Generic Modeling Technique for Flexible and Highly Available Assembly Systems

Rainer Müller and Martin Karkowski

Abstract

To face the major challenges posed by the increasing product variants with shortening life cycles and fluctuating market conditions, modular and adaptable assembly systems are used. Their strong dependency on software creates a new void in the planning and implementation processes of these assembly systems. Usually, the programmer fills this void based on his knowledge, which leads to frequent and large adaptations of the code base. This is rather counter-productive. To address this challenge, we developed a generic user-friendly graphical API (Application Programming Interface) for a process owner in our previous work (Müller et al, Proc CIRP 81:730–735, 2019, [1]). This API can describe any assembly system and the associated task and is used to get the boilerplate code needed to execute the process on a programmable logic controller (PLC)—the standard hardware used in the industry. In this paper, the virtual description of an assembly system used by the API is extended to include a goal-oriented task description by defining the process and the structure. We believe that this extension provides the proper abstraction needed by the process owner. In addition, this extension significantly reduces the modeling effort.

Keywords

Petri nets • Agents • Planning • Process control • PLC • AI

R. Müller · M. Karkowski (✉)

Zentrum für Mechatronik und Automatisierungstechnik gemeinnützige GmbH, Eschberger Weg 46,
66121 Saarbrücken, Germany

E-mail: m.karkowski@zema.de

URL: <http://www.zema.de>

© The Author(s) 2022

T. Schüppstuhl et al. (eds.), *Annals of Scientific Society for Assembly, Handling and Industrial Robotics 2021*,

https://doi.org/10.1007/978-3-030-74032-0_1

1 Introduction

Integration of new production technologies—such as HRC (Human Robot Collaboration)—into assembly systems and requirements for mass customization lead to frequent adjustments of the assembly systems. For this reason, it is desirable to use flexible and versatile assembly systems [2].

However, constraints such as usage of off-the-shelf hardware (in the form of PLCs), assurance of higher availability of systems, deployment of different recovery strategies after an equipment's malfunction, presence of various analyzing and debugging capabilities, etc., lead to difficulty in the development of reusable, maintenance-friendly and well-structured software implementations [3].

Current methods often perform fundamental modifications to the software to ensure compatibility with the used equipment. This often leads to increased cost (in terms of software development time), slower deployment and increased time-to-manufacture, which contradicts the requirements of modern assembly systems [4].

In this paper, an extension to a comprehensive behavioral description of an assembly system based on Petri nets is presented. Our extension allows a goal-oriented task description of the assembly system and satisfies the aforementioned constraints while also satisfying the requirements of the software-defined modern assembly systems.

The text is structured as follows. Section 2 presents related approaches found in literature that aim to simplify the implementation of assembly systems. Section 3 presents previous fundamental work done on our system. Section 4 details the various building blocks used in our approach. The extension of our behavioral description is presented in Sect. 5. Section 6 presents a modified Refinement Action Engine used in our System.

2 Related Works

Plug and Produce represents a vision to adapt the structure of automated systems with a minimal manual reconfiguration of the software of the system [5]. Various techniques such as high-level programming [5, 6], Automatic Programming [7, 8] etc., form the components of any *Plug and Produce* system.

SMACH—a planning and execution system for ROS (Robot Operating System)—relies on a set of hierarchical state machines where the states are mapped to the execution of user-defined code to interact with ROS nodes [9]. SMACH is not designed for implementing low-level systems.

SkiROS is developed to control robots by efficiently linking semantically described tasks. Tasks are defined as sequences of skills. They comprise of atomic actions, like grabbing, transporting or placing, which are described semantically by an ontology. The actual task is planned by a modular task planner based on PDDL (Planning Domain Definition Language) [10].

These systems presented above don't satisfy one or more of the constraints mentioned in the previous section. For example, SkiROS assumes a deterministic planning domain and assigns the tasks by utilizing the matched production equipment. Usage of hierarchical state machines (like in [9]) in SMACH complicates the aggregation of an assembly system using construction kits that are modular in nature. Stateless description of the assembly process (like in [7, 8] or [10]) in *Plug and Produce*-type systems often forgo the modeling of relevant aspects for industrial applications such as errors or disturbances in processes. Other self-organizing approaches based on multi-agent systems are not suitable for industrial applications because of their unpredictable behavior [11] and high software complexity. Furthermore, a common recurring observation is that approaches lacking a graphical (e.g. [10]) task definition prevents integrating the knowledge of process owners at early stages of the development process.

3 Previous Works

In our previous works (see [1, 12]), a modeling technique based on Petri nets is developed. In order to obtain a virtual model of the assembly system, detailed product analysis is performed. Based on that analysis, a generic description of the process is manually defined graphically. The description uses Petri nets to express the logical sequence of the process. It models the assembly operations as skills which are used to derive the resources of an assembly system. These resources provide services that are linked to their capabilities via a taxonomy and are invoked at runtime. The transitions of Petri nets can be linked with services provided by the resources and the entire structure of the petrinet specifies when the services are called.

The mapping between resources and services can be derived automatically as long as the selected resources provide unique services and the mapping between the used skills and the services are unique. Alternatively, the mapping can be done manually. When deriving resources from the defined skills, it may be necessary to integrate additional skills. This leads to an iterative design process until appropriate resources are identified. The aggregation of these resources along with its manually modeled relations, restrictions and requirements denotes the model of our system's behavior. The complete model is obtained by adding the mechanical and electrical structure of the system as well as the flow of information. An exemplary application of the modeling technique is given in [1].

To perform the assembly task, the virtual model is transferred into the standardized control system (i.e. *Refinement Action Engine*). To optimize and validate the model, the standardized control system can be coupled to a simulation environment (see [13] for more details). This enables a fast adaptation and validation of the system behavior before the mechanical setup.

The expressiveness of the modeling technique is also its weakness since a complete representation of the entire process is very exhausting. This work addresses the following question: *How can the modeling of an assembly task be simplified?*

4 Theoretical Background of the Work

The following sections briefly describe the different building blocks used in our system. First, *Petri nets*, a de facto standard used for modeling discrete-event systems are presented. In our approach, Petri nets are used to model the base behavior of individual resources, the behavior of the assembly system, and the goal-oriented task-description. Second, *marking graphs* which enable exploration of the state space of the system are introduced. These graphs are derived from the Petri nets and they form the basis for automatic planning. The marking graph will be explored to determine plans for the goal-oriented task description. Afterward, the concept of a *Refinement Action Engine (RAE)*, which is used to execute model of an assembly system automatically, is presented. Integrating *planners* into a RAE enables automatic generation and adaptation of a plan. This approach is called *Automatic Planning* in the related literature.

4.1 Petri Nets

Petri nets comprises of *places*, *transitions*, and *edges* represented by circles, rectangles and arcs respectively. They are used for modeling discrete-event dynamic systems graphically. States of a system or subsystem are represented by places, whereas state changes, e.g. by actions or events, are represented by transitions. Active states are expressed by *marking* a place with tokens (filled circles), which are consumed and produced by transitions. Tokens represent abstract conditions or elements of the modeled system. A Petri net is defined by its places, transitions and their relations, modeled by edges that define the flow of tokens. The distribution of the tokens in the network is denoted as marking of the net [14].

The states of the Petri net are modeled explicitly, whereas other event-based modeling techniques define their states only implicitly. This is helpful because explicitly modeled states simplify the description and analysis of the modeled system. Besides Petri nets have a formal mathematical description and thence, they support simple methods for model-checking and simulation [15].

4.2 Marking Graphs

A marking M of the Petri net is reachable if there is a sequence of transitions that transforms the initial marking M_0 to the desired marking M . All reachable markings can be transformed

into the marking graph or reachability graph of the Petri net, where markings are expressed as nodes and the steps between the markings as edges. The Marking Graph defines the potential state space of the modeled system. It is generally infinitely large and often used as a starting point for an automated analysis [14].

4.3 Refinement Action Engine

A Refinement Action Engine (RAE) uses *events*, *facts*, *tasks* and *interactions* to influence an environment. The state of the system is described by facts, which are used to select refinement methods to fulfill the desired task. Refinement methods describe the effect on the environment and the conditions to be met. Each refinement method provides commands which are executed by an execution platform. The interaction of the platform with the surrounding environment results in events. These events are forwarded to the RAE and may cause new tasks. Tasks can be provided by additional external components, like users or planners [16].

4.4 Planners

AI planning systems—although called *planners* in this paper—are used to generate problem-specific solutions to fulfill a task. A plan itself comprises of actions, which can only be executed under certain conditions. A planning system usually develops a plan by either following some heuristics or learning on the fly. Please refer to [16] to gain a more in-depth understanding of different planning models.

4.5 Automated Planning and Acting

To check the correctness and validity of the plan, it must be executed or *acted* and in order to be able to perform an **automated acting**, the planning must be integrated into the *acting* of the system. In a RAE, this could be achieved by refinement planning. Ghallab et al. [16] provide a good overview of refinement planning methods. Our system uses a classical AI planning approach, utilizing the marking graph as search space. The planner generates full plans that are repaired immediately after unexpected events occur.

5 Extension of the Modeling Technique

In order to reduce the modeling effort, we propose an integration of a goal-based task description into our modeling technique and an integration of a planner into our control system.

For this purpose, the behavioral model of the resources is extended. The extension of the behavior description allows to distinguish between event-based state changes and planned state changes. Both are expressed via transitions. Event-based state changes occur in an uncontrolled manner and cannot be influenced, such as a human entering the workspace of a robotic system. They are called **events**. Planned state changes, also called **actions**—such as the movement of the robot from position A to B—occur intentionally. In addition, the target state of the execution of an action must be specified via special edges (see red edges in Fig. 1), which, however, is not necessarily achieved. As an example, the state “Robot at target” represents the target state of the action “Command robot to target”, whereas, for example, the event of a “Collision detected” leads to the state “Error” (not shown in the graphics).

The goal-oriented task description can then be automatically derived using the taxonomy. The taxonomy decouples the process description from the implementation by the resources. For this purpose, the linked actions of the transitions, which implement the skills used in the process description, are determined automatically and the entire target system state is derived based on their target states. Furthermore, the use of special edges enables the manual definition of desired system states. The following constraints can be modeled:

Fig. 1 Model to implement industrial applications connected to the standardized control system

- A resource must be in the defined state. (For this purpose, the marked places of the resource can be specified.)
- A resource must not be in the defined state. (For this purpose, the marked places of the resource, which must be avoided, can be specified.)

To perform the assembly task, the virtual model is transferred to the non-real-time standardized Control System (RAE) and IEC-61131-3-compatible PLC code is generated and transferred to the hardware used in production. This eliminates the need to manually translate the model into control code and improves the efficiency of the commissioning process. To simplify the PLC code, the generated code uses the RAE's broker to perform non-real-time critical tasks.

The manually and automatically defined goals are extracted and forward to a planner during runtime. The aggregated system behavior is utilized by the planner to determine the search space for plan generation by dynamically exploring and expanding the marking graph of underlying Petri net. Thus, a generic process is used to derive the currently active goals of the planner. Based on the current system state and active goal, plans are derived. Changing the system's state or achieving a goal, results in checking the planner's goals and plan. The plan will be updated or repaired if necessary. To optimize the plan generation different aspects like occurrence probabilities of events are learned while performing the assembly task. Different policies of the planner allow optimizing the plan in terms of different criteria.

6 Developed Control System

To perform the defined assembly task, the RAE uses five base elements (see Fig. 1). The defined logical sequence of the assembly task is dynamically interpreted by a dynamic logic interpreter (**DLI**), whereas the flow of data is implemented via a volatile memory system (**VMS**). The dynamic logic interpreter extracts the required assembly subtasks described in the virtual model and forwards them to a planner. The planner uses the virtual model to determine an appropriate solution. The plan is transferred to the dynamic logic interpreter, which ensures its execution. A **broker** delegates the tasks defined by the logic to the resources by utilizing the corresponding **communication layer** of the resource. The **resources** implement the skills as services. A detailed overview of components of the RAE is given in [12]. In the following sections, we present the developed planner and an approach to dynamically optimize the planner which enables dynamic adaptations of the assembly system to changes in its environment.

6.1 Planner

The Planner uses the aggregated behavior model of the system (see Fig. 2 left) in the form of a search space generated using the Petri net. For this purpose, the marking graph of the provided Petri net is dynamically expanded. A partial marking graph of the Petri net is shown on the right of Fig. 2, where the aggregated system states are represented by nodes and the transitions by edges.

Fig. 2 Aggregated system behavior and its corresponding marking graph

The underlying algorithm, essentially a modified A^* algorithm, is used to determine an optimal path to the desired goal state based on user-defined criteria (for e.g. shortest time and probability). Before executing the algorithm, useless transitions are automatically determined. These transitions are pruned to reduce the search space of the algorithm and guide it towards the desired goal (see Fig. 2 red line). The pseudo code of the algorithm is presented in Algorithm 1.

In each iteration, the algorithm needs to determine which of its paths to extend, where a path $p_{(0,n)}$ represents the traversal from the start node 0 to the node n . The cost $\vec{c}_{(0,n)}$ of a path $p_{(0,n)}$ and an estimate of the cost $f_h(n)$ required to extend the path to the desired goal are used to implement a search strategy where paths are explored in increasing order of their rating calculated by a user-defined rating function f_r . The values of the cost vector $\vec{c}_{(0,n)}$ represent the user-defined criteria. After selection, the corresponding node n of the selected path $p_{(0,n)}$ is removed from the open list *nodesToTest* and the costs to reach its successors are determined.

Thus, every edge of the marking graph contains a cost vector $\vec{c}_{(i,k)}$ (see $\vec{c}_{(4,6)}$ in Fig. 2 as example) which holds the additional required costs for traversal between marking M_i and M_k . To determine the cost vector $\vec{c}_{(0,k)}$, every criterion provides a function $f_c := \mathbb{R}^n \mapsto \mathbb{R}^n$ to combine two cost values. These functions apply to the corresponding cost vectors as follows:

$$\vec{c}_{0,k} := f_c(\vec{c}_{(0,i)}, \vec{c}_{(i,k)}) \quad (1)$$

Algorithm 1 Search Algorithm

```

1:  $nodesToTest = [0]$  where  $0 := startNode$ 
2:  $\vec{c}_{(0,0)} = \text{user-defined value}$ 
3: repeat
4:    $currentNode = n$  with  $\min f_r(f_c(\vec{c}_{(0,n)}, f_h(n)) \cdot \vec{w})$  where  $n \in nodesToTest$ 
5:   if  $currentNode$  is goal then
6:     Reconstruct the path from 0 to  $currentNode$ 
7:   end if
8:   remove  $currentNode$  from  $nodesToTest$ 
9:   for each  $edge, successor$  of  $currentNode$  do
10:    if  $edge$  not useless and  $successor \neq currentNode$  then
11:       $\vec{c}_{temp} := f_c(\vec{c}_{(0,currentNode)}, \vec{c}_{(currentNode,successor)})$ 
12:      if  $f_r(\vec{c}_{temp}) \cdot \vec{w} < f_r(\vec{c}_{(0,successor)}) \cdot \vec{w}$  or  $\vec{c}_{(0,successor)}$  is unknown then
13:         $\vec{c}_{(0,successor)} = \vec{c}_{temp}$ 
14:        put  $successor$  in  $nodesToTest$ 
15:         $parent(successor) = currentNode$ 
16:      end if
17:    end if
18:  end for
19: until length of  $nodesToTest = 0$ 

```

If the required cost to reach the successor node are lower than the previously determined cost or the successor has not been discovered, the newly found path $p_{(0,k)}$ is added to the open list. The algorithm terminates when the open list is empty or when the selected path leads to the desired goal.

To rate a node and select the most promising one for graph expansion, every criterion is rated by rating function $f_r := \mathbb{R}^n \mapsto \mathbb{R}^n$. The aggregated rating is calculated by combining the individual ratings with its user-defined weighting factor w_i :

$$r_k := f_r(\vec{c}_{(0,k)}) \cdot \vec{w} \quad (2)$$

with $\sum_{i=1}^n w_i = 1$ and $0 \leq f_r(c_{(0,k)}) \leq 1$. This enables finding an optimized solution for mixed criteria.

In our approach, a delete relaxation of the Petri net (see [17] for more details) is used as heuristic f_h to calculate the cost estimate required to extend a path to the desired goal. The relaxation assumes that transitions only require tokens in their preset places instead of consuming them when fired. A place in the relaxed Petri net, once marked, will never be unmarked. This implies that any reachable marking is reachable by a short sequence of transitions.

6.2 Optimizing the Generated Plans

To optimize the generated plans, the system can learn the cost vectors $\vec{c}_{(i,k)}$ to get more accurate information like an occurrence probability of an event. For this purpose, the events of the system and trackable cost values are recorded. The difference of the tracked values between two markings is used to update the cost vectors of the marking graph.

In our case, the system stores the differences of the tracked costs in a limited ring buffer. During updating the cost vectors of the marking graph, we generate the average value of the ring buffer. Thus, the size of the buffer defines the agility of the system. The system adaptability can be varied by varying the size of the buffer.

7 Conclusion and Outlook

To reduce the implementation effort of flexible assembly systems, a graphical modeling method in combination with a generic modular control system has been extended with aspects of planning. The graphical modeling technique and goal based description of the assembly task allow considering the know-how of the process owners in the early stages of the development process and reduces iterative optimizations during commissioning.

The actual process sequence of the assembly task is determined at runtime by using the behavioral model of the system and a generic description of the assembly task which further reduces the implementation effort. Additionally, we present an approach to automatically optimize the generation of process sequences. This approach adapts the system to changes in its environment during the runtime.

After successful validation of the behavioral model, generated code based on the model can be transferred to common hardware used by the industry. Code generation considerably simplifies and shortens the commissioning of the assembly system.

We are currently developing a physical demonstrator to validate our approach. In future research, aspects of automatic assembly operation optimization will be considered.

Acknowledgements This article is written within the project Mittelstand 4.0-Kompetenzzentrum Saarbrücken, as part of the Support Initiative “Mittelstand-Digital” of the BMWi. The nationally funded competence centers provide information to small and medium-sized companies about the opportunities and challenges of digitization. The authors are responsible for the content of the publication.

References

1. Müller, R., Scholer, M., Karkowski, M.: Generic automation task description for flexible assembly systems. *Proc. CIRP* **81**, 730–735 (2019). <https://doi.org/10.1016/j.procir.2019.03.185>
2. Michalos, G., Markis, S., Papakotas, N., Mourtzis, D., Chrysosolouris, G.: Automotive assembly technologies review: challenges and outlook for a flexible and adaptive approach. *CIRP J. Manuf. Sci. Technol.* **2**(2), 81–91

3. Schlick, J., Stephan, P., Loskyll, M., Lappe, D.: Industrie 4.0 in der praktischen Anwendung. In: M. Hoppel, B. Vogel-Heuser, T. Bauernhansl (eds.) *Handbuch Industrie 4.0*, Springer Reference Technik, 2., erweiterte und bearbeitete auflage edn. (2016)
4. Selig, A.: Informationsmodell Zur Funktionalen Typisierung von Automatisierungsgeräte. ISW/IPA-Forschung Und -Praxis, vol. 180. Jost-Jetter (2011)
5. Zöttl, A.: AutoPnP - Plug& Play für Automatisierungssysteme: Schlussbericht - Konsortialbericht (2014)
6. Antzoulatos, N., Castro, E., Scrimieri, D., Ratchev, S.: A multi-agent architecture for plug and produce on an industrial assembly platform. *Prod. Eng.* **8**(6), 773–781 (2014). <https://doi.org/10.1007/s11740-014-0571-x>
7. Danny, P., Ferreira, P., Lohse, N., Guedes, M.: An automation ML model for plug-and-produce assembly systems. In: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN): University of Applied Science Emden/Leer, Emden, Germany, 24–26 2017 : Proceedings, pp. 849–854. IEEE (2017). <https://doi.org/10.1109/INDIN.2017.8104883>
8. Anandan, P., Ferreira, P., Dorofeev, K., Lohse, N.: An event-based automationml model for the process execution of “plug-and-produce” assembly systems (2018). <https://doi.org/10.1109/INDIN.2018.8471955>
9. Bohren, J., Cousins, S.: The smach high-level executive. *IEEE Robot. Autom. Mag.* **17**, 18–20 (2011). <https://doi.org/10.1109/MRA.2010.938836>
10. Rovida, F., Crosby, M., Holz, D., Polydoros, A.S., Großmann, B., Petrick, R., Krüger, V.: SkiROS—a skill-based robot control platform on top of ROS. In: *ROBOT OPERATING SYSTEM: The Complete Reference*, pp. 121–160. Springer (2017)
11. Wooldridge, M.J.: *An Introduction to Multiagent Systems*, 2nd edn. Wiley (2009)
12. Müller, R., Scholer, M., Karkowski, M.: Increasing the flexibility of customized assembly systems with a modular control system. In: 2018 Fifth International Conference on Internet of Things: Systems, Management and Security, pp. 46–53 (2018). <https://doi.org/10.1109/IoTSMS.2018.8554528>
13. Illmer, B., Karkowski, M., Vielhaber, M.: Petri net controlled virtual commissioning – a virtual design-loop approach. In: *Enhancing Design Through the 4th Industrial Revolution Thinking*, p. 6. Elsevier B.V. (2020)
14. Reisig, W.: *Understanding Petri Nets*. Springer, Berlin (2013). <https://doi.org/10.1007/978-3-642-33278-4>
15. Aalst, W.: Discovering Petri Nets: A Personal Journey, pp. 3–9 (2019). https://doi.org/10.1007/978-3-319-96154-5_1
16. Ghallab, M., Nau, D., Traverso, P.: *Automated Planning and Acting*. Cambridge University Press (2016)
17. Bonet, B., Haslum, P., Hickmott, S., Thiebaux, S.: Directed unfolding of petri nets. *T. Petri Nets Other Model. Concurr.* **1**, 172–198 (2008). https://doi.org/10.1007/978-3-540-89287-8_11

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.