Plug & Produce by Modelling Skills and Service-Oriented Orchestration of Reconfigurable Manufacturing Systems

Julius Pfrommer[1], Denis Stogl[2], Kiril Aleksandrov[3], Stefan Escaida Navarro[2], Björn Hein[2], and Jürgen Beyerer[1]

[1] Fraunhofer IOSB, Fraunhoferstraße 1, D-76131 Karlsruhe
julius.pfrommer@iosb.fraunhofer.de,
[2] Karlsruhe Institute of Technology (KIT), Intelligent Process Control and Robotics (IPR), Engler-Bunte-Ring 8, D-76131 Karlsruhe
[3] FZI Forschungszentrum Informatik, Haid-und-Neu-Str. 10–14, D-76131 Karlsruhe

Abstract. Shortening product lifecycles and small lot sizes require manufacturing systems to adapt increasingly fast. Many existing machine tools, handling and logistics systems provide a generic functionality that is not bound to a specific product. But this flexibility and reconfigurability on the level of individual resources is lost in automated systems that are limited to the production of a fixed set of product variants. We propose a unified abstraction for the skills provided by the available resources and the product-specific manufacturing requirements. From these high-level descriptions, executable manufacturing procedures are derived, exposed as services and dynamically orchestrated at runtime in order to achieve the manufacturing goals.

1 Introduction

Automated systems for discrete manufacturing need to adapt increasingly fast [1,2]. However, replacing a resource or introducing a new product variant often requires manual integration work and considerable downtime. This is a major bottleneck for increasing the ability of companies to react to market changes, bespoke customer demands and unforeseen events in the supply chain [3]. The SkillPro project[4] develops techniques for Plug & Produce [4] where changes to the plant layout and products can be introduced even at runtime. This is achieved bya) a *unified abstraction* to describe the skills of production resources and the requirements of the product-specific manufacturing steps, b) the (partially automated) *generation of resource-specific actions* with a formal execution semantics from these high-level descriptions, and c) the *runtime orchestration* of the manufacturing system where these actions are exposed as *services*.

The paper is organized as follows. Section 2 gives an overview on related work and motivates this contribution. In Section 3, we introduce the overall system architecture. The skill concept used to model manufacturing resources and the requirements of the manufacturing steps is presented in Section 4. In contrast to this high-level abstraction, Section 5 gives the execution semantics of actions on the resource level. Section 6 closes the gap by showing how actions can be derived from skill-based descriptions of the resources and the product-specific manufacturing steps. Section 7 shows how the actions are exposed as services and dynamically orchestrated at runtime in order to achieve the manufacturing goals. Short descriptions of the implemented demonstration scenarios are presented in Section 8. The paper concludes in Section 9 with a summary and future outlook.

[4] Skill-based Propagation of Plug & Produce-Devices in Reconfigurable Production Systems by AML, see www.skillpro-project.eu

2 Related Work

Plug & Produce Arai et al. [4] first coined the term Plug & Produce in analogy to the Plug & Play concept in the computer world. Naumann et al. [5] propose an interconnector module that provides a uniform interface and that generates low-level commands for the underlying device. Lepuschitz et al. [6] show reconfiguration of manufacturing resources based on distributed IEC 61499 function blocks and a semantic description of the manufacturing setting and the expected behavior. Onori et al. [7] describe a self-configuring assembly system at the shop-floor level. Note that many works mentioned in the following sections on service-oriented manufacturing systems and manufacturing skill modelling also make use of Plug & Produce principles. The automated configuration of industrial communication systems in a Plug & Produce fashion [8,9] is not in the scope of this publication.

Service-Oriented Manufacturing Systems The authors from the SOCRADES project [10,11,12] and Shen et al. [13] develop a service-oriented manufacturing system architecture. They use semantic technologies to match possible providers of functionality. Loskyll et al. [14,15] expand the concept of semantic service discovery to the parametrisation and orchestration of services. For this, they develop a domain-specific ontology for the use in semantic reasoning tools. Puttonen et al. [16] describe a set of specialized web services for composing and invoking semantically enriched automated procedures in a manufacturing setting. They also present an algorithm to identify the steps required to reach a predefined goal state. Dürkop et al. [17] discuss the use of service-oriented architectures in reconfigurable manufacturing systems and the technical challenges that need to be overcome.

Manufacturing Skill Modelling The authors from the SIARAS project [18] use ontologies to store skills and their relations for production. This information is then used for automatic reconfiguration of production systems. Naumann et al. [5] use an ontology to model robot skills and use state-charts for sequences within manufacturing processes. Järvenpää et al. [19] define an ontology for capabilities in manufacturing and use it to map resources to manufacturing steps. Huckaby and Christensen [20] provide a taxonomy for assembly tasks and related skill primitives. Constraints specify whether they are executable in a certain situation. Björkelund et al. [21] first make use of the PPR (Product, Process, and Resource) concepts in the context of skills. They relate skills to all three views of PPR and represent skills as finite state machines. Pfrommer et al. [22] also define resource skills in relation to PPR. They are the first to differentiate between the generic skills of a resource and their executable realization for a specific product with formal pre- and postconditions. Keddis et al. [23] define a description vocabulary for capabilities of production resources and an accompanying algorithm for production planning and scheduling. Legat et al. [24] use a description of the resource capabilities to guide engineers in the implementation of field control code. Backhaus et al. [25] present a classification of manufacturing skills to enable task-oriented programming. This concept is expanded in [26] and used to generate executable tasks for a welding robot.

Many interesting approaches to describe the capabilities of manufacturing resources via ontologies and high-level description schemas have been proposed in the literature. All authors report the successful application in the demonstration scenarios for which their description schemas were developed. We argue that modelling semantic knowledge about manufacturing systems facilitates the integration of resources in a Plug & Produce fashion. But it is not enough. First, as ontologies for describing manufacturing skills become more detailed, the less general they are. This leads to the difficult situation wherea) general skill description schemas require additional information to cover implementation-specific edge-cases and constraints, and b) detailed skill description schemas are applicable to only a narrow subset of resources and will have difficulties in getting the required support across vendors and integration tooling suppliers. A possible scenario is the emergence of a standardized core skill description schema from which domain-specific and sometimes overlapping schemas will branch off. Second, semantic reasoners were developed to infer further information from an existing knowledge-base. But they are not equipped for reasoning about numerical optimization problems, such as the resource- and time-efficient manufacturing of many products on concurrent resources. Therefore, we see the role of semantic skill descriptions primarily for integration and configuration tasks. To make the flexibility of generic resource skills available on the level of large-scale systems, they need to be propagated to dedicated planning and runtime control systems that may use different representations. This also offers the possibility to integrate overlapping and domain-specific skill description schemas if they can be interfaced to a unified abstraction used for planning and runtime control.

Planning and Scheduling In the classical scheduling literature [27,28], the production of a specific product, called a job, is split into distinct operations with optional constraints on the ordering of the operations. Each operation can be processed by a subset of the available resources. Switching between types of operations on a resource may entail a downtime for tooling. The goal is to find a schedule that completes all jobs best according to some optimization criterion, such as order tardiness. Scheduling is a mature field and routinely deployed in manufacturing settings. However, many details are brushed over in the classical scheduling model. Logistics steps between manufacturing operations and the detailed cooperation of resources are often-times handled implicitly by custom automation solutions or the human workers. This tight coupling between resources, planning and runtime execution makes it difficult to add or remove resources in a Plug & Produce fashion.

For this reason, the planning approach used in SkillPro is not based on scheduling theory, but rather on results from the field of AI planning that arose out of McCarthy's original Situation Calculus [29,30]. In the Situation Calculus, the overall system state, called a *situation*, is transitioned via the execution of *actions* with well-defined preconditions and effects. The planner's task is to find an action sequence that reaches a situation matching the goal description. This has been an active research topic for more than 40 years and many important results, concerning for example numerical and temporal planning, as well as the search-guiding heuristics required to solve complex planning tasks, have been achieved only in recent years [31]. For completeness, we mention that further approaches for manufacturing planning exist and have been compared in [32].

3 System Architecture

The architecture of the SkillPro framework consists of three main components, spanning from long-term planning to the runtime orchestration of available resources during production. See Figure 1 for an overview. The system components bear some resemblance to the architecture in [33], but take on additional functionality as part of a skill-based manufacturing system.

Asset Management System (AMS) The AMS constitutes the central knowledge base of a manufacturing facility and provides this information to the adjacent components. It contains semantic descriptions of the available resources and their skills, as well as a detailed plant model including topological (e.g. how resources are arranged in work cells) and topographical (e.g. layout and position of the resources) relations. The AMS also holds product models, including drawings, bills of material and bills of processes. The AMS furthermore manages customer orders on a long-term horizon and ensures that the required resources with the right skills are available. Lastly, it interfaces the SkillPro framework with enterprise level ERP (Enterprise Resource Planning) and PDM (Product Data Management) systems.

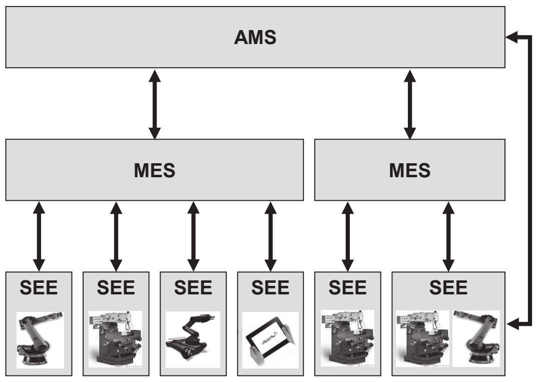


Fig. 1. Overview of the system architecture used in SkillPro.

Manufacturing Execution System (MES) The MES is responsible for orchestrating the available resources in order to achive short- to mid-term manufacturing goals. For this, the MES implements two main features working in lockstep: computing a fine-grained execution plan that accomplishes the manufacturing goals, and the orchestration of the manufacturing resources at runtime.

Skill Execution Engine (SEE) The SEE provide smart wrappers to the physical resources, which range from conveyor belts with little configurability to complex machine tools and even human workers. Facing towards the different resources, the SEE implement domain-specific connectivity, e.g. based on a fieldbus protocol or the Robot Operating System (ROS) [34]. In case of the human worker, this is accomplished with a tablet-based graphical-user-interface. Towards the MES, a single unified interface has been developed to ensure compatibility. This interface, based on the OPC UA protocol [35], provides access to the runtime state of the resource and the services that trigger the execution of manufacturing operations (called actions, cf. Sections 4 and 5). The SEE are self-aware in a sense that they contain an AutomationML-based [36] description of the wrapped resource that is used to exchange semantic data for Plug & Produce. Also, the SEE may contain domain-specific knowledge on how to derive actions from high-level skill-based descriptions.

4 Skill Modelling

In this work, we focus on discrete manufacturing. Process manufacturing applications, e.g. from the chemical and pharmaceutical industries, are out of scope. Furthermore, no time-constraints are assumed to exist between manufacturing steps. Wait times can be freely chosen by the planning system. A counterexample for this is the need for timely processing of molten metal.

The starting point for our skill model is the well-known product, process, and resource (PPR) model in the production context. We now recapitulate PPR and extend it for skill-based modelling (see also Figure 2):

Product Products represent marketable outputs as well as their intermediate forms during production and raw material. Products denote product types and not individual work-pieces.

Process Processes refer to a type of manufacturing, logistics or other production related process (such as information exchange, retooling, and so on). In SkillPro, processes are associated with a set of attributes used to describe the individual process implementations and the requirements towards them. Processes form a hierarchy where children inherit the attributes of their parent (see Figure 3).

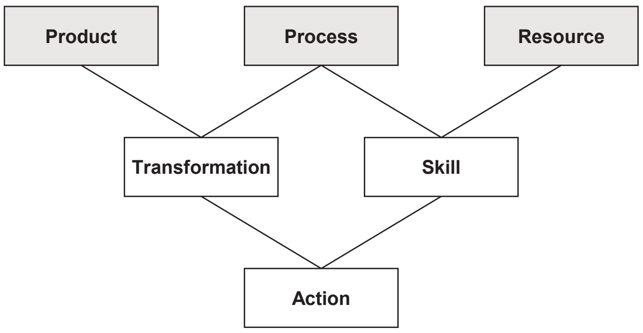


Fig. 2. Skill concept based on Product-Process-Resource

Resource Resources denote specific machines that are either standalone (e.g. a CNC mill) or bundled together (such as a robot manipulator with a gripper). In SkillPro, all resources are assumed to be accompanied by a smart wrapper, the SEE representing the resource. Via the SEE, each resource provides an AutomationML based self-description.

Skill Skills describe the ability of a resource to implement a production process. The skill constraints are defined via the attributes specified for the process. For example, a welding robot could indicate the range of supported temperatures in reference to the temperature attribute specified for the welding process. Depending on the specific domain, attributes may be stated as complex data structures, e.g. to describe possible product geometries for a 5-axis CNC mill.

Transformation Transformations describe the transition of one or more (intermediary) input products into output products by the application of one or more processes. The requirements for transformations are stated based on the attributes defined for the applied processes. This information is later used to identify resources with matching skills. Note that transformations state necessary requirements in order for a resource to be compatible. But they are not necessarily sufficient to guarantuee successful execution without additional information.

Action Actions are “executable skills”, i.e. concrete implementations of one or several product transformations or auxiliary processes, such as transportation, on the participating resources. Actions are known to be successfully executable if the specified preconditions hold. The SEE representing the participating resources contain all necessary information (such as compiled program code, IEC-61131 function blocks or configuration parameters), so that the execution can be triggered simply by naming the identifier of the action.

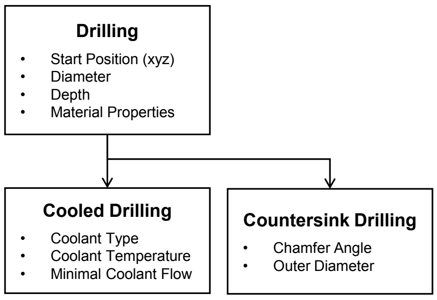
5 Formal Execution Semantics of Actions

The formal execution semantics of actions is defined by a single transition rule that can be used to describe any dynamics of discrete manufacturing systems, includinga) product transformations, b) transportation and storage, c) concurrency, i.e. parallel execution on many resources and the synchronization of resources for collaboration, and d) changes to resource configurations (e.g. tooling).

Assume for simplicity that resources $r \in R$ are always operational. The possible resource states $s \in S_r$ are defined as a tuple

$$s = \langle c, \boldsymbol{p} \rangle \in S_r \subseteq C_r \times \mathbb{N}_0^{|P|}$$

of the resource *configuration* $c \in C_r$ and a vector $\boldsymbol{p}$ denoting the number of products *currently contained* in the resource. Remember that product types $\rho \in P$ also denote intermediary product types



```
<RoleClass Name="Drilling"
   RefBaseClassPath="MetalIndustry/Separation">
   <Attribute Name="Start Position"
      AttributeDataType="Point" />
   <Attribute Name="Diameter" />
   <Attribute Name="Depth" />
   <Attribute Name="Material Properties" />
   <RoleClass Name="CooledDrilling"
      RefBaseClassPath=
         "MetalIndustry/Separation/Drilling">
      <Attribute Name="Coolant Type" />
      <Attribute Name="Coolant Temperature" />
      <Attribute Name="Minimal Coolant Flow" />
   </RoleClass>
   <RoleClass Name="CountersinkDrilling"
      RefBaseClassPath=
         "MetalIndustry/Separation/Drilling">
      <Attribute Name="Chamfer Angle" />
      <Attribute Name="Outer Diameter" />
   </RoleClass>
</RoleClass>
```

Fig. 3. Excerpt of the process hierarchy and its representation in AutomationML. The arrows indicate the inheritance of attributes from the parent processes. All attributes may refer to custom (structured) datatypes.

occuring during production as well as material. The configuration of a mobile robot might for example describe its position, whilst the configuration of a CNC mill would indicate the milling head currently used. Note that the S_r may become too large to enumerate every element, in which case it can be defined via the constraints that every element needs to hold. The overall system state is captured in a vector $\boldsymbol{\sigma}$ of time-indexed resource states

$$\sigma_r = \langle s, t \rangle \in S_r \times \mathbb{R} .$$

Its semantics is the next time-index t when r becomes available with state s. Until then, the resource executes (a series of) actions, during which no exact state is known. Actions a are defined as tuples comprised of a set of participating resources R_a, a set of valid initial states $V_a \subseteq S_a$ where $S_a = \times_{r \in R_a} S_r$, the effect function mapping the initial state to the post-execution state of the participating resources $\boldsymbol{e}_a : V_a \to S_a$, and the duration and slack timing functions $\boldsymbol{\tau}_a^{\text{slack}} : V_a \to \mathbb{R}_0^{|R_a|}$ and $\boldsymbol{\tau}_a^{\text{dur}} : V_a \to \mathbb{R}_0^{|R_a|}$.

$$a = \left\langle R_a, V_a, \boldsymbol{e}_a, \boldsymbol{\tau}_a^{\text{slack}}, \boldsymbol{\tau}_a^{\text{dur}} \right\rangle$$

The components of the vector-valued functions are indexed as $e_{a,r}$ or similar. After the start of an action, resources $r \in R_a$ join the execution after a slack time $\tau_{a,r}^{\text{slack}}$ for a duration of $\tau_{a,r}^{\text{dur}}$. Resources may be involved in a preceding action during the slack time (see Figure 4). A sequence of actions is concatenated as $w = a_1 a_2 \ldots$. The single rule that governs the transition between any $\boldsymbol{\sigma}^w$ and $\boldsymbol{\sigma}^{wa}$ is as follows. If the precondition $\boldsymbol{s}_a^w \in V_a$ is fulfilled (with $\boldsymbol{s}_a^w$ denoting the state of the involved resources R_a in $\boldsymbol{\sigma}^w$), then action a is executed at the earliest possible starting time $t_a^{\text{start}}(\boldsymbol{\sigma}^w) = \max \left\{ t^{\text{now}} \cup_{r \in R_a} \left(t_r^w - \tau_{a,r}^{\text{slack}}(\boldsymbol{s}_a^w) \right) \right\}$ and the system reaches the new time-indexed state $\boldsymbol{\sigma}^{wa}$.

$$\sigma_r^{wa} = \begin{cases} \langle e_{a,r}(\boldsymbol{s}_a^w), t_a^{\text{start}}(\boldsymbol{\sigma}^w) + \tau_{a,r}^{\text{slack}}(\boldsymbol{s}_a^w) + \tau_{a,r}^{\text{dur}}(\boldsymbol{s}_a^w) \rangle, & \text{if } r \in R_a \\ \langle s_r^w, t_r^w \rangle, & \text{else} \end{cases} \tag{1}$$

We now consider two abstractions of actions in order to simplify reasoning about large manufacturing systems and production volumes.

First, many manufacturing systems are organized in lines, within which a product will undergo a *fixed sequence of actions*. Such a sequence $\bar{a}$ can be treated atomically. The preconditions, effects

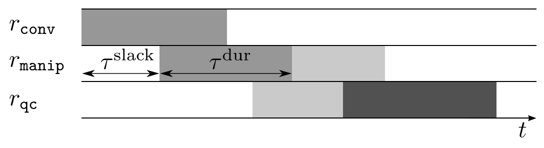


Fig. 4. Example with three resources, a conveyor, a robotic manipulator and a quality control unit, performing three actions in a row: picking an object from the conveyor (gray), placing the object in the quality control unit (light gray), and the quality control itself (dark gray). For the action of picking the object from the conveyor, the robotic manipulator has some slack time until the conveyor has moved the object in place.

and timings of sequences are modeled by composing the contained actions according to (1). Let $\bar{a}_i$, $i \in \{1, \dots, N_{\bar{a}}\}$ denote the ordered actions within the sequence and $\bar{a}_{\to i}$ the subsequence of the first i actions. The set of valid initial states $V_{\bar{a}} \subseteq \times_{r \in R_{\bar{a}}} S_r$ for the participating resources $R_{\bar{a}} = \cup_{a \in \bar{a}} R_a$ needs to assert that all intermediate states within the sequence are valid initial states for the following action.

$$\forall \boldsymbol{s}^w \in V_{\bar{a}}, \forall i \in \{0, \dots, N_{\bar{a}} - 1\}, \boldsymbol{s}^{w\bar{a}_{\to i}} \in V_{\bar{a}_{i+1}}$$

System parts with little flexibility can thus be modelled with comparatively few sequence-actions. These can be integrated seamlessly with more fine-grained actions where increased flexibility is desired.

Second, many actions have a *linear effect* on the number of contained products in the participating resources. Consider an action where parts are stamped from a metal coil and collected in a lattice box. Every execution increases the number of parts in the lattice box by a fixed amount. We say an action $\tilde{a}$ is linear if it fulfills the following three requirements: The configuration of the involved resources has not changed after the execution of $\tilde{a}$. The numerical effect on the number of products contained in the participating resources $\boldsymbol{\delta}_r^{\tilde{a}} \in \mathbb{N}^{|P|}$ is invariant to their initial state, so that $\forall \boldsymbol{s}^w \in V_{\tilde{a}}, \forall r \in R_{\tilde{a}}, \boldsymbol{p}_r^{w\tilde{a}} - \boldsymbol{p}_r^w = \boldsymbol{\delta}_r^{\tilde{a}}$. The possible combinations of initially contained products for each resource $\forall r \in R_{\tilde{a}}, \forall c_r \in C_r, \{\boldsymbol{p}_r : \exists \boldsymbol{s} \in V_{\tilde{a}}, s_r = \langle c_r, \boldsymbol{p}_r \rangle\}$ form a convex set. Knowing that an action is linear simplifies reasoning about preconditions and effects of its repeated execution. If the preconditions are fulfilled before and after n executions of $\tilde{a}$, with $\boldsymbol{s}_{\tilde{a}}^w \in V_{\tilde{a}}$ and $\langle c_r^w, \boldsymbol{p}_r^w + n\boldsymbol{\delta}_r^{\tilde{a}} \rangle_{r \in R_{\tilde{a}}} \in V_{\tilde{a}}$, then $\tilde{a}$ can be repeatedly executed up to $n+1$ times and the effect on the number of contained products is multiplied accordingly. Thus, linear actions simplify reasoning about manufacturing systems where lot sizes are large and the same (sequences of) actions are repeated many times in a row. Coming back to the example of stamping metal parts, we focus on the lattice box where finished parts are collected. Assuming a single resource configuration $c_{\texttt{box}}$, a maximum load weight $l_{\max}$ and the per-product weight $\boldsymbol{l} \in \mathbb{R}_+^{|P|}$, the valid initial states are $V_{\texttt{stamp}} = \{\boldsymbol{s} \in S_{\texttt{stamp}} : s_{\texttt{box}} = \langle c_{\texttt{box}}, \boldsymbol{p}_{\texttt{box}} \rangle, \boldsymbol{l}^\top (\boldsymbol{p}_{\texttt{box}} - \boldsymbol{\delta}_{\texttt{box}}^{\texttt{stamp}}) \leqslant l_{\max}\}$. Note that any combination of linear constraints leads to a convex set of allowed contained products.

6 From Skill-Based Descriptions to Executable Actions

Assume that either a new resource is added to the system or that a new product variant is being introduced. In both cases, new actions need to be generated. The newly generated actions can then be used by the MES in order to reach the manufacturing goals.

Matching product transformations to skills Assume that a list of product transformations (bill of processes, BOP) is initially provided or that the product description can be used to infer manufacturing steps, e.g. for assembly [37]. Remember that transformations and skills both use the attributes defined in the hierarchy of production processes (see Figure 3). Thus, the AMS can filter matching skills for the required transformations.

This task of determining the right operations to manufacture a specific product was originally investigated as Computer-aided Process Planning (CAPP, [38,39]). Note how CAPP differs from production planning and scheduling where the production of many products on a given plant layout is considered.

Adding auxiliary processes Auxiliary production steps, such as tooling and intra-logistic steps are usually not part of the BOP, since they are dependent on the specific production plant. In accordance to the matched production skills and the plant topology, appropriate skills are added to the solution set. For flexible transportation, conveyor belts, robotic manipulators with object recognition and mobile manipulators [40,41] were investigated and integrated in the demonstration scenarios of SkillPro.

Generation of executable actions The generation of executable actions from high-level skill-based descriptions requires the communication of AMS and SEE framework components. While actions are uniformly described according to Section 5, the SEE may internally implement them with a variety of technologies. Possible ways for implementing actions are:

- Customization of predefined procedures by parametrization, where the defined parameters are either resource- or manufacturing-domain specific [42]. Note that a description of the entire product in an appropriate format may constitute a parameter in this context. More examples of reasoning on the execution of a high-level task model can be found for example in the RoboEarth project [43,44].
- Automatic code synthesis from a high-level description [45], which can also be either resource-specific or based on a domain specific language [46,47,48].
- Manually programmed procedures, for example IEC-61131 function blocks.

Simulation and evaluation External simulation tools such as the *3DAutomate*[5] suite can provide further insight regarding the efficiency of the plant layout and a prediction for overall execution times. Possible production plan variants are evaluated based on key performance indicators, such as energy consumption, production time, retooling time etc. This information is later utilized during mid-term execution planning by the MES.

7 Service-Oriented Orchestration of Production Systems

The manufacturing operations for specific products are modelled as actions on the SEE and exposed as services that can be triggered over the network. The runtime planning and execution control is performed by the MES. All communication between MES and SEE is based on OPC UA [35]. Each resource is represented by a complex object in the OPC UA information model, providing state information and a simple interface to trigger actions. In addition to the *state* of the represented resource (see Section 5), the resource-objects contain a state machine for their current *mode*, i.e. the overall status that is unrelated to the specifics of the manufacturing context (see Figure 5). The resource-objects are hosted either on a central OPC UA server, or the individual SEE provide an embedded OPC UA server to which the MES connects as a client. The MES is subscribed to events that are emitted by the resource-objects to keep its internal system model synchronized.

At first, customer orders are placed in the AMS. The actions required to fulfill the present orders are identified by the AMS and instantiated on the SEEs. To start production, the AMS sends manufacturing

[5] Provided by the project partner Visual Components (`http://www.visualcomponents.com`).

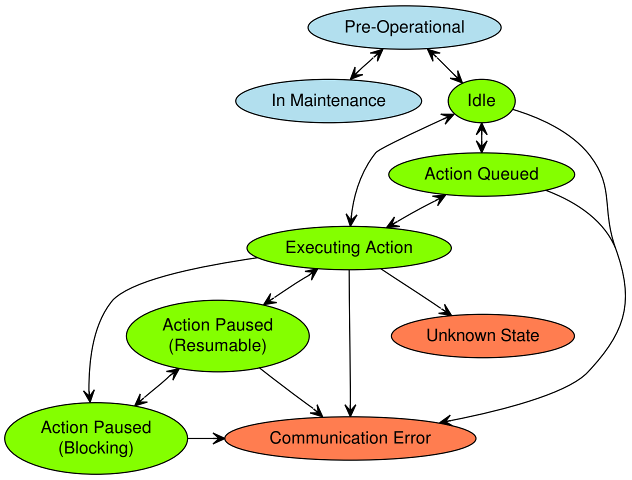


Fig. 5. State-machine of SEE modes and possible transitions. Blue modes denote long-term unavailability. The green modes are for runtime of the SEE and require active communication. Otherwise, the MES internally assumes the *Communication Error* mode. If the execution of an action aborts, the SEE may not know the exact state of its configuration and the contained products. In that case, it will switch to the *Unknown State* error mode that can only be resolved by human intervention.

goals for a mid-term horizon, such as having a certain combination of finished products in stock, to the MES. The MES infers how the actions made available by the SEE can be combined to reach the manufacturing goals. During the initial phase of the SkillPro project, an off-the-shelf planner [49] was used. It receives as input the current system state, available actions and the goal description "compiled" into standard PDDL (Planning Domain Definition Language, [50]) with numerical state variables. Its output is an ordered list of actions, each identified by a unique identifier. Based on the execution semantics defined in Section 5, this plan may also represent the cooperation between resources and concurrency. The planning step is repeated either periodically or after unforeseen events. This allows for very flexible reactions to quality problems, time delays and machine breakdowns. Together with the generation of actions as described in Section 6, it is also the basis for Plug & Produce on a functional level, as newly introduced resources and products are directly usable for the runtime orchestration. Ongoing work targets planning with multi-objective optimization criterion (order tardiness, energy consumption, material costs, quality metrics, and so on). The computed action sequence is processed in the control loop of the MES. It performs the following steps in a soft-realtime loop:

1. Update the internal system model of the MES according to status information received from the SEE.
2. Adjust the execution plan if small time delays have occured.
3. Test if the system state is consistent with the current execution plan. If not, set the current plan to an empty list.
4. Trigger replanning if the plan horizon is too short. During replanning (that might take a few seconds in a parallel thread), already triggered actions will continue to be executed by the SEE, but no new actions are triggered until the updated plan is available.
5. Notify the SEE whose next action execution is imminent.

OPC UA is an Ethernet-based protocol and does not make guarantuees for realtime communication. To achieve time-sensitive cooperation between SEEs, clock synchronisation [51] in combination with queued actions is used. That is, the MES may queue the execution of an action with a timestamp in the near future (see Figure 5). Safety-critical features, such as failsafes, are implemented within the SEE and do not make use of OPC UA. In addition, some SEE may be interconnected with additional communication channels to exchange runtime information (e.g. a conveyor makes use of a camera for positioning). But these are implementation-specific and unknown to the MES.

8 Validating Demonstration Scenarios

The approach developed in the SkillPro project has been demonstrated during the project's midterm-demo, with a demonstration scenario at Hannover Fair 2015, and with an electronics industry scenario. The midterm-demo scenario[6] combined simulated and physical resources executing processes of electronic assembly, soldering, transportation (with industrial manipulators and an autonomous mobile robotic platform) and safety area supervision. The system was able to gracefully resolve unexpected events, such as human workers in a safety area, by rerouting products and reallocating production tasks to different resources. The demonstration scenario at Hannover Fair 2015[7] combined human workers and machines, allowing a high degree of flexibility. The system allowed visitors to place customized product orders, that were then manufactured in the booth. Visitors could also take part in the fulfillment of their order by plugging into the system with a mobile human-machine interface, thus becoming one of the available resource. The electronics industry scenario was based on manufacturing processes of the industrial partner Dresden Elektronik and conducted at their premises. Here, human workers equipped with mobile human-machine interfaces were automatically triggered to reconfigure the available resources. The numerical reasoning capabilities in the MES planning component were leveraged to rebalance the production of customer orders as the availability of resources changed in the planning model.

The demonstration scenarios showed that the proposed approach can be used to model manufacturing systems and to use that model for runtime control. We report that even comparatively small scenarios yield a large number of actions in order to make the systems's flexibility visible in the MES. But the required effort can be mitigated by generating actions from high-level descriptions instead of manually defining them. During the development of the demonstration scenarios, sometimes auxiliary actions, e.g. for logistics steps, were missing for a complete action sequence from raw material to finished product. Consequently, the MES planning component simply stated the inability to fulfill the manufacturing goals. Therefore, custom tools were developed to visualize the predecessor-successor relations between actions and the potential product routes within the manufacturing system.

9 Summary and Outlook

This work presented an approach for Plug & Produce, making use of a unified model for the generic skills provided by manufacturing resources and the requirements of the different production steps. Automated actions are derived from this high-level description either automatically or with the directed assistance of automation engineers. Once available, the resulting actions are dynamically orchestrated as services in order to achieve the current manufacturing goals. As new resources announce their skills via a self-description mechanism, they can be integrated at runtime of the system in the sense of Plug & Produce. The same mechanism can also be used to introduce new and updated products. Note that the presentation in this work was kept largely agnostic of the different manufacturing domains (e.g. assembly, CNC milling, and so on). The described concepts aim to provide the interface where domain-specific description schemas and tools can be integrated into a coherent representation.

[6] See `https://www.youtube.com/watch?v=NgOeC7DeUsQ`

[7] See `https://www.youtube.com/watch?v=_5MN0lDwH8E`

We could show the successful application of the presented approach in three demonstration scenarios. Still, much work remains to be done before skill-based reconfigurable production systems can be used commercially at scale. Many researchers are investigating this timely topic. We hope that efforts will consolidate in the next years by agreeing on core concepts from which the domain-specific and sometimes overlapping description schemas can branch off.

Acknowledgements

This work has been partially funded by the European Commission through SkillPro project (Grant agreement ICT-287733). The authors would like to thank the SkillPro partners for the intensive exchange and collaboration.

References

1. Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. Van Brussel, “Reconfigurable manufacturing systems,” *CIRP Annals–Manufacturing Technology*, vol. 48, no. 2, pp. 527–540, 1999.
2. H.-P. Wiendahl, H. A. ElMaraghy, P. Nyhuis, M. Zäh, H.-H. Wiendahl, N. Duffie, and M. Brieke, “Changeable manufacturing-classification, design and operation,” *CIRP Annals–Manufacturing Technology*, vol. 56, no. 2, pp. 783–809, 2007.
3. D. Gerwin, “Manufacturing flexibility: a strategic perspective,” *Management science*, vol. 39, no. 4, pp. 395–410, 1993.
4. T. Arai, Y. Aiyama, Y. Maeda, M. Sugi, and J. Ota, “Agile assembly system by ’plug and produce’,” *CIRP Annals-Manufacturing Technology*, vol. 49, no. 1, pp. 1–4, 2000.
5. M. Naumann, K. Wegener, and R. D. Schraft, “Control architecture for robot cells to enable plug’n’produce,” in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 287–292.
6. W. Lepuschitz, A. Zoitl, M. Vallee, and M. Merdan, “Toward self-reconfiguration of manufacturing systems using automation agents,” *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 1, pp. 52–69, 2011.
7. M. Onori, N. Lohse, J. Barata, and C. Hanisch, “The IDEAS project: plug & produce at shop-floor level,” *Assembly automation*, vol. 32, no. 2, pp. 124–134, 2012.
8. L. Dürkop, H. Trsek, J. Jasperneite, and L. Wisniewski, “Towards autoconfiguration of industrial automation systems: A case study using profinet io,” in *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*. IEEE, 2012, pp. 1–8.
9. G. Reinhart, S. Krug, S. Hüttner, Z. Mari, F. Riedelbauch, and M. Schlögel, “Automatic configuration (plug & produce) of industrial ethernet networks,” in *Industry Applications (INDUSCON), 2010 9th IEEE/IAS International Conference on*. IEEE, 2010, pp. 1–6.
10. F. Jammes and H. Smit, “Service-oriented paradigms in industrial automation,” *Industrial Informatics, IEEE Transactions on*, vol. 1, no. 1, pp. 62–70, 2005.
11. L. M. S. De Souza, P. Spiess, D. Guinard, M. Köhler, S. Karnouskos, and D. Savio, “Socrades: A web service based shop floor integration infrastructure,” in *The internet of things*. Springer, 2008, pp. 50–67.
12. G. Cândido, A. W. Colombo, J. Barata, and F. Jammes, “Service-oriented infrastructure to support the deployment of evolvable production systems,” *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 759–767, 2011.
13. W. Shen, Q. Hao, S. Wang, Y. Li, and H. Ghenniwa, “An agent-based service-oriented integration architecture for collaborative intelligent manufacturing,” *Robotics and Computer-Integrated Manufacturing*, vol. 23, no. 3, pp. 315–325, 2007.
14. M. Loskyll, J. Schlick, S. Hodek, L. Ollinger, T. Gerber, and B. Pirvu, “Semantic service discovery and orchestration for manufacturing processes,” in *Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2011, pp. 1–8.
15. M. Loskyll, I. Heck, J. Schlick, and M. Schwarz, “Context-based orchestration for control of resource-efficient manufacturing processes,” *Future Internet*, vol. 4, no. 3, pp. 737–761, 2012.

16. J. Puttonen, A. Lobov, and J. L. Martinez Lastra, "Semantics-based composition of factory automation processes encapsulated by web services," *Industrial Informatics, IEEE Transactions on*, vol. 9, no. 4, pp. 2349–2359, 2013.
17. L. Dürkop, H. Trsek, J. Otto, and J. Jasperneite, "A field level architecture for reconfigurable real-time automation systems," in *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*. IEEE, 2014, pp. 1–10.
18. J. Malec, A. Nilsson, K. Nilsson, and S. Nowaczyk, "Knowledge-based reconfiguration of automation systems," in *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*, Sept 2007, pp. 170–175.
19. E. Jarvenpaa, P. Luostarinen, M. Lanz, and R. Tuokko, "Presenting capabilities of resources and resource combinations to support production system adaptation," in *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 1–6.
20. J. Huckaby and H. I. Christensen, "A taxonomic framework for task modeling and knowledge transfer in manufacturing robotics," in *Workshops at 26th AAAI Conference on Artificial Intelligence*, 2012.
21. A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, and P. Nugues, "Knowledge for intelligent industrial robots." in *AAAI Spring Symposium: Designing Intelligent Robots*, 2012.
22. J. Pfrommer, M. Schleipen, and J. Beyerer, "PPRS: Production skills and their relation to product, process, and resource," in *Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2013, pp. 1–4.
23. N. Keddis, G. Kainz, and A. Zoitl, "Capability-based planning and scheduling for adaptable manufacturing systems," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*. IEEE, 2014, pp. 1–8.
24. C. Legat, D. Schütz, and B. Vogel-Heuser, "Automatic generation of field control strategies for supporting (re-) engineering of manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 25, no. 5, pp. 1101–1111, 2014.
25. J. Backhaus, M. Ulrich, and G. Reinhart, "Classification, modelling and mapping of skills in automated production systems," in *Enabling Manufacturing Competitiveness and Economic Sustainability*. Springer, 2014, pp. 85–89.
26. J. Backhaus and G. Reinhart, "Adaptive and device independent planning module for task-oriented programming of assembly systems," *Procedia CIRP*, vol. 33, pp. 545–550, 2015.
27. M. L. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer, 2012.
28. J. Y. Leung, *Handbook of scheduling: algorithms, models, and performance analysis*. CRC Press, 2004.
29. J. McCarthy, "Situations, Actions and Causal Laws," Stanford University, Tech. Rep., 1963, reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pages 410–417.
30. R. Reiter, *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. Cambridge Univ Press, 2001, vol. 16.
31. J. Hoffmann, "Everything you always wanted to know about planning," in *KI 2011: Advances in Artificial Intelligence*. Springer, 2011, pp. 1–13.
32. A. Anis and O. Niggemann, "A comparison of modeling approaches for planning in cyber physical production systems," in *Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2014.
33. G. Schuh, S. Gottschalk, and T. Höhne, "High resolution production management," *CIRP Annals-Manufacturing Technology*, vol. 56, no. 1, pp. 439–442, 2007.
34. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.
35. W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Springer, 2009.
36. R. Drath, A. Lüder, J. Peschke, and L. Hundt, "AutomationML-the glue for seamless automation engineering," in *Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2008, pp. 616–623.
37. H. Tönshoff, E. Menzel, and H. Park, "A knowledge-based system for automated assembly planning," *CIRP Annals-Manufacturing Technology*, vol. 41, no. 1, pp. 19–24, 1992.
38. H. A. ElMaraghy, "Evolution and future perspectives of capp," *CIRP Annals-Manufacturing Technology*, vol. 42, no. 2, pp. 739–751, 1993.
39. D. Kiritsis, "A review of knowledge-based expert systems for process planning. methods and problems," *The International Journal of Advanced Manufacturing Technology*, vol. 10, no. 4, pp. 240–262, 1995.
40. A. Kamagaew, J. Stenzel, A. Nettstrater, and M. ten Hompel, "Concept of cellular transport systems in facility logistics," in *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*. IEEE, 2011, pp. 40–45.

41. S. Bogh, C. Schou, T. Ruehr, Y. Kogan, A. Doemel, M. Brucker, C. Eberst, R. Tornese, C. Sprunk, G. D. Tipaldi, and T. Hennessy, “Integration and assessment of multiple mobile manipulators in a real-world industrial production facility,” in *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, June 2014, pp. 1–8.
42. J. Otto and O. Niggemann, “Automatic parameterization of automation software for plug-and-produce,” *The AAAI-15 Workshop on Algorithm Configuration (AlgoConf 2015), Austin, Texas*, 2015.
43. M. Tenorth and M. Beetz, “Knowrob: knowledge processing for autonomous personal robots,” in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 4261–4266.
44. M. Waibel, M. Beetz, J. Civera, R. D’Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, “RoboEarth,” *IEEE Robotics Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.
45. B. Vogel-Heuser, D. Witsch, and U. Katzke, “Automatic code generation from a uml model to iec 61131-3 and system configuration tools,” in *Control and Automation, 2005. ICCA’05. International Conference on*, vol. 2. IEEE, 2005, pp. 1034–1039.
46. D. Dragomatz and S. Mann, “A classified bibliography of literature on nc milling path generation,” *Computer-Aided Design*, vol. 29, no. 3, pp. 239–247, 1997.
47. S. M. Mok, K. Ong, and C.-h. Wu, “Automatic generation of assembly instructions using step,” in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1. IEEE, 2001, pp. 313–318.
48. S. Mitsi, K.-D. Bouzakis, G. Mansour, D. Sagris, and G. Maliaris, “Off-line programming of an industrial robot for manufacturing,” *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 3, pp. 262–267, 2005.
49. R. M. Patrick Eyerich and G. Röger, “Using the context-enhanced additive heuristic for temporal and numeric planning,” in *In Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009).* AAAI, 2009, pp. 130–137.
50. M. Fox and D. Long, “PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains.” *J. Artif. Intell. Res.(JAIR)*, vol. 20, pp. 61–124, 2003.
51. J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.