the object type's relevant characteristics. The robot can be at a location (*robot-at*), and a set of areas can be assigned to each location (*area-at-location*). All areas assigned to a location where the robot is currently located, as well as areas on the robot's mobile platform, are *reachable* for the robot by means of its manipulator. One part can be placed in each area (*part-in-area*). If a part is placed in an area, the area is *occupied*. If the robot has picked up a part, it is *gripped*. The predicate *free* indicates weather the robot has a part in its gripper or not. To deliver parts between different areas and locations, we modeled in the abstract planning domain that the robot can execute the three actions, *pick*, *place*, and *drive*, by means of its corresponding skills. Figure 8a shows the definition of the abstract action *pick* as an example. The action takes the *gripper* to be used, the *part* to be picked up, and the *area* where the part is currently located as input. All parameters must be named uniquely within the definition of the action. As already mentioned earlier, we name these based on their abstract type and a prefix if required. This way, the names of specific skill parameters can be determined automatically (see also "Developing a specific skill and integrating it into the PDDLAutoGen tool" section), and the mapping between abstract and specific skill parameters can be automatically looked up in the planning knowledge base later on. All preconditions must be fulfilled for the robot to perform the action. Thus, to pick up a part, the robot's gripper must be free, the area must be reachable for the robot, and the part must be in the area. In the same way, the effects represent the state change after executing the action.

Extension of an abstract planning domain to a specific planning domain

This section analyzes the necessary adaptions from an abstract to a specific planning domain. As introduced in "Planning domains for robotic applications" section, specific planning domains extend abstract planning domains to represent individual characteristics of an application case. Thus, the basic structure of the abstract planning domain remains in the specific planning domain. Within an application field, we consider the objects the robot interacts with and the skills it uses for this purpose at an abstract level. In contrast, within an application case, the robot is equipped with specific skills tailored for handling objects of defined specific object classes. Below, we go through the sections of a planning domain and describe the necessary adaptations. The modifications result from the specific skills that system integrators have chosen for their particular application case. Thus, we start with the actions section.

Actions section: To consider the selected specific skill set for task planning, each specific skill must be represented by

```
1  (:action pick
2     :parameters (?gripper - gripper
3     ?part - part ?area - area)
4     :precondition (and
5        (free ?gripper)
6        (reachable ?area)
7        (part-in-area ?part ?area)
8
9     )
10    :effect (and
11       (not(free ?gripper))
12       (gripped ?part)
13       (not(occupied ?area))
14       (not(part-in-area ?part ?area))
15    )
16 )
```

(a)

```
1  (:action pick_box_from_surface
2     :parameters (?boxgripper - boxgripper
3     ?box - box ?surface - surface)
4     :precondition (and
5        (free ?boxgripper)
6        (reachable ?surface)
7        (part-in-area ?box ?surface)
8        (surface-has-viewpose ?surface)
9     )
10    :effect (and
11       (not(free ?boxgripper))
12       (gripped ?box)
13       (not(occupied ?surface))
14       (not(part-in-area ?box ?surface))
15    )
16 )
```

(b)

Fig. 8 Abstract action *pick* (**a**) in the abstract planning domain *kitting* and specific action *pick box from surface* (**b**) in the specific planning domain *box kitting*. Modifications in the specific action definition compared to the abstract action definition are written italic. Extensions of the specific action definition are additionally grey colored

a specific action in the specific planning domain. We create each specific action based on its superordinate abstract action. The general structure of the abstract action with its parameters, preconditions, and effects remains as it is valid for all subordinated specific actions. Based on the skill's specific implementation, two modifications must be conducted between an abstract action and its derived specific action. Figure 8b illustrates these modifications for the already considered specific skill *pick box from surface*. First, it must be considered that specific skills can only be applied to objects of the specific object classes specified in their parameters. To address this, we replace the abstract types with related specific types in the definition of the action's parameters (e.g., *area* by *surface*) and rename the variables accordingly (e.g., *?area* to *?surface*). To keep the action's definition consistent, this renaming must further be performed for the variables of all specific action's preconditions and effects. Second, specific skill parameters can be characterized during skill implementation by additional object parameters required for the skill's execution (e.g., the previously considered *viewpose* of a *surface*). Consequently, and in the case of an input object parameter, the presence of this information is a further prerequisite for the executability of a specific skill respectively action. This can be considered within the specific action's definition by adding additional preconditions. Thus, we create for each input object parameter a precondition using a predicate of the form ***[specific object class]*-has-*[object parameter]*** (e.g., *surface-has-viewpose*). In the same way, we create additional effects in the specific action's definition for considering output object parameters.

Types section: All specific types we introduced in the parameter definition of specific actions must also be incorporated into the types section of the specific planning domain. During skill implementation, skill developers derived the specific object classes from the defined abstract object classes (e.g., *surface* from *area)*. In the same way, we now integrate the specific object types as subtypes of the related abstract object types. In this way, specific object types also inherit the predicates of superordinate abstract types. Figure 7 illustrates this for our example of use in its lower part.

Predicates section: Finally, to keep the description of the specific planning domain consistent, we must also add the newly defined predicates for describing the additional preconditions and effects of specific actions to the predicates section (see also the lower part of Fig. 7).

To sum up, a specific planning domain extends an abstract planning domain. The types and predicates sections contain the abstract elements and additional specific types and predicates. The abstract actions are replaced by specific actions that the robot can execute using its specific skills.

Storing of planning knowledge from the implementation of specific skills in the planning knowledge base

To automatically adapt abstract planning domains for individual specific application cases, we extract relevant information during the implementation of specific skills and store it in the central planning knowledge base for later access (see step 3 in Fig. 4). This section describes the general structure of the planning knowledge base and how information from the implementation of specific skills is stored in it. The planning knowledge base is composed of three modules (see Fig. 4). The **planning ontology** links considered skills and object classes in the abstract planning domains to their specific instances. In addition, the **skills** and **objects libraries** store additional characteristics of the specific skills and object classes. We start by introducing the planning ontology as the central component based on Fig. 9.

The planning ontology consists of three base classes **application field**, **skill**, and **object**. We create a subclass to the application field base class for each considered application field. Analogously, the skill and object base class have

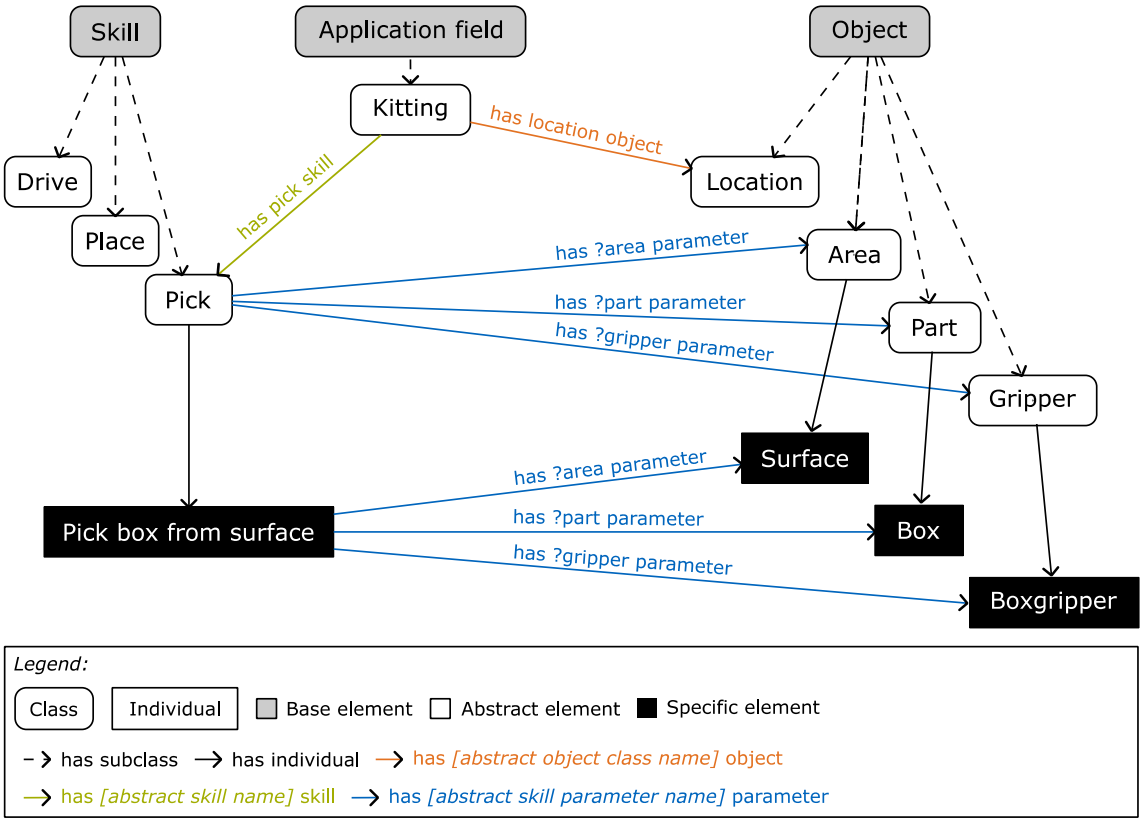


Fig. 9 Base structure of the planning ontology

a subclass for each abstract skill that relates to an action in an abstract planning domain respectively for each object class that occurs as an object type in an abstract planning domain. In Fig. 9, this is again exemplified for the application field *kitting*. The correlations between application fields, abstract skills, and abstract object classes are further specified by three types of object properties. The properties **has *[abstract skill name] skill*** and **has *[abstract object class name] object*** link an application field to the abstract skills the robot requires for operating in it respectively the abstract object classes occurring therein. Further, the property **has *[abstract skill parameter name] parameter*** refers from an abstract skill to the abstract object classes, which the skill takes as input parameters. Within all three types of object properties, the placeholders ***[…]*** become replaced by the respective name of the abstract skill, object class, or skill parameter. As mentioned earlier, for example, the abstract skill *pick* takes the *part* to be picked up, its *area*, and the *gripper* to be used as input. Within the use of the PDDLAutoGen tool during skill implementation, diverse specific skill instances and related specific object classes are added to the planning ontology as individuals. Figure 9 exemplifies this for the specific skill *pick box from surface* in its lower part. Individuals inherit the object properties of their parent classes. In this context, we reuse the object property **has *[abstract skill parameter name]*** parameter to additionally link the specific skill to the specific object classes representing its specific parameters and thus map the relationship between abstract and specific skill parameters.

The skills and objects libraries supplement the planning ontology. In the basic setup, we create one category for each abstract skill as well as object class occurring in the planning ontology. Later during skill implementation, each newly created specific skill, including all its parameters, gets subordinated to its abstract parent element in the skills library. Similarly, specific objects and for these created subordinate characterizing parameters are integrated into the objects library.

Automated generation of specific planning domains during robot configuration

System integrators use the PDDLAutoGen tool to generate a specific planning domain for their intended application case during the configuration of the robot application (see "Generating a specific planning domain for an application case" section). For this, system integrators specify the intended application field of their application case, the name of the specific planning domain to be generated, and the specific skills their robot requires to operate in the application case.

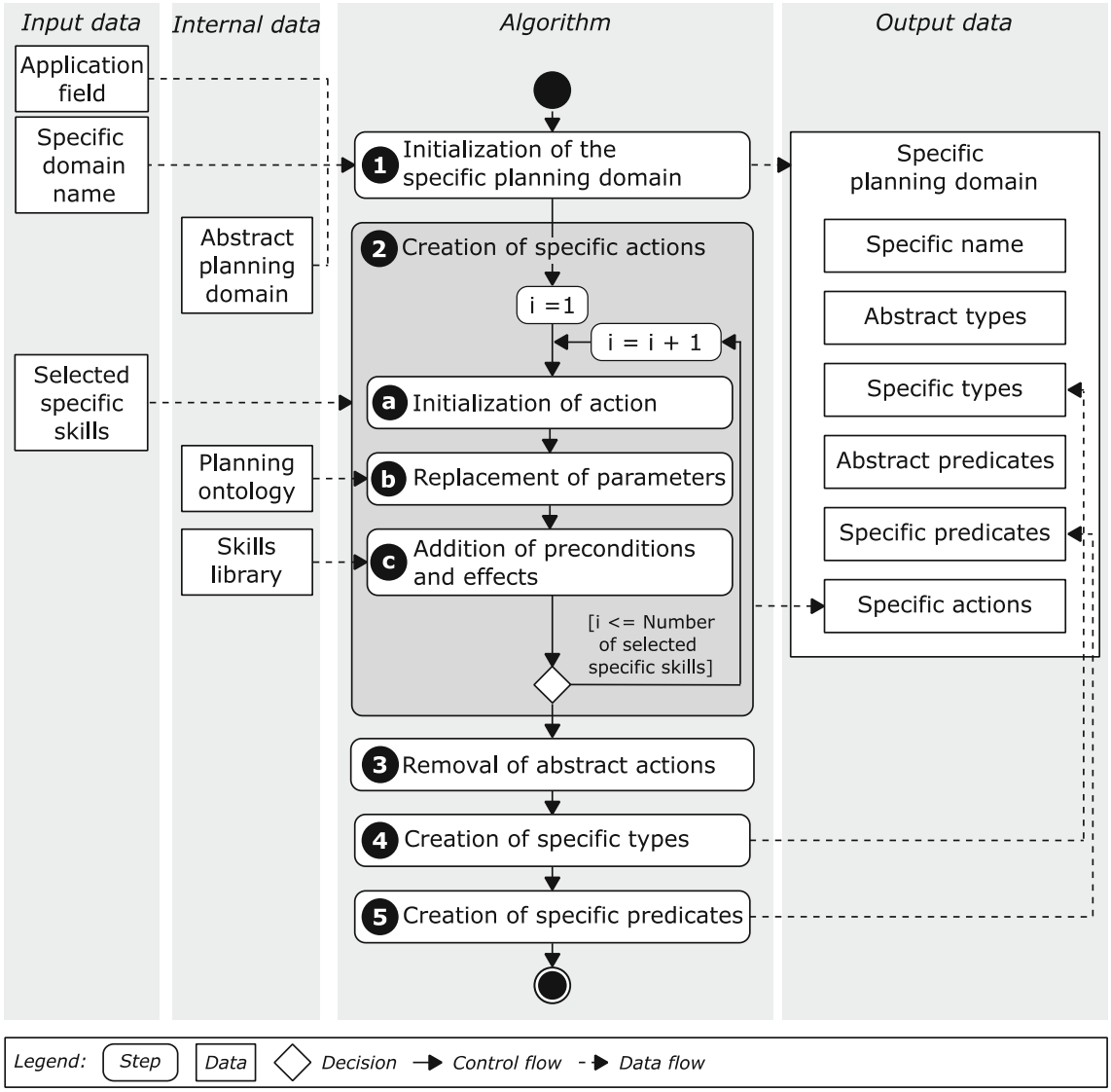


Fig. 10 Algorithm for the automated generation of a specific planning domain

Then, the PDDLAutoGen tool uses this information to adapt the abstract planning domain of the considered application field for the specific application case (see step 6 in Fig. 4). We illustrated the necessary adaptions from an abstract to a specific planning domain in "Extension of an abstract planning domain to a specific planning domain" section. In this section, we summarize the algorithm that performs these adaptions. Figure 10 illustrates its flow. The algorithm's input data specified by the system integrators is summarized on the left side of the figure. In addition, the algorithm utilizes the internal data of the PDDLAutoGen tool stored in its database to look up relevant information that was extracted during the implementation of specific skills. The expected output is the specific planning domain, as illustrated on the right side of the figure. In the following, we go through the algorithm's steps in the figure's center. The specific planning domain is initialized based on the related abstract planning domain and named accordingly (1). In the next step, the algorithm adds the specific actions the robot can execute using the selected specific skills (2). For this, the algorithm performs the same steps for each action: It initializes the specific action based on its superordinate abstract action (2a), replaces/renames the parameters and variables of the specific action accordingly to the related specific skill (2b), and adds preconditions and effects for considering additional object parameters characterizing the specific skill's parameters (2c). The abstract actions are removed after all specific actions have been added because the robot cannot directly execute these (3). All new specific types that occur in the definition of a specific action are added as a subtypes of the related abstract types (4). All specific predicates newly created in step 2c are added to the predicates "Discussion" section.

Reference scenarios

In this section, we present the application of the previously introduced methodology in three different application cases spread over two reference scenarios. Then, based on the generated results, we conduct the evaluation of our methodology. For the use of the PDDLAutoGen tool in the reference scenarios, we implemented it as a software demonstrator. In the database, we described the abstract planning domain in PDDL 2.2 (Edelkamp & Hoffmann, 2004), the planning ontology as OWL ontology, and the skills and objects libraries using JSON. Based on that, we implemented the logic within the PDDLAutoGen tool and a user interface with Python 3. Here, Owlready2 (Lamy, 2017) was used for manipulating OWL ontologies. We modeled one abstract planning domain for the application field of kitting, as already introduced in "Abstract planning domains" section, and integrated it into the database of the PDDLAutoGen tool. Afterward, the PDDLAutoGen tool was used to generate specific planning domains for the application cases of the reference scenarios.

Reference scenario 1: commissioning of parts by an autonomous mobile robot

In the first reference scenario, we aimed to evaluate the general functionalities of the PDDLAutoGen tool, including its benefits and limitations. For this, we build on the FORobotics project that we introduced in the context of the example of use in "Industrial example of use based on the FORobotics project" section. To evaluate the PDDLAutoGen tool, we derived two exemplary application cases in the kitting application field from the industrial partners of the FORobotics project. In the following, we introduce our setup and procedure within this reference scenario before describing the application cases in detail and presenting our results.

Setup and procedure

Our setup consisted of the PDDLAutoGen tool and a stand-alone automated planning system. We used the fast downward planning system (Fast Downward, 2023) as an automated planner. The fast downward planning system supports PDDL 2.2 as well as subordinate requirements, which we used for modeling the abstract planning domain and is frequently used in other research works.

Based on this setup, we proceeded for each application case as follows: First, we identified and designed specific skills that the robot requires to operate in the respective application case and integrated these into the PDDLAutoGen tool following the procedure for implementing skills from "Developing a specific skill and integrating it into the PDDLAutoGen tool" section. If possible, we reused specific skills from previous application cases. Within the first reference scenario, we have not implemented specific skills in software. Second, we used the PDDLAutoGen tool to generate a specific planning domain for the considered application case, as described in "Generating a specific planning domain for an application case" section. Third, we configured the automated planning system with the specific planning domain generated in the previous step. Fourth, we planned five exemplary missions per application case. For this, we specified related planning problems and called the automated planner to solve one after the other.

We have refrained from the full implementation of the application cases in this reference scenario, as the feasibility of the basic motion sequences to realize the application cases have already been demonstrated in the FORobotics project (Berger et al., 2021; Heuss & Reinhart, 2020). Furthermore, we present an overall implemented skills-based robot application in the second reference scenario.

- box9 in shelve-surface8
- box10 in shelve-surface9
- box22 in shelve-surface15
- box12 in shelve-surface16
- box7 in shelve-surface21
- box1 in workplace-surface1

Fig. 11 Exemplary mission in the application case of box kitting. In the mission, we specify one goal per line. During the evaluation, we transferred it into a PDDL planning problem

Application case 1: transporting of boxes between a storage and working site

The first application case relates to our example of use from "Industrial example of use based on the FORobotics project" section, which is based on the application case of the industrial FORobotics project partner Mey Maschinenbau Prien GmbH & Co. KG (Berger et al., 2021). Here, a mobile robot should transport boxes between a storage site and workplaces. In terms of our work, the robot needs five specific skills to operate in this application case. The robot must be able to drive between different waypoints as well as pick up and place boxes from/on surfaces or fixtures on its mobile platform.

Based on the automatically generated planning domain, we specified five planning problems. The initial state of one exemplary planning problem is illustrated in Fig. 2b. For each mission, we specified six boxes that should be moved to a different surface from the storage site to a workplace or the other way. Figure 11 illustrates one exemplary mission. We integrated these missions as goals into the planning problems. The five planning problems differ in the initial areas of the

boxes, the starting waypoint of the robot, and the missions to be completed by considering different boxes and delivery areas.

Application case 2: delivering motors from a storage site to workplaces

We derived the second application case from the industrial FORobotics project partner Krones AG (Berger et al., 2021). Here, the robot is tasked with transporting motors instead of boxes from shelves in a storage site to different workplaces and assisting a human by assembling the motors. In terms of the publication, we do not consider the cooperative assembly and assume that the robot only delivers the motors to the workplaces. In the following, we introduce the application case as considered within the scope of this work and as illustrated by way of the example in Fig. 2c. Similar to the first application case, we assume that the shelves, as well as workplaces, can be reached by the robot from predefined *waypoints* and have defined *surfaces* for storing the *motors*. We use the same mobile platform and manipulator as in the first application case. This time, the robot is equipped with a specially designed *motorgripper*, a camera, and a construction with *holders* to store the motors during transport on its mobile platform. We identified five specific skills the robot requires to act in this application case. For driving between different waypoints, the robot can reuse the specific skill from the first application case. In addition, this time, the robot requires specific skills for picking and placing motors from/on surfaces and holders. Picking and placing motors from/into holders relates to a peg-in hole insertion and, thus, needs a different motion sequence compared to picking and placing motors from/on surfaces.

To illustrate the structure of the specified planning problems, Fig. 2c presents the initial state of one planning problem. In this application case, we change between the five planning problems in the initial state the number of motors present in the storage site, the areas occupied by them, and the starting waypoint of the robot. For each mission, we specify six motors to be delivered from the storage site to various surfaces at the workplaces. We integrated these missions as goals into the planning problems. The five missions differ in the considered motors and delivery surfaces.

Generated specific planning domains and planning results

In this section, we summarize the results of both application cases. We first manually checked the generated specific planning domains. Both specific planning domains have been correctly adapted for the application cases from the abstract planning domain by the PDDLAutoGen tool. We already used the resulted specific planning domain for the first box-kitting application case as illustrating example in "Methodology" section. Figure 7 illustrates all types and predicates of the specific planning domain. All types and predicates of the abstract planning domain remain. Specific object types for specifying specific actions' parameters have been added as subtypes of the related abstract types. Furthermore, all new predicates used to define additional preconditions and effects in specific actions' definitions have been created. The actions section contains the five selected specific actions for driving to different waypoints, picking up boxes from surfaces and fixtures, and placing boxes on surfaces and into fixtures. These have been created based on the superordinate abstract actions with correct adaptions. Figure 8b shows by way of example the definition of the specific action for picking a box from a surface.

Second, we look at the workload we save using the PDDLAutoGen tool compared to manually adapting the planning domains. For this, we analyzed the structure of the generated specific planning domains. In the following, we describe the procedure used here and the results. Comparable to the code excerpts in this publication (see Fig. 1 and Fig. 8), each line of the generated specific planning domains contains a type/predicate definition, an action's precondition/effect, etc. We reviewed each specific planning domain and counted the lines directly adopted from the abstract planning domain, adopted with modification, and newly added by our algorithm. We skipped empty lines or lines with only a parenthesis. Figure 12 presents the results for all application cases. In the first two application cases, the specific planning domains were successfully composed of the abstract planning domain, with 62 modifications in each case (45 lines modified and 17 lines newly added). The PDDLAutoGen tool created the specific planning domains for the application cases automatically based on the chosen application field, the specified application case names and selected specific skills. In contrast, without the PDDLAutoGen tool, one must manually assemble the specific planning domains and incorporate relevant modifications in an error-prone and test-intensive procedure. Thus, this analysis demonstrates the high efficiency of the PDDLAutoGen tool compared to manually adapting the PDDL planning domains. Important to mention that when the robot's skills set becomes more extensive, the size of specific planning domains and necessary adjustments to be incorporated increase significantly more compared to the input information of the PDDLAutoGen tool.

Third, we reviewed the action plans generated by the automated planning system to solve the exemplary missions within the two application cases. The automated planner found an action plan for each of the ten planning problems. We have gone through all action plans and checked that all parts were delivered to the requested surfaces. In this way, we verified that by using the specific planning domains generated by the PDDLAutoGen tool, we could plan action plans for solving missions in the two different application

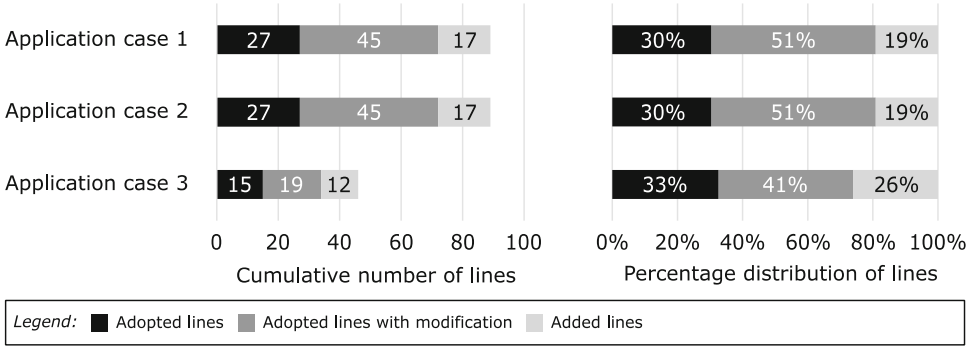


Fig. 12 Analysis of the structure of the specific planning domains created by the PDDLAutoGen tool on the basis of the original abstract planning domain. On the left side, we show a cumulative representation; on the right side, we illustrate the percentage distribution of the overall planning domains

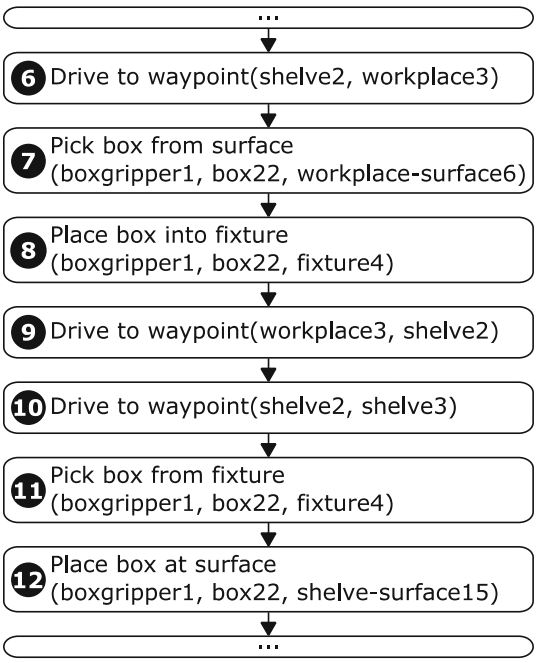


Fig. 13 Excerpt of the action plan to complete the mission in Fig. 11. The complete action plan consists of 38 actions

cases. By way of example, Fig. 13 shows an excerpt of the action plan to complete the mission illustrated in Fig. 11. The robot drives between the different waypoints, picks up the requested boxes, stores these during transport in its fixtures, and places them on the specified surfaces using its available skills. In doing so, it successfully fulfills the given mission.

Reference scenario 2: composition of customized parts sets

In the second reference scenario, we demonstrated the use of the PDDLAutoGen tool in combination with a holistic robot application. In this way, we furthermore showed the execution of created plans by a real robot. Within the application case, the robot is tasked with composing customized parts sets from a storage. We present this reference scenario with the same structure as the first one.

Setup and procedure

The setup for this reference scenario consisted of the PDDLAutoGen tool and the robot cell illustrated in Fig. 14a. The robot cell uses an ABB IRB140 robot controlled by an ABB IRC5 compact control unit and equipped with a 2F-85 two finger gripper and a FT 300 force torque sensor from Robotiq. The force-torque sensor was not used in this application case and is, therefore, also not labeled in the figure. All components are connected to a central computer that runs Ubuntu 18.04.6 LTS and ROS Melodic. In addition, we installed our skills-based robotic software framework on the central computer. We used a slightly revised version compared to the one we presented in Heuss et al. (2022). Similar to the first reference scenario, we internally used the fast downward planner (Fast Downward, 2023) as an automated planner within the task planning system of the skills-based software framework. In this way, the overall setup correlates to the one illustrated in Fig. 4. Using the skills-based software framework, firstly, the robot's skills can be freely configured. Secondly, by providing a planning domain, the robot can be enabled for autonomous task planning. In that case, based on the specification of the planning domain, the robot stores and updates the current state of itself and its surroundings within its internal knowledge base.

We describe our procedure for this reference scenario compared to the first one: First, we initialized and described the required specific skills in the PDDLAutoGen tool (according to "Developing a specific skill and integrating it into the PDDLAutoGen tool" section) and also implemented these in software based on a previous project. Second, we generated the specific planning domain with the aim of the PDDLAutoGen tool and as described in "Generating a specific planning domain for an application case" section. Third, we configured the robot cell for the application case. This step differs from the first reference scenario. Based on the previously described setup of the robot system, we equipped the robot with the required specific skills and subordinate software components, configured the task planner with the specific planning domain generated by the PDDLAutoGen

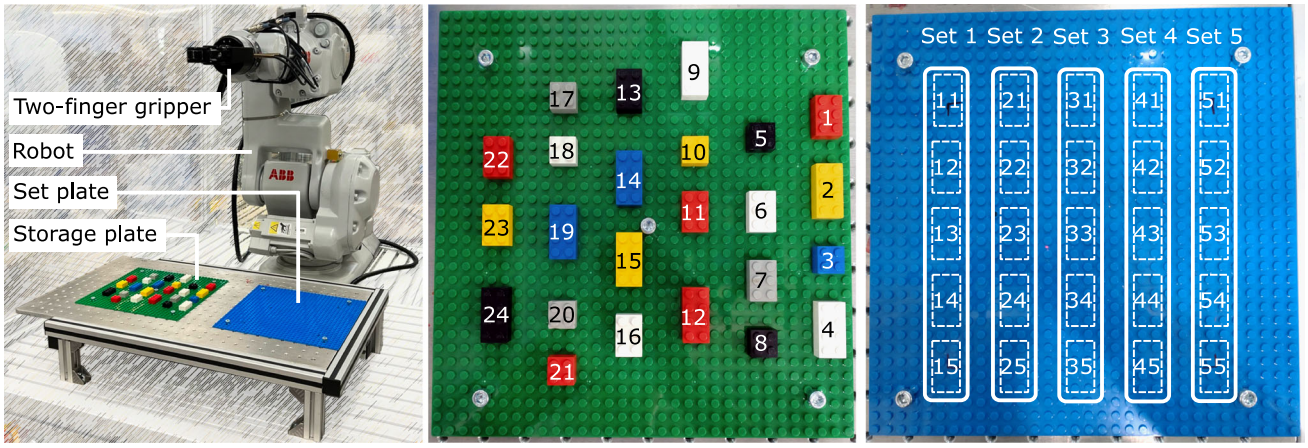


Fig. 14 Setup of the robot cell (**a**) and detailed view of the storage plate (**b**) and set plate (**c**). The storage plate contains 24 plate poses filled with 24 bricks initially. The set plate has 25 plate poses for five sets composed of a maximum of five bricks

tool, and described the initial state known by the robot in its internal knowledge base. Fourth, the robot completed five exemplary missions. Whereas we only planned missions in the first reference scenario, the robot also executed these this time. For this, we only had to specify the missions, and the robot itself constructed the planning problem based on its internal knowledge during the task planning process.

Application case 3: composition of diverse LEGO bricks

In this third application case, the robot in Fig. 14a picks up diverse parts with the *two-finger gripper* from a storage (left green plate; see Fig. 14b) and composes these in defined ordered sets on a set plate (right blue plate; see Fig. 14c) for further processing. Our range of parts consists of LEGO *bricks* of different sizes and colors. We call areas on the LEGO plates for stacking bricks *plate poses*. We predefined the plate poses for providing bricks in the storage and setting bricks on the set plate. To realize the application case, we implemented two specific skills for picking and placing bricks of variable size. In its internal knowledge base, the robot maintains the state of its two-finger gripper, the considered bricks, and defined plate poses. The initial state known by the robot relates to the state shown in Fig. 14. Based on this setup, we tasked the robot to complete five missions. In each mission, we specified 2–5 bricks available in the storage to be composed at defined plate poses in one set by the robot. In this way, the robot built the five sets of parts one after another.

Generated specific planning domain and planning results

We reviewed the results of this second reference scenario in a similar manner to the first one. The PDDLAutoGen tool correctly modified the abstract planning domain for this third application case. In comparison to the abstract planning domain, the resulting specific planning domain contains the additionally specified types two-finger gripper, brick, and plate pose, the automatically derived predicates for representing object input and output parameters, and two specific actions for picking and placing bricks.

The analysis of the specific planning domain that the PDDLAutoGen tool generated for this application case is also shown in Fig. 12. Here, the specific planning domain was successfully assembled with 31 modifications (19 modified lines and 12 added lines). In this way, we can confirm the reduction of the workload using the PDDLAutoGen tool compared to the manual creation of the specific planning domain also for this third application case. The number of modifications is fewer than for the previous two application cases. The specific planning domains of the first two application cases consist of five action definitions, and this third specific planning domain has only two action definitions. Due to the resulting smaller specific planning domain, fewer modifications had to be conducted.

The robot successfully planned and executed five missions based on the generated specific planning domain. Each mission contained between 2 and 5 goals. The resulting action plans consisted of an alternating sequence of actions for picking and placing bricks. Thus, during the execution of the plans, the robot picked the requested bricks from the storage plate and composed these into the defined sets on the set plate.

Additionally, for the second reference scenario, we compare the manual effort for specifying a task as an action plan versus a mission based on Fig. 15. As introduced in the beginning, the potential of integrating autonomous task planning into industrial robotics is to lift the abstraction level

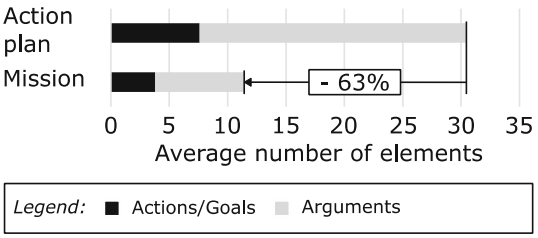


Fig. 15 Comparison of manual effort for workers to specify a task as an action plan or as a mission

for specifying tasks from the action plan to the mission level. To approximate the effort for specifying a task at these two levels, we added up the number of actions or goals and subordinate arguments to specify a task. The figure shows the average values for the five missions created and the action plans to fulfill them. By way of the example of the third application case, the average effort required to specify the task at the mission level is 63% less than at the action plan level in terms of the amount of information to be stated. At this point, we would like to mention that this consideration does not replace a full effort analysis from setting up the system to its use in operation. Nevertheless, in this way, we can confirm the potential of autonomous task planning by industrial robots, which the state of the art shows.

Discussion

Based on the presented results in the previous section, we start discussing our work by evaluating our technical objectives from "Objectives" section. Afterward, we highlight the innovation of the presented work, point out the resulting benefits, and discuss promising application fields for our approach. Finally, we consider the current limitations and challenges and give recommendations for future research.

Evaluation of technical objectives

For the discussion of our results, we apply a bottom-up approach and start with the evaluation of our technical sub-objectives:

E1 *Modeling of abstract planning knowledge:* We designed one abstract planning domain for the application field of kitting that we could later automatically adapt for three specific application cases. The modeling of abstract planning domains in PDDL has to be carried out by an expert in automated planning. However, this expert is only required once at the beginning to formalize the domain knowledge for an application field.

E2 *Expansion of abstract planning knowledge by relevant information from specific skill instances:* During the realization of the three application cases, we designed eleven specific skills based on the template of the three abstract skills *drive*, *pick* and *place*. All information relevant to the later generation of specific planning domains was extracted during this process. We aligned the procedure for initializing and describing specific skills within the PDDLAutoGen tool to our previous guideline for implementing skills based on our skills-based robotic software framework (Heuss et al., 2022). In the resulting procedure, skill developers do not require specialized knowledge in automated planning or PDDL, and no extra workload arises for integrating a specific skill into the PDDLAutoGen tool compared to describing the skill during implementation.

E3 *Automated generation of planning domains for specific application cases:* With the help of the PDDLAutoGen tool, we generated specific planning domains for three application cases. We used two different robot systems with a combination of diverse specific skills. The specific skill for driving was reused across two application cases. The specific skill instances of the pre-defined abstract skills allow the handling of different objects and require individual input parameters. As input information for the PDDLAutoGen tool, we only had to choose the considered application field, name the specific application case and select the specific skills required by the robot to operate in the intended application case. Thus, using the PDDLAutoGen tool, system integrators can use the information to be collected anyway when setting up a new application case and do not require detailed knowledge in another topic that goes beyond the application case. By analyzing the adapted specific planning domains of the three application cases, we were able to show that our approach reduces the workload for creating the specific planning domains enormously compared to the manual procedure.

E4 *Enabling industrial robots for autonomous task planning:* Finally, we planned five exemplary missions per application case using the specific planning domains created by the PDDLAutoGen tool in combination with an automated planning system available in the state of the art. All exemplary planning problems were solved. In this way, we demonstrated that we could enable industrial robots for autonomous task planning using the previously automatically generated specific planning domains. In addition, in the third application case, we build a holistic robot application based on a real robot cell and our skills-based control framework from previous works. Based on this application case, we were able to demonstrate further the executability of the derived action plans and, thus, the usability of the PDDLAutoGen tool within skills-based robotics. In addition, we have shown the efficiency benefits of robots that

autonomously plan their tasks. In our example, the manual effort for task instruction based on the information to be specified was 63% less compared to specifying the entire sequence of actions.

Innovation, benefits, and promising application fields in manufacturing

This section highlights the innovation and resulting benefits from our contribution. The innovation of the presented work is to automatically adapt abstractly modeled planning domains for specific individual application cases. This has not been addressed in the state of the art so far (see "Evaluation and contribution" section). As a first contribution, we investigated this idea on a conceptual basis and transformed it into a structured methodology. The designed methodology has two main benefits compared to existing approaches in the state of the art. After an abstract planning domain has been initially modeled by an expert in automated planning (see E1), it can be easily and automatically adapted for individual specific application cases (see E3). Beneficially to related works, application experts do not require additional expertise to adjust the specific PDDL planning domain for their individual needs. In addition, through the automatic adaption of planning domains, the workload decreases drastically. As a second contribution, we integrated the methodology into the procedure for developing skills-based industrial robot applications. In doing so, we could demonstrate the following benefits. First, autonomous task planning for industrial robots can be easily set-up by incorporating the new methodology for generating PDDL planning domains for specific application cases (see E3). The necessary information for the automated generation of planning domains can be acquired during the established development procedure of robot skills (see E2). Second, industrial robots capable of autonomously planning their tasks during operation can be easily instructed for manifold tasks with much lower effort compared to classical programming approaches (see E4). To conclude, based on the presented work, the deployment possibilities of industrial robots will increase and gain flexibility. In the long term, integrating easy-to-use and flexible robotic solutions has positive economic and international implications for manufacturing companies. By improving production processes, supporting employees, and counteracting the loss of skilled workers, intelligent robot applications can increase manufacturers' productivity and help them maintain national and international competitiveness.

In the scope of this publication, we have considered kitting as an exemplary application field and demonstrated the feasibility of our approach for three application cases with a stationary and mobile robot. Our approach is especially suited for application fields that fulfill the following two criteria. First, the application field has high task variability, so it profits from robots that perform autonomous task planning. Second, within the application field exist manifold application cases with a common basic structure and sequence but which differ individually, for example, in their environment, objects considered, or process characteristics. In addition to kitting and based on our considerations in "Preliminary considerations" section, we see further application fields in handling and assembly as promising following extensions.

Limitations, challenges, and recommendations for future research

Finally, we discuss the limitations of our work and the raised challenges and provide recommendations for future work. We start with the discussion of aspects directly related to the use of automated planning for robot task planning. Firstly, abstract planning domains are only valid in a defined application field. Therefore, the validity range must be specified and considered when setting up a specific application case. The focus of the presented work was the automated adaption of abstract planning domains to specific application cases and was not modeling the best planning domain. However, the validity range of an abstract planning domain could be extended by improving the modeling of the abstract planning domain. In this way, our approach could also become wider applicable. Secondly, when analyzing the exemplarily illustrated action plan of the first application case in Fig. 13, we identified some robot behavior to be improved for a more efficient operation. For example, sometimes the robot executes two driving actions in a row, whereas it could directly drive to the final waypoint (e.g., actions 9 and 10). This could be addressed by improving the abstract planning domain and testing further automated planners and settings that take optimization criteria into account. In this context, we recommend the analysis of different automated planning systems in combination with the previously stated improvement of abstract planning domains for future work.

In the following recommendations, we now steer our focus more to the integration of our approach into overall skills-based industrial robot applications. Firstly, in the presented work, we had to make trade-offs between generalizability and individuality of planning domains and related robot skills. The introduced approach can potentially simplify the creation of planning domains for a wide range of comparable skills-based application cases. However, application cases with particular characteristics that prevent the automated generation of a specific planning domain will always exist. Here, manual modeling of the PDDL planning domain or another approach from related works may be more suitable. Second, we introduced the concept of automatically adapting abstract planning domains for specific

applications and demonstrated its feasibility in skills-based industrial robotics. A holistic evaluation of the approach with a user study is promising in the next step. This allows usability and acceptance of the approach to be tested and used for further improvements. Third, we demonstrated the use of the PDDLAutoGen tool in the application field kitting in this work. Building on these experiences, we see the use and testing of the PDDLAutoGen tool in further application fields as promising, for instance, assembly or machine loading. In this context, also the extension from one actor to scenarios with multiple actors might be interesting (e. g., like the cooperative task of the second application case). For this, abstract planning domains for these further application fields can be integrated into the database of the PDDLAutoGen tool. In this context, abstract object types and actions should be reused once defined. This way, related subordinate specific skills will be applicable across different application fields. Lastly, within our methodology, we aim to combine skills from different developers in an application case and generate a specific planning domain for task planning. In this context, we define a validity range for abstract planning domains that specific application cases must also meet. In the future, the integration of procedures for formally validating and verifying automatically adapted planning domains might be interesting further to increase the effectiveness and safety of the approach. Bezrucav and Corves (2022), for instance, have presented an interesting approach to this that could be built upon.

Conclusion and outlook

Industrial robots that autonomously plan and execute upcoming production tasks are considered promising for future transformable factories. By means of automated planning techniques from the field of artificial intelligence, robots can be enabled for autonomous task planning. In this context, we identified the manual modeling of the robot's working environment as a planning domain as challenging and time-consuming. To ease this process, we presented the PDDLAutoGen tool for automatically adapting abstractly modeled planning domains for individual application cases. Building on the potential of skills-based robots, we further demonstrated the integration and use of the PDDLAutoGen tool within their life cycle. The basic idea of skills-based systems is to provide functional system capabilities called skills as modular building blocks that can be combined as needed for implementing an individual application case. In doing so, a common approach in skills-based robotics is to offer robot skills via a central database. In this way, users can browse available skills and download the skills they require for implementing a specific application case with their robot system. In parallel and based on the set of selected skills, the PDDLAutoGen tool provides users with a specific planning domain for their individual application case that they can use to configure a task planner based on automated planning. This way, users without expertise in automated planning can easily enable a robot for autonomous task planning without much extra effort. By enabling the robot for autonomous task planning, the abstraction level for instructing the robot for manifold tasks increases, and in turn, the required time decreases. Finally, we demonstrated the use of the presented methodology within three application cases.

Based on these results and the derived recommendations for future research in the previous chapter, we plan to deepen our work on two aspects. First, we like to evaluate the use of the PDDLAutoGen tool within further application fields. Second, we want to investigate how to improve the task planning capabilities of the robot system by evolving the modeling of abstract planning domains in combination with testing different automated planners.

Funding Open Access funding enabled and organized by Projekt DEAL. The research was developed with the support of the Bavarian Research Foundation in the research network FORobotics (AZ-1225-16), the Franco-German Alliance for Factory Supported by the Federal Ministry of Education and Research (BMBF) as well as the Free State of Bavaria under the Excellence Strategy of the Federal Government and the Laender, and the French alliance for the industry of the future, in the context of the German–French Academy for the Industry of the Future of Institute Mines-Telecom (IMT) and Technical University Munich (TUM) in the research project RDS2-Production and the Bavarian State Ministry for Economic Affairs, Regional Development, and Energy in the research project KIRO (DIK-2105-0045//DIK0357/01). We thank them for their funding and all participating project partners for supporting these projects.

Data availability The data that support the findings of this publication are available from the corresponding author upon reasonable request.

Code availability The code that supports the findings of this publication is available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Berger, J., Colceriu, C., Blank, A., Franke, J., Härdtlein, C., Hellig, T., Heinrich, D., Heuss, L., Hiller, M., Krä, M., Leichtmann, B., Lottermoser, A., Lu, S., Nitsch, V., Reinhart, G., Riedl, M. Roder, S., Schäfer, K., Schilp, J, Vogt, L., & Zäh, M. F. (2021). Abschlussbericht: FORobotics—Mobile adhoc kooperierende Roboterteams. Retrieved April 15, 2023, from https://publica.fraunhofer.de/entities/publication/c5a9b2b6-7ba2-4f72-bb70-f992d853f240/details

Bezrucav, S.-O., & Corves, B. (2022). Modelling automated planning problems for teams of mobile manipulators in a generic industrial scenario. *Applied Sciences, 12*(5), 2319. https://doi.org/10.3390/app12052319

Bøgh, S., Nielsen, O. S., Pedersen, M. R., Krüger, V., & Madsen, O. (2012). Does your robot have skills? In *Proceedings of the 43rd international symposium on robotics*. VDE Verlag GMBH.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N., & Carreras, M. (2015). ROSPlan: Planning in the robot operating system. In *Proceedings of the 25th international conference on automated planning and scheduling* (Vol. 25, pp. 333–341). https://doi.org/10.1609/icaps.v25i1.13699

Crosby, M., Rovida, F., Pedersen, M. R., Petrick, R. P. A., & Krüger, V. (2016). In planning for robots with skills. In *Proceedings of the 4th workshop on planning and robotics* (pp. 49–57).

drag and bot GmbH. (2023). drag&bot website. Retrieved February 19, 2023, from https://www.dragandbot.com/de/

Edelkamp, S., & Hoffmann, J. (2004). PDDL2.2: The language for the classical part of the 4th international planning competition (Technical Report No. 195)

Fast Downward. (2023). Fast downward. The wiki of the Fast Downward planning system. Retrieved April 15, 2023, from https://www.fast-downward.org/

FRANKA EMIKA GmbH. (2023). FRANKA EMIKA website. Retrieved February 19, 2023, from https://www.franka.de/de

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). PDDL—The planning domain definition language: Version 1.2 (Tech Report CVC TG-98–003/DCS TR-1165).

Ghallab, M., Nau, D., & Traverso, P. (2016). *Automated planning and acting*. Cambridge University Press.

Green, A., Reji, B. J., Muise, C., Scala, E., Meneguzzi, F., Rico, F. M., Stairs, H., Dolejsi, J., Magnaguagno, M., & Mounty, J. (2023). Planning.Wiki—The AI planning & PDDL wiki. Retrieved July 7, 2023, from https://planning.wiki/

Hägele, M., Nilsson, K., Pires, J. N., & Bischoff, R. (2016). Industrial robotics. In B. Siciliano & O. Khatib (Eds.), *Springer handbook of robotics* (2nd ed., pp. 1385–1421). Springer.

Hammerstingl, V., & Reinhart, G. (2018). Skills in assembly. Retrieved April 15, 2023, from https://mediatum.ub.tum.de/1428286

Heuss, L., Blank, A., Dengler, S., Zikeli, G. L., Reinhart, G., & Franke, J. (2019). Modular robot software framework for the intelligent and flexible composition of its skills. In F. Ameri, K. E. Stecke, G. von Cieminski, & D. Kiritsis (Eds.), *IFIP advances in information and communication technology. Advances in production management systems. Production management for the factory of the future* (Vol. 566, pp. 248–256). Berlin: Springer.

Heuss, L., Gonnermann, C., & Reinhart, G. (2022). An extendable framework for intelligent and easily configurable skills-based industrial robot applications. *The International Journal of Advanced Manufacturing Technology, 120*(9–10), 6269–6285. https://doi.org/10.1007/s00170-022-09071-w

Heuss, L., Lux-Gruenberg, G., Hammerstingl, V., Schnös, F., Rinck, P., Reinhart, G., & Zäh, M. (2018). Mobile Autonome Roboter in der Smart Factory: Dynamische Planung und Adaption mobiler Roboter für die flexible Produktion. *Wt Werkstattstechnik Online, 108*(9), 574–579.

Heuss, L., & Reinhart, G. (2020). Integration of autonomous task planning into reconfigurable skill-based industrial robots. In *25th IEEE international conference on emerging technologies and factory automation* (pp. 1293–1296). https://doi.org/10.1109/etfa46521.2020.9212005

Hofmann, T., Niemueller, T., & Lakemeyer, G. (2017). Initial results on generating macro actions from a plan database for planning on autonomous mobile robots. In *Proceedings of the 27th international conference on automated planning and scheduling* (Vol. 27, pp. 498–503). https://doi.org/10.1609/icaps.v27i1.13868

Huckaby, J., Vassos, S., & Christensen, H. I. (2013). Planning with a task modeling framework in manufacturing robotics. In *2013 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5787–5794). https://doi.org/10.1109/IROS.2013.6697194

International Federation of Robotics. (2020). How connected robots are transforming manufacturing. Retrieved February 10, 2023, from https://ifr.org/papers

International Federation of Robotics. (2022). Market presentation of World Robotics 2022. Retrieved February 19, 2023, from https://ifr.org/downloads/press2018/2022_WR_extended_version.pdf

Kootbally, Z. (2016). Industrial robot capability models for agile manufacturing. *Industrial Robot: An International Journal, 43*(5), 481–494. https://doi.org/10.1108/IR-02-2016-0071

Kootbally, Z., Schlenoff, C., Lawler, C., Kramer, T., & Gupta, S. K. (2015). Towards robust assembly with knowledge representation for the planning domain definition language (PDDL). *Robotics and Computer-Integrated Manufacturing, 33*, 42–55. https://doi.org/10.1016/j.rcim.2014.08.006

Koren, Y., Gu, X., & Guo, W. (2018). Reconfigurable manufacturing systems: Principles, design, and future trends. *Frontiers of Mechanical Engineering, 13*(2), 121–136. https://doi.org/10.1007/s11465-018-0483-0

Lamy, J.-B. (2017). Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine, 80*, 11–28. https://doi.org/10.1016/j.artmed.2017.07.002

Liang, Y. S., Pellier, D., Fiorino, H., & Pesty, S. (2019). End-user programming of low-and high-level actions for robotic task planning. In *2019 28th IEEE international conference on robot and human interactive communication* (pp. 1–8). https://doi.org/10.1109/RO-MAN46459.2019.8956327

Marrella, A., Mecella, M., & Sardiña, S. (2018). Supporting adaptiveness of cyber-physical processes through action-based formalisms. *AI Communications, 31*(1), 47–74. https://doi.org/10.3233/AIC-170748

Ortiz, J., García-Olaya, A., & Borrajo, D. (2013). Using activity recognition for building planning action models. *International Journal of Distributed Sensor Networks*. https://doi.org/10.1155/2013/942347

Pane, Y., Mokhtari, V., Aertbelien, E., De Schutter, J., & Decre, W. (2021). Autonomous runtime composition of sensor-based skills using concurrent task planning. *IEEE Robotics and Automation Letters, 6*(4), 6481–6488. https://doi.org/10.1109/LRA.2021.3094498

Pantano, M., Eiband, T., & Lee, D. (2022). Capability-based frameworks for industrial robot skills: a survey. In *2022 IEEE 18th international conference on automation science and engineering* (pp. 2355–2362). https://doi.org/10.1109/CASE49997.2022.9926648

Paxton, C., Hundt, A., Jonathan, F., Guerin, K., & Hager, G. D. (2016). CoSTAR: Instructing collaborative robots with behavior trees and vision. In *2017 IEEE international conference on robotics and*

automation (pp. 564–571). https://doi.org/10.1109/ICRA.2017.7989070

Pedersen, M. R., Nalpantidis, L., Andersen, R. S., Schou, C., Bøgh, S., Krüger, V., & Madsen, O. (2016). Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing, 37*, 282–291. https://doi.org/10.1016/j.rcim.2015.04.002

Peichl, A., Sauer, S., & Wohlrabe, K. (2022). Fachkräftemangel in Deutschland und Europa—Historie, status quo und was getan werden muss. *Ifo Schnelldienst, 10*, 70–75.

Rovida, F., Crosby, M., Holz, D., Polydoros, A., Großmann, B., Petrick, P. A., & Krüger, V. (2017). SkiROS—A skill-based robot control platform on top of ROS. In A. Koubaa (Ed.), *Robot operating system (ROS). Studies in computational intelligence* (pp. 121–160). Springer.

Schäfer, P. M., Steinmetz, F., Schneyer, S., Bachmann, T., Eiband, T., Lay, F. S., Padalkar, A., Sürig, C., Stulp, F., & Nottensteiner, K. (2021). Flexible robotic assembly based on ontological representation of tasks, skills, and resources. In E. Erdem, M. Bienvenu, & G. Lakemeyer (Eds.), *Proceedings of the eighteenth international conference on principles of knowledge representation and reasoning* (pp. 702–706). International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/kr.2021/73

Stenmark, M., & Malec, J. (2015). Knowledge-based instruction of manipulation tasks for industrial robotics. *Robotics and Computer-Integrated Manufacturing, 33*, 56–67. https://doi.org/10.1016/j.rcim.2014.07.004

Thomas, U., Hirzinger, G., Rumpe, B., Schulze, C., & Wortmann, A. (2013). A new skill based robot programming language using UML/P Statecharts. In *2013 IEEE international conference on robotics and automation* (pp. 461–466). https://doi.org/10.1109/ICRA.2013.6630615

Vaquero, T. S., Silva, J. R., Tonidandel, F., & Christopher Beck, J. (2013). itSIMPLE: Towards an integrated design system for real planning applications. *The Knowledge Engineering Review, 28*(2), 215–230. https://doi.org/10.1017/S0269888912000434

Wenger, M., Eisenmenger, W., Neugschwandtner, G., Schneider, B., & Zoitl, A. (2016). A model based engineering tool for ROS component compositioning, configuration and generation of deployment information. In *2016 IEEE 21st international conference on emerging technologies and factory automation* (pp. 1–8). https://doi.org/10.1109/ETFA.2016.7733559