Intuitive Instruction of Industrial Robots: Semantic Process Descriptions for Small Lot Production

Alexander Perzylo*, Nikhil Somani†, Stefan Profanter*, Ingmar Kessler*, Markus Rickert*, Alois Knoll†

***Abstract*— In this paper, we introduce a novel robot programming paradigm. It focuses on reducing the required expertise in robotics to a level that allows shop floor workers to use robots in their application domain without the need of extensive training.**

Our approach is user-centric and can interpret underspecified robot tasks, enabling communication on an abstract level. Such high-level task descriptions make the system amenable for users that are experts in a particular domain, but have limited knowledge about robotics and are thus not able to specify low-level details and instructions. Semantic models for all involved entities, i.e., processes, workpieces, and workcells, enable automatic reasoning about underspecified tasks and missing pieces of information.

We showcase and evaluate this methodology on two industrial use cases from the domains of assembly and woodworking, comparing it to state-of-the-art solutions provided by robot manufacturers.

I. Introduction

After a long fordistic period of industrial production, there has been a trend in some parts of today's industry toward individualized products. Additionally, robot-based industrial automation increasingly diffuses into small and medium-sized enterprises (SMEs). In both cases, the industry has to adapt their production processes for small lot sizes and a high number of product variants. Hence, they require their robot systems to allow for rapid changeovers and efficient teaching. In this context, commercial viability of automated production is highly influenced by the time required to teach new processes and to adapt existing processes to variations of a product. However, most SMEs cannot build the necessary expertise in robotics in-house and have to rely on system integrators. This drastically reduces the usability of robot systems for small batch assembly. As a result, SMEs represent a market segment which has not been properly penetrated by robot manufacturers, yet.

Classical teaching concepts for robot systems offer dedicated robot programming languages, which require a significant amount of training and cannot be intuitively used [1]. Human operators have to understand and use concepts like Euler angles and work with raw Cartesian coordinates, e.g., in order to define a grasp pose. An important shortcoming of those programming languages is the lack of semantic meaning. For instance, from looking at such robot programs it is not possible to see what kind of object is being manipulated or to easily keep track of the structure of the overall process.

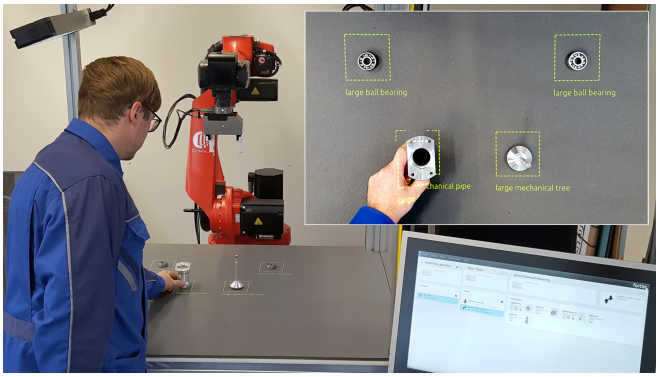


Fig. 1: Cognitive robotic workcell with features such as object recognition, projector-based highlighting, hand and body gesture recognition, and touch-based graphical interface for the task-level instruction of the robot.

A more intuitive teaching paradigm is teaching through manual guidance. Based on physical interaction of the operator with the robot, robots can be quickly moved and the resulting poses can be stored and re-used. It spares the human operator the cumbersome effort of teaching positions through jogging the robot via a teach pendant. Obviously, this approach is only feasible for small robots. Besides, the resulting robot program still does not know anything about the operator's intention.

Industrial solutions such as *Delmia*, *Process Simulate*, or *ABB RobotStudio* offer sophisticated robot programming environments that scale to a full production line with multiple robots. Due to their high complexity, it is very difficult for untrained shop floor workers to make adequate use of them [2]. The operators typically have to train for several weeks to learn how to use these interfaces. For instance, transfering an already implemented process from one workcell to a different one requires significant reprogramming effort.

In contrast to this, research—particularly in the domain of service robotics—is being carried out to develop fully autonomous cognitive robotic systems that are able to translate high-level task descriptions to appropriate robot actions. These systems are able to perceive their environment and to reason about the implications of their tasks. Many of them provide natural language interfaces, e.g., by making use of task descriptions from wikis or by interpreting speech input from the operator. However, these systems have not had a real impact on industrial applications yet, due to the early

*Alexander Perzylo, Stefan Profanter, Ingmar Kessler, and Markus Rickert are with fortiss GmbH, Guerickestr. 25, 80805 München, Germany. {perzylo|profanter|kessler|rickert}@fortiss.org

†Nikhil Somani and Alois Knoll are with the Department of Informatics VI, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany. {somani|knoll}@in.tum.de

stage of development and their disruptive nature compared to the established approaches.

In this paper, we introduce a concept tailored to intuitive teaching of tasks for industrial robotic workcells (Fig. 1) by using proven techniques from the field of cognitive robotics. The operator is still in charge of controlling most aspects, whereas the cognitive capabilities of the robot system are used to increase the efficiency in human-robot communication. Compared to classical approaches, it requires less knowledge about the system. As a result, the required time to train a human worker can be significantly reduced. The system may use previously modeled knowledge about certain industrial domains, processes, interaction objects, and the workcell itself. Hence, the communication between the operator and the robot system can be lifted to a more abstract and semantically meaningful level, where the operator can talk about, e.g., which object to pick instead of specifying raw coordinates.

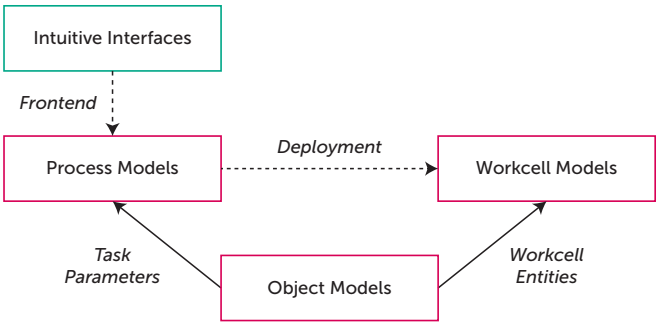


Fig. 2: Overview of semantic models and interrelations. Object models may serve as parameters for certain tasks specified in process models. A workcell model relies on object models for describing its entities. In order to execute process models, they are deployed on workcells using the information in their workcell models. Intuitive interfaces serve as a frontend to the semantic process models.

II. Related Work

The strive for an easier, intuitive, and more automated teaching process for robots dates back more than 40 years now. In 1972 a prototype was developed that was able to assemble simple objects from plan drawings using a vision system [3]. Another early approach defined constraints between two objects using planar and cylindrical matching and developed an offline object-level language for programming robot assembly tasks [4], [5]. RALPH [6] is a similar system which uses information from CAD drawings to automatically generate a process plan for the manipulator to perform the task. A hierarchical graph-based planning algorithm for automatic CAD-directed assembly is described in [7]. All these systems have in common that they are not human-centric, i.e., the robot program is generated automatically giving a human worker no option to adjust or modify the proposed assembly. They also require accurate models to perform automatic plan generation. Additionally, the drawings or CAD models do not include any semantic annotation to enable further reasoning about its properties, e.g., if a cylindrical part is the thread of a screw, which needs to be thightened in order to install it.

Pardo-Castellote [8] described a system for programming a dual-arm robot system through a high-level graphical user interface. Despite being one of the more advanced systems at the time of publishing, it mainly supported pick and place and hand-over actions, but it did not allow, e.g., in-air assembly or peg-in-hole tasks.

In recent years, there have been research efforts toward teaching robot programs by observing human demonstration [9], [10], [11]. They rely on a predefined set of speech commands and demonstrating actions or require various different sensors to detect human motions. Additional effort is required to teach human workers how to demonstrate tasks in a compatible way. Sensor requirements reduce the mobility of such systems since all sensors need to be set up and calibrated in each working area.

A comprehensive overview of programming methods for industrial robots, including online programming (operator assisted and sensor guided), offline programming (based on CAD data), and augmented reality is presented in [2]. Current research increasingly focuses on task-level programming that requires a library of predefined lower-level building blocks, called skills [12], [13], [14].

With the rapid growth of the World Wide Web and the widespread availability of knowledge, Knowledge Bases gained importance for supporting the design and implementation of reconfigurable manufacturing systems [15]. RoboEarth [16] and RoboHow [17] developed knowledge-based software frameworks that use semantic description languages to encode and share knowledge between different robots.

The IEEE-RAS Ontologies for Robotics and Automation (ORA) working group has been working toward defining standardized ontologies including a common upper ontology and specialized complementary ontologies covering the needs of particular domains, e.g., service robotics or industrial robotics [18]. As of now, the industrial robotics group mainly investigated the application of autonomous kit building and further work on other and more complex types of tasks still needs to be carried out [19], [20].

This paper is an updated and extended version of a workshop paper presented at the RSS conference in 2015 [21].

III. An Intuitive Teaching Paradigm

Current industrial robotic systems require the user to be an expert not only in the application domain but also at robot programming. The key motivation in our approach is to substantially reduce the level of robotics expertise required to use such systems to a level where a shop floor worker with minimal knowledge about robotics can instruct, interact with, and operate the system. In this programming paradigm, the robotics and domain specific knowledge is modeled explicitly (Fig. 2) and the system needs to be able to understand, interpret, and reason about it. We have chosen the Web Ontology Language (OWL) for this knowledge

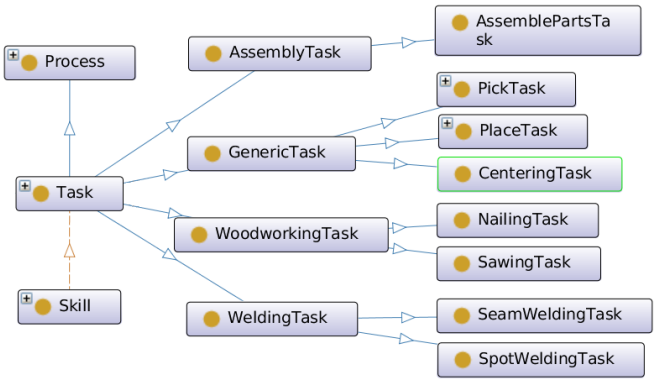


Fig. 3: Excerpt of task taxonomy. An abstract task description might have multiple actual implementations, i.e., robot or tool skills.

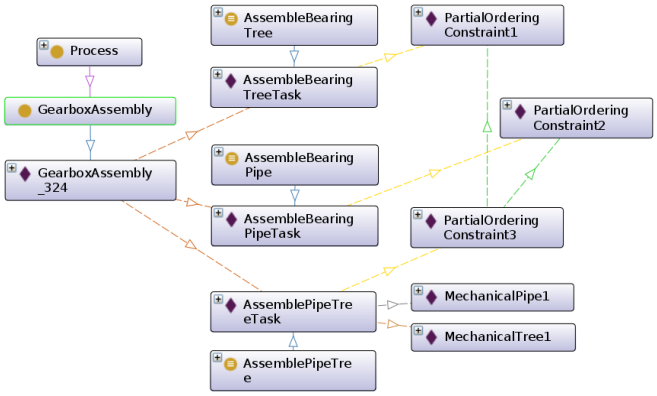


Fig. 4: Process model for the 3-step assembly of the core part of a gearbox. Boxes featuring a *yellow* circle or a *purple* rhombus represent classes and instances of those classes respectively.

representation, primarily because OWL is based on a formal specification, i.e., a description logic, which facilitates logical inference. Another advantage is the ease with which additional knowledge (e.g., new workpieces) can be added to the system. This facilitates the separation of knowledge and code, thus enabling addition of information without changing the implementation.

A key concept in this design is programming at object level. In this approach, the user specifies robot tasks in terms of the objects involved and the relevant parameters. This is in contrast to the traditional teach pendant-based approaches where tasks are specified in terms of raw coordinate frames. While there exist some approaches that involve task specification in terms of the coordinate frames attached to an object [22], they are restricted to coordinate frames only.

In several manufacturing domains, especially assembly, products are designed by domain experts using specialized CAD software. In this process, the information and ideas that the product designer had in mind while designing the product are lost and only the finished CAD model of the product is sent to manufacturing. In contrast to this, we aim to model, store, and exploit this information in order to ease the programming of the manufacturing process. The first step in this direction—most suitable for assembly tasks—is to include in the product description not only the final CAD model, but also the semantically relevant geometric entities in it, the constraints between these geometric entities, and the individual assembly steps that the designer took in designing the final product.

A. Semantic Process Models

Semantic process models are descriptions of the steps required for manufacturing a product, built by arranging tasks that have been hierarchically defined in a potentially constrained order. Each of these tasks is an object-level description of individual steps in the process, which might be underspecified and have pre-post-conditions. Due to this abstract description, they can be understood and executed not only by a machine but also by a human operator. Robotic systems provide corresponding low-level implementations

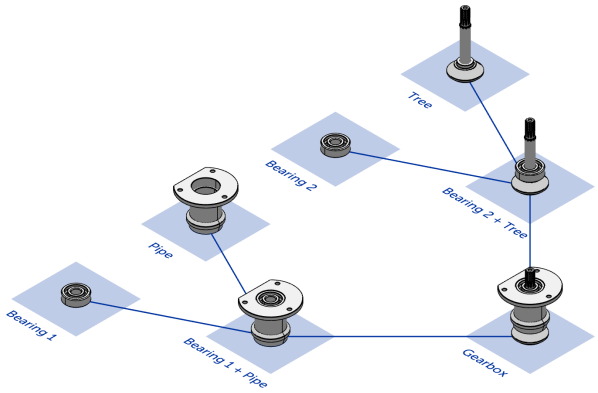


Fig. 5: Industrial use case based on assembling four work pieces in three steps to form the core part of a gearbox

(called *skills*) that require a complete parametrization with actual numerical values.

We have implemented a taxonomy of industrial task types to deal with different industrial domains, e.g., woodworking, welding, and assembly (Fig. 3). Every type of task has a certain set of parameters that are required to be set by the user during the Teach-In phase. There are optional parameters which can either be specified by the operator or automatically inferred by the system upon deployment.

Fig. 4 visualizes an excerpt of a semantic process model that describes the process of assembling the core part of a gearbox as shown in Fig. 5. It shows the class level concept *GearboxAssembly* and one arbitrary instantiation *GearboxAssembly_324*. Every attempt at executing the assembly would result in an additional instantiation of the class level description of the task. This allows to log all relevant information collected during execution, e.g., in case of an anomaly we can determine which task in which process instance failed, when it happened, and due to what error. The *GearboxAssembly* process consists of three *Assembly* tasks, i.e., *AssembleBearingTreeTask*, *AssembleBearingPipeTask*, and *AssemblePipeTreeTask*.

The order of these tasks is not yet fully specified. Instead,

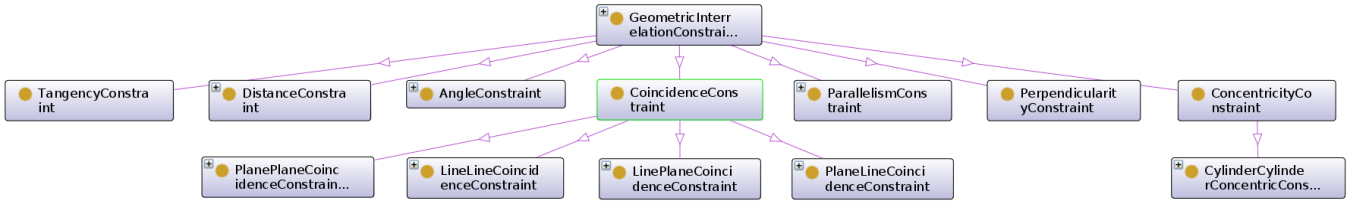


Fig. 6: Upper taxonomy of geometric interrelation constraints

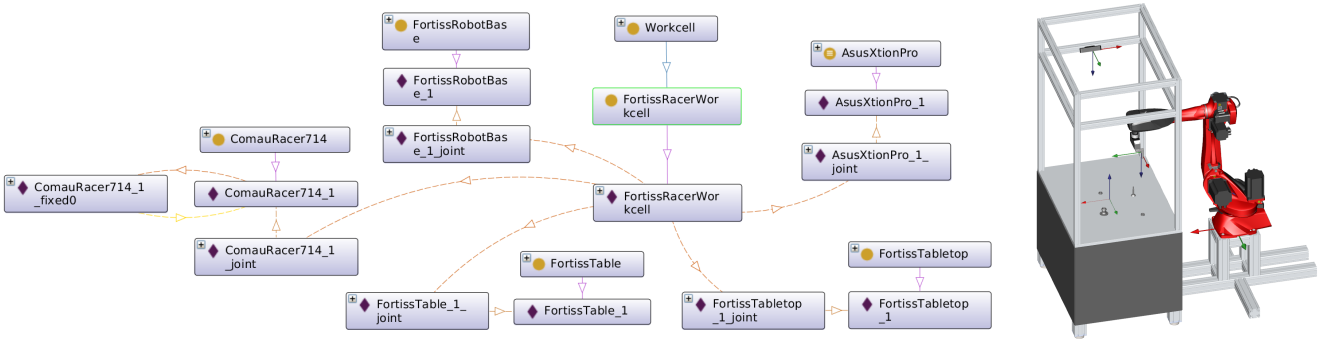


Fig. 7: Excerpt of a workcell model and a visualization of the contained information

three partial ordering constraints have been specified, which assert that the task instance associated with *PartialOrderingConstraint**3*** has to succeed the tasks associated with *PartialOrderingConstraint**2*** and *PartialOrderingConstraint**1***. No further restrictions have been modeled, which results in the order of the two tasks *AssembleBearingTreeTask* and *AssembleBearingPipeTask* not being constrained. The *AssembleBearingPipeTask* links two interaction objects *MechanicalPipe1* and *MechanicalTree1* as its parameters. The geometric constraints specifying the assembly pose have been defined on a sub-object level between single faces of the two object models as explained in Section III-B.2.

B. Semantic Object Models

Object models form one of the key pillars of our proposed robot programming approach. They are modeled in a hierarchical fashion, wherein properties of generic object types can be re-used in the more specific ones. Process descriptions refer to objects and their properties as parameters for their robot tasks. The requirements of a task can help to filter suitable types of objects.

Simple object properties include information such as the name, weight, pose, or material. Additional information, such as specialized appearance models that can be used by computer vision modules to detect the objects in a workcell, can also be linked to an object model. Basic geometric properties include the bounding box, the corresponding dimensions, and the polygon mesh used for rendering the object.

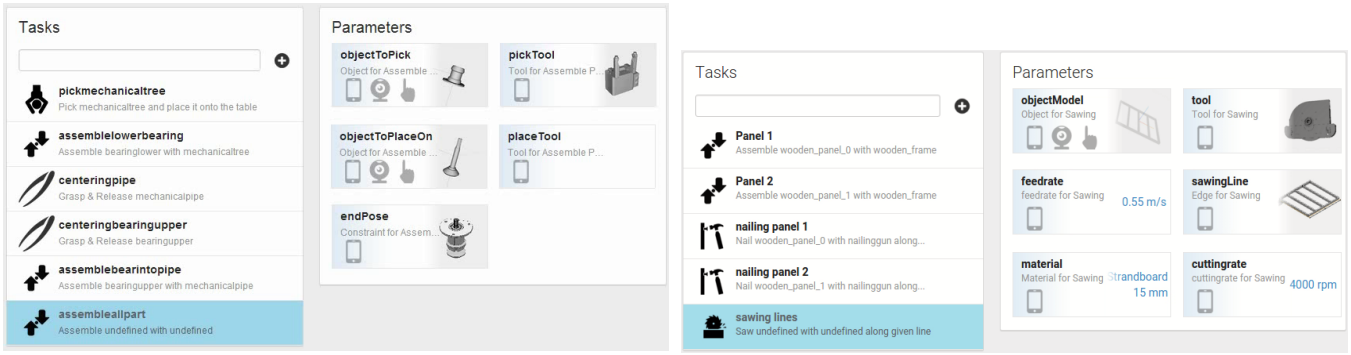
In contrast to only representing approximated geometric information, we aim to preserve all relevant information produced while designing an object. This includes the CAD model used for generating polygon meshes. We support geometric representations at multiple levels of detail, from points and coordinate frames, to semantically meaningful entities such as lines and circles or planes and cylindrical surfaces. Geometric constraints between these entities are used in our system to describe parameters for robot tasks in an intuitive way.

1) Boundary Representation of Objects: A Boundary Representation (BREP) of CAD data describes the geometric properties of points, curves, surfaces, and volumes using exact mathematical models as its basis. CAD models are created by defining boundary limits to given base geometries. The BREP specification distinguishes between geometric and topological entities. Geometric entities hold the numerical data, while topological entities group them and arrange them in a hierarchical fashion. We use a BREP-based semantic modeling of object geometries, so that additional information can be attached to objects not only at a frame-level but at any meaningful subpart, e.g., individual faces or edges [23].

Object recognition can also benefit from exploiting the BREP information of an object by matching contained shape primitives [24].

2) Geometric Constraints between Objects: Given the rich semantic description of CAD models, it is possible to refer to arbitrary parts of it and to link them with additional information. Fig. 6 shows the upper taxonomy of the geometric constraints that have been designed to represent assembly constraints. Those constraints are meant to be specified between points, curves, and surfaces of objects.

In our representation a geometric constraint refers to two geometric entities: A base entity with a defined pose and a constrained entity whose pose depends on the base entity and the constraint itself [23].



(a) Process description for the assembly of a gearbox

(b) Process description for the manufacturing of a wooden wall

Fig. 8: Snippets of the relevant subpart of the teaching interface that acts as a frontend to the semantic process descriptions

C. Semantic Workcell Models

A workcell model describes the physical setup of the workcell, including robots, tools, sensors, and available skills.

For instance, the cognitive robotic workcell depicted in Fig. 1 contains a robot arm, a work table, and an RGBD sensor. Fig. 7 shows an excerpt of the corresponding semantic workcell description. It asserts the *Workcell* instance called *FortissRacerWorkcell* that links to its contained entities. These links are represented by *FixedJoint* instances. In order to specify the poses of the sensor, table, and robot base with respect to the workcell origin, the *FixedJoint*s further refer to instances of *RigidTransformationMatrix*. The model can be augmented with constraints that shall be respected during the execution of tasks, e.g. workspace constraints or velocity constraints that may be defined for certain regions of the overall workspace.

D. Teaching Interface

Having to manipulate the semantic process descriptions manually would only shift the required expert knowledge for using the robot system from the domain of robotics to the domain of knowledge engineering. Therefore, we include a graphical user interface for the human operator based on HTML5 and JavaScript, that abstracts from the semantic modeling language and can run in any modern web browser.

Fig. 8a depicts the main view of the GUI showing the task sequence of a process consisting of six tasks and the parameters of the selected task. Parameters *objectToPick* and *objectToPlaceOn* can be set by selecting the desired objects from a list or by using other modalities [25], which have been evaluated in a separate user study [26]. *pickTool* and *placeTool* are optional parameters, which might be used to specify compatible tools for grasping the two objects involved in the assembly. The *endPose* parameter defines the assembly pose and can be set in a dedicated 3D interface. In the example shown in Fig. 9, two cylindrical surfaces have been selected and a concentricity constraint has been specified.

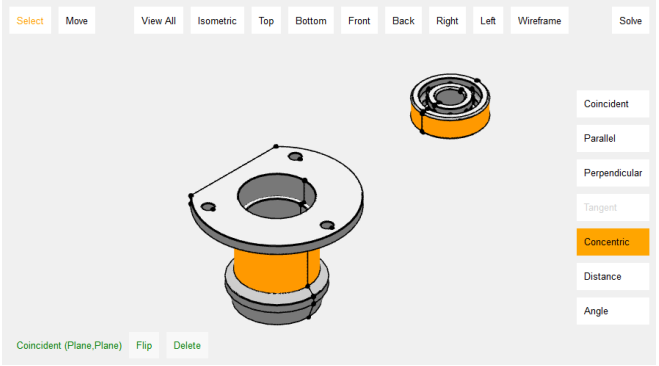


Fig. 9: Mode of the intuitive teaching interface for defining geometric constraints between two objects. As the two objects are described in a boundary representation, each of the two highlighted cylindrical surfaces can be selected with a single click. Geometric constraints can be chosen from the list on the right. A *PlanePlaneCoincidenceConstraint* (Fig. 6) has already been specified and is visualized in the bottom left corner of the GUI.

IV. Execution Framework

The process plan designed using the intuitive teaching interface is an underspecified, hardware independent description of the robot's tasks. In order to execute the process plan on a particular workcell, the tasks need to be fully specified and mapped to executable actions for the robot. As an example, the tool for grasping an assembly object is an optional parameter in the task specification interface. The appropriate tool can be automatically selected by matching the set of available tools in the workcell to the ones suitable for manipulating the object. The system can reason if this mapping is even possible for a specific task on a specific workcell and provide the user with appropriate feedback. This robot control framework distinguishes task constraints from environment and robot model constraints. Hence, the same set of task constraints can be instantiated on different workcells. A world model component maintains the state of all involved entities in the world, including instances of objects, tools, robots, etc. In our experimental setup, poses of

detected objects in the workcell are obtained from the vision system and updated in the world model.

When an assembly task has been completely specified, the contained constraints are solved using our constraint-based robot control framework [27], and mapped to a set of primitive actions (e.g., move, open/close gripper commands) that are offered by the robot's control interface. Since robot tasks are not always completely defined in the geometric sense, such a controller can exploit the nullspace of task constraints to satisfy secondary objectives, such as environment constraints or posture optimization [28].

Once the task execution begins, monitoring can be done at different levels of abstraction. The individual actions (e.g., open/close gripper) might fail, or the task as a whole might not be possible due to unmet preconditions (e.g., assembly object missing). In either case, the error message that is generated contains semantically meaningful information for the user.

V. Evaluation

A. Quantitative Evaluation

In order to evaluate the efficiency of our teaching concept, we conducted a preliminary study with one test person for each of two use cases. The robotic tasks used in the experiments are taken from the domains of industrial assembly and woodworking.

The first use case deals with the partial assembly of a gearbox using an industrial dual-arm robot system. The gearbox comprises four parts which have to be assembled in three steps (Fig. 5). Each assembly step requires sub-millimeter precision, making it an ideal use case for a precise industrial robot.

The second use case involves a large linear gantry robot for manufacturing a wall of a wooden house. The task requires picking and placing two wooden panels on top of a pre-assembled frame. The panels are fixed to the frame using a nailing gun and the excess overhang is cut off using a circular saw (Fig. 10). The different tools are made available to the robot through a tool changer.

In the experiments, the subjects first followed the classical approach of programming the task in a robot programming language using a combination of teach pendant and PC. Afterwards, they taught the same task using a touch screen showing the graphical user interface of our cognitive teaching framework. This comparison is biased against our system as the test persons had several years of experience in using teach pendants and robot programming languages, but only received a 10-minute introduction to the intuitive framework right before they used it.

Videos covering the mentioned use cases and demonstrating our intuitive teaching framework and the results of this evaluation can be found online [1] [2].

[1]Gearbox assembly use case: https://youtu.be/B1Qu8Mt3WtQ
[2]Woodworking use case: https://youtu.be/bbInEMEF5zU

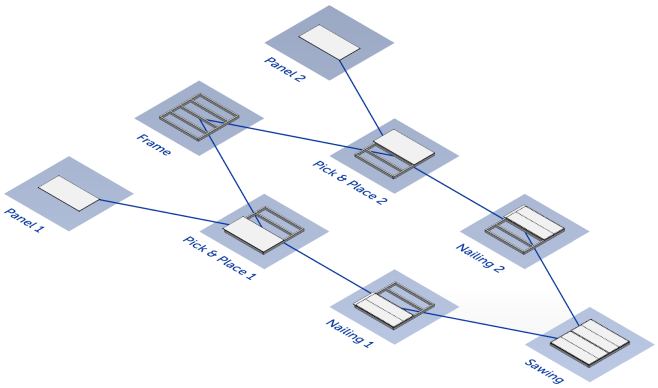


Fig. 10: Industrial use case based on assembling a frame and two wooden panels in five steps to build a wall of a wooden house. Different types of tasks are involved, i.e., pick and place, nailing, and sawing.

1) Classical Teaching: The programs for performing the two tasks are fairly complex, involving numerous robot movement commands and, in case of the gearbox assembly, synchronization between the two robot arms. For the woodworking use case numerous tool changing operations were required. Hence, the skeletons for the programs were created on a PC and the specific robot poses for each movement were obtained by jogging the robots using a teach pendant (Fig. 11a and Fig. 11c). Given the precision required, the parts needed to be placed into fixtures and the robot poses fine-tuned accordingly.

2) Intuitive Teaching: The resulting processes created using the intuitive teaching interface (Fig. 11b and Fig. 11d) are shown in Fig. 8a and Fig. 8b respectively.

The gearbox assembly process consists of six steps, of which three are assembly tasks parametrized using the BREP mating interface. We employ a CAD model-based vision system [29] using a Kinect-like camera placed on top of the table (at a distance of 1 m) to recognize objects, providing a positioning accuracy of approximately 1 cm. There are two centering steps that use the imprecise object poses obtained from the vision system and center the parts between the gripper fingers of a parallel gripper, thereby reducing their pose uncertainties.

The process plan for the manufacturing of the wooden wall involves two pick and place operations for putting two panels on top of a wooden frame. The BREP mating interface was used to define the corresponding assembly poses. Tool changes could be automatically inferred based on the type of operations that were selected by the user. The nailing and sawing lines were specified through a specific mode of the BREP mating interface, in which the tools could be aligned with respect to the panels. For this use case, no vision system was used and the parts were placed into fixtures.

3) Results of Preliminary Study: The experiments were used to investigate the times required to accomplish various programming tasks. For the gearbox use case, this included the times for teaching the assembly process from scratch, adjusting the object poses in the existing program, and

(a) (b) (c) (d)

Fig. 11: Toward efficient robot programming: moving from (a,c) a classical approach using the teach pendant using low-level programming languages to (b,d) efficient robot programming using high-level semantic process descriptions.

TABLE I: Required times for accomplishing various programming tasks for the classical and the intuitive approach.

Task	**Classical**	**Intuitive**	
	time in min	time in min	saved time in %
Gearbox			
Full assembly	48	8	83
Adj. object poses	23	0	100
Adj. approach poses	5	2	60
Wooden Wall			
Full assembly	47	13	72
Adj. task sequence	1.6	0.2	87

adjusting the robots' approach poses for the objects. The woodworking use case was evaluated with respect to the times required to teach the full process, and to change the order of the placing and nailing tasks for the two panels.

Comparing the times required to teach the processes using the two different methods shows a time-saving of 83 % for the intuitive teaching approach in case of the gearbox use case and 72 % for the woodworking use case (Table I).

Updating the gearbox assembly robot program to reflect changes in the work pieces' positions took 23 min for the classical teaching approach, whereas the intuitive teaching framework's vision system naturally coped with the changes automatically. When the 23 min for the teaching of the object poses are neglected, the intuitive approach still is more than three times faster. Adjusting the workpieces' approach poses was accomplished in 5 min on the teach pendant. Using the intuitive teaching interface it took 2 min. While the programmer decided to use precise positioning by jogging the robot using the classical approach, this option was not available to our intuitive approach. With a more precise vision system, the additional centering tasks would not be necessary. This centering technique could have also been used in the classical approach. As can be seen from the second comparison (Table I), fine tuning object poses takes much more time than programming centering tasks.

For reordering the placing and nailing tasks, the human operator spent 1.6 min using the classical approach and 0.2 min using the intuitive approach.

B. Qualitative Evaluation

Exploiting the logical formalism behind our semantic description language, the teaching framework can interpret and automatically augment underspecified process descriptions.

For instance, dealing with the gearbox assembly use case, the teaching framework's reasoning system compares the detected work pieces' pose uncertainties with the assembly's tolerances. If the workcell's perception system is not accurate enough, additional centering tasks are automatically inserted into the process, in order to lower the pose uncertainties to the level of the robot's repeatability. In the woodworking use case, the reasoning system automatically infers required tool change operations from the types of tasks present in the process. As a result, the operator can concentrate on the core process itself and does not have to manually specify these additional tasks.

The semantic process descriptions avoid to store absolute coordinates. Instead, relative transformations, defined as geometric constraints, are used. This allows the framework to ground relative assembly poses in sensor-based detections of the involved objects. As a result, the same process description can be re-used, when object locations change. The relative transformations might also be underspecified to express that certain degrees of freedom are invariant to the current task, e.g., when grasping a cylindrical object.

The product-centric process description does not contain information about a workcell's hardware until it is eventually deployed. Once a process is designed, it may be deployed without any reprogramming effort to different workcells with variations in robots, sensors, tools, or software features. By matching a workcell's capabilities, which are provided by its entities, with process requirements, incompatible workcell and process combinations can be determined. The gearbox assembly process has been successfully deployed on a single and a dual-arm robot workcell (see Fig. 1 and Fig. 11b).

VI. Conclusion

The preliminary study indicates the potential of semantically meaningful and object-centric robot programming. We demonstrated that domain-specific interfaces in a cognitive system with domain knowledge eases the tedious task of programming a robot to a large extent. By programming at the object level, the low-level details pertaining to robot

execution do not have to be programmed. The resulting process plans are more concise, readable, and reuseable. The communication between the operator and the robot system can be lifted to an abstract level, relying on already available knowledge on both sides. This enables a new operator to use the system with a minimal amount of training.

We plan to perform additional evaluations and full-scale user studies in the future, which will provide a more holistic assessment of the proposed robot teaching approach.

ACKNOWLEDGEMENTS

We would like to thank Comau Robotics and Mivelaz Techniques Bois SA for providing the industrial robot workcells used in our experiments, and Alfio Minissale[3] and Ludovic Resch[4] for their support and participation in the evaluation. Furthermore, we would like to gratefully acknowledge the contributions of Mathias Haage[5] and Ahmed Rehman Ghazi[6] regarding the robot controller interfaces for the Güdel gantry robot and the Comau dual-arm robot.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 287787 in the project SMErobotics.

REFERENCES

[1] R. D. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," in *Joint Conference of the International Symposium on Robotics (ISR) and the German Conference on Robotics (ROBOTIK)*, Munich, Germany, May 2006.

[2] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, Apr. 2012.

[3] M. Ejiri, T. Uno, H. Yoda, T. Goto, and K. Takeyasu, "A prototype intelligent robot that assembles objects from plan drawings," *IEEE Transactions on Computers*, vol. C-21, no. 2, pp. 161–170, 1972.

[4] A. Ambler and R. Popplestone, "Inferring the positions of bodies from specified spatial relationships," *Artificial Intelligence*, vol. 6, no. 2, pp. 157–174, June 1975.

[5] D. F. Corner, A. P. Anbler, and R. J. Popplestone, "Reasoning about the spatial relationships derived from a RAPT program for describing assembly by robot," pp. 842–844, Aug. 1983.

[6] B. O. Nnaji, *Theory of automatic robot assembly and programming*, 1st ed. London: Chapman & Hall, 1993.

[7] P. Gu and X. Yan, "CAD-directed automatic assembly sequence planning," *International Journal of Production Research*, vol. 33, no. 11, pp. 3069–3100, 1995.

[8] G. Pardo-Castellote, "Experiments in the integration and control of an intelligent manufacturing workcell," PhD Thesis, Stanford University, Sept. 1995.

[9] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning," *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, p. 241, 2003.

[10] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, June 2004.

[11] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artificial Intelligence*, vol. 172, no. 6-7, pp. 716–737, Apr. 2008.

[12] S. Bøgh, O. S. Nielsen, M. R. Pedersen, V. Krüger, and O. Madsen, "Does your robot have skills?" *Proceedings of the International Symposium on Robotics*, p. 6, 2012.

[3]Comau S.p.A., Grugliasco, Italy

[4]Mivelaz Techniques Bois SA, Le Bry, Switzerland

[5]Lund University, Lund, Sweden

[6]fortiss GmbH, Munich, Germany

[13] M. R. Pedersen, L. Nalpantidis, A. Bobick, and V. Krüger, "On the integration of hardware-abstracted robot skills for use in industrial scenarios," in *Proceedings of the IEEE/RSJ International Conference on Robots and Systems, Workshop on Cognitive Robotics Systems: Replicating Human Actions and Activities*, 2013.

[14] M. Stenmark, "Instructing industrial robots using high-level task descriptions," Licentiate Thesis, Lund University, 2015.

[15] M. Baqar Raza and R. Harrison, "Design, development & implementation of ontological knowledge based system for automotive assembly lines," *International Journal of Data Mining & Knowledge Management Process*, vol. 1, no. 5, pp. 21–40, 2011.

[16] M. Tenorth, A. Perzylo, R. Lafrenz, and M. Beetz, "The RoboEarth language: Representing and exchanging knowledge about actions, objects and environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, St. Paul, MN, USA, May 2012, pp. 1284–1289.

[17] M. Tenorth and M. Beetz, "KnowRob – a knowledge processing infrastructure for cognition-enabled robots," *International Journal of Robotics Research*, vol. 32, no. 5, pp. 566 – 590, Apr. 2013.

[18] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. R. Kramer, and E. Miguelanez, "An IEEE standard Ontology for Robotics and Automation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Algarve, Portugal, October 2012.

[19] S. Balakirsky, Z. Kootbally, T. Kramer, A. Pietromartire, C. Schlenoff, and S. Gupta, "Knowledge driven robotics for kitting applications," *Robotics and Autonomous Systems*, vol. 61, no. 11, pp. 1205–1214, Nov. 2013.

[20] S. Balakirsky and A. Price, "Implementation of an ontology for industrial robotics," *Standards for Knowledge Representation in Robotics*, pp. 10–15, 2014.

[21] A. Perzylo, N. Somani, S. Profanter, M. Rickert, and A. Knoll, "Toward efficient robot teach-in and semantic process descriptions for small lot sizes," in *Robotics: Science and Systems (RSS), Workshop on Combining AI Reasoning and Cognitive Science with Robotics*, Rome, Italy, July 2015, http://youtu.be/B1Qu8Mt3WtQ.

[22] T. De Laet, S. Bellens, R. Smits, E. Aertbeliën, H. Bruyninckx, and J. De Schutter, "Geometric relations between rigid bodies (part 1): Semantics for standardization," *IEEE Robotics Automation Magazine*, vol. 20, no. 1, pp. 84–93, Mar. 2013.

[23] A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 2015.

[24] N. Somani, A. Perzylo, C. Cai, M. Rickert, and A. Knoll, "Object detection using boundary representations of primitive shapes," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Zhuhai, China, December 2015.

[25] A. Perzylo, N. Somani, S. Profanter, M. Rickert, and A. Knoll, "Multimodal binding of parameters for task-based robot programming based on semantic descriptions of modalities and parameter types," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Multimodal Semantics for Robotic Systems*, Hamburg, Germany, September 2015.

[26] S. Profanter, A. Perzylo, N. Somani, M. Rickert, and A. Knoll, "Analysis and semantic modeling of modality preferences in industrial human-robot interaction," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 2015.

[27] N. Somani, A. Gaschler, M. Rickert, A. Perzylo, and A. Knoll, "Constraint-based task programming with CAD semantics: From intuitive specification to real-time control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 2015, https://youtu.be/qRJ1JmNoFEw.

[28] N. Somani, M. Rickert, A. Gaschler, C. Cai, A. Perzylo, and A. Knoll, "Task level robot programming using prioritized non-linear inequality constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Republic of Korea, October 2016, https://youtu.be/baet9IkTK04.

[29] N. Somani, E. Dean-Leon, C. Cai, and A. Knoll, "Scene Perception and Recognition in industrial environments," in *9th International Symposium on Visual Computing (ISVC)*. Springer, July 2013.