Capabilities and Skills in Production Automation

Consolidating the concept from the perspective of the mechanical and plant engineering industry with a focus on OPC UA
Capabilities and Skills in Production Automation

Consolidating the concept from the perspective of the mechanical and plant engineering industry with a focus on OPC UA
Contents

Editorial 5
Introduction 6
Management summary 8
1 Introduction and motivation 9
  1.1 Introduction 9
  1.2 Simplifying problem-solving for automation systems 11
  1.3 Simplifying engineering using skills and capabilities 11
  1.4 Skills or capabilities? 13
2 Generating added value through the use of capabilities and skills 15
  2.1 Objectives for mechanical and plant engineers: 15
     2.1.1 Planning phase 15
     2.1.2 Commissioning 18
  2.2 Objectives for end users 20
3 Models for describing capabilities and skills 22
  3.1 Categorizing capability and skill models 22
  3.2 General metamodel for capabilities and skills 24
  3.3 Implementing capabilities as ontology in OWL 24
  3.4 Implementing skills 28
  3.5 Approach for using capabilities and skills 39
4 Examples of implementing skills using OPC UA 43
  4.1 Implementing skills using assembly technology as an example 43
  4.2 Implementing skills using manufacturing technology as an example 46
5 Editorial 48
The field of mechanical and plant engineering is in the midst of a fundamental change in its digital transformation. Producing machines and plants that are high in quality and also offer peak production will be just one of many USPs and sales arguments. The previous focus on the machine, its quality and efficiency is expanding to encompass functions and services that represent direct added value for the user. In the future, operators will only want to pay for what they actually use. "As-a-service" (XaaS) business models are therefore shifting into the spotlight for forward-looking mechanical and plant engineering. On the manufacturer’s side, advantages include projectable, recurring sources of income and more in-depth insight into operator behavior.

Using procedural instructions and concrete examples as a basis, this publication supports and consolidates the understanding and implementation of the concept of production automation capabilities and skills from the perspective of the mechanical and plant engineering industry. Interoperability is a key prerequisite for facilitating the concept’s efficiency, scaling and, with these, its acceptance and profitability. With the possible implementation of production automation capabilities and skills described here using the Open Platform Communication Unified Architecture (OPC UA) communication standard, taking into account the industry-wide information model — the OPC UA for Machinery companion specification —, the concept fits perfectly with existing interoperability activities in the mechanical and plant engineering industry.

At this point we would like to thank the companies and research institutes involved and the Industrie 4.0 (I4.0) platform for the successful coordination and joint efforts to create this document.

Special thanks are extended to Professor Rüdiger Daub, representing the Fraunhofer Institute for Casting, Composite and Processing Technology (IGCV), and Patrick Zimmermann for their academic input into the creation of this publication and the organization of the supporting industrial task force.

We hope you enjoy reading our publication.
The increasing complexity of products and customer demand for individualization present ever-growing challenges for manufacturing companies. On top of this, they also face shorter product and innovation cycles along with growing global competition. In order to still ensure economic production, even in high-wage countries like Germany, a high degree of automation is required. At the same time as this, automation generates increased outlay for planning and commissioning, meaning that production often does not become economical until high production quantities are reached. This is primarily due to the generally high degree of complexity in automation systems, which also differ in terms of their communication interfaces, program code and the functions provided. The problem affects everyone from manufacturers of micro components and machine and plant manufacturers through to major end clients. Small and medium-sized enterprises in the mechanical and plant engineering industry are particularly hard hit as they tend only to produce smaller quantities or even offer exclusively bespoke solutions to their customers.

The introduction of solutions from the area of Industrie 4.0 is able to provide support here and — by deftly combining different concepts and technology — can lead to a significant reduction in the outlay for the planning and commissioning of automated production systems. One goal here is to increase interoperability between arbitrary production resources. Arbitrary systems should be able to interact with one another more easily all the way up to fully automated setup (Plug & Produce) — just as users are familiar with from devices in the consumer sector.
Here, it is now regarded as par for the course that devices are able to communicate with one another via standardized interfaces such as Bluetooth®, WiFi or USB. The key to this is for the systems involved to share a common understanding of their existing functions and to communicate these in a standardized manner. For instance, a keypad uses USB to communicate its ability to facilitate inputs, while a Bluetooth loudspeaker shares its ability to play audio.

This publication shows how this concept can be transferred to applications in industry. In this process, standardized skills (e.g., “Move”) can be assigned to production resources, just like in the consumer sector. However, this raises the question as to how these skills are communicated and presented. Interoperability is only increased when other production resources “understand” this skill and respond to it. In this regard, the publication sets out a combination of various technologies, such as the use of the Open Platform Communication Unified Architecture (OPC UA) communication standard. Over recent years, use of this standard in the industrial sector has spread at an increasingly fast pace. The word “architecture” is decisive here as OPC UA facilitates the provision of a variety of communication mechanisms and information that can be adapted to the concept of capabilities and skills. Fraunhofer IGCV has spent a number of years studying the various ways in which OPC UA can be applied and — as part of this work — has taken an in-depth look at the implementation of skills.

With this publication, we now want to provide a comprehensive overview of the application of skills in the field of planning and commissioning and the technology available. In doing so, we would like to encourage machine and plant manufacturers in particular to engage with these innovative concepts in order to be compatible with production systems designed for interoperability. This skill is required to reduce planning costs for automated production systems as well as to continue guarantee economic production in Germany into the future.
These days, mechanical and plant engineering firms and operators find themselves confronted with a need to offer new business models and technical services. They are faced with trying to strike a balance between increasingly complex products, shortening innovation cycles and growing global competition. A wide array of research strategies exist for addressing these challenges, with particular focus placed on the potential offered by the use of skills in planning, commissioning and operation in the area of automation. The goal of this publication is to reflect a consolidated and consistent understanding and to provide guidance for the concept of skills in production automation.

Here, the word “skills” is a generic description of what certain automation resources are capable of. So, a linear drive unit can “move” or a gripper can “grip” regardless of the specific model or manufacturer. This concept is therefore a cross-manufacturer and cross-resource concept that can drastically reduce engineering outlay for the users of these resources.

During the planning phase, abilities are used as “capabilities” to create a standardized functional description and, as such, to generically select the right resources based on the requirements. In this process, ontologies help to improve understanding of the links between capabilities. Examples of this include the link between “Move” and the sub-category of “Linear movement”, and also the fact that, together, three “Linear movement” capabilities facilitate three-dimensional movement. The capabilities themselves can be described in the asset administration shells (AAS) of the respective resources.

If the goal is for these abilities to then be executed on the actual resources, the term “skills” is used

While it is currently common for resources to be programmed in bits and bytes in the control code — which also differs from manufacturer to manufacturer — skills take a service-oriented approach. Standardized interfaces such as OPC UA can be used to present capabilities as self-describing and directly executable skills. This enables them to be uniformly addressed across all resources without requiring any specific expert knowledge of how each resource is controlled, thereby drastically reducing the effort needed for commissioning from the end user’s perspective.

However, this type of standardized capability description can also help to optimize production processes during operation. Typical examples would be fluctuating capacity, a common occurrence in the current globalization-driven economy. From a planning perspective, uniform descriptions and opportunities for control enable decisions to be made directly in production, meaning that jobs can be prioritized dynamically and directly and production capacities can be increased. As a result, these priorities do not need to be translated across the various software systems, a process that also involves significant time delays. As a result of this increased momentum in production, the process can be adapted to job requirements in a much more direct manner, increasing delivery reliability and the end user’s satisfaction.

On the whole, skills and capabilities can therefore make the entire mechanical and plant engineering process easier — from the early planning stages through to commissioning — and increase flexibility and responsiveness during operation at the end customers.
1 Introduction and motivation

1.1 Introduction

The goal behind the Industrie 4.0 vision is to make manufacturing companies more resilient to market changes. These changes relate primarily to heavy fluctuations in demand caused by volatile global markets and ever-increasing customer demand for customized products (LINDEMANN ET AL. 2006). In order to meet the latter requirement, manufacturers are diversifying their product portfolio on the one hand and offering customer individualization depending on the product on the other (WIENDAHL ET AL. 2004). One example of this is the automotive industry, which has stepped up the development of SUV, hybrid and electric versions of its most common model series in recent years and has also made it possible for these vehicles to be customized using countless equipment options. A trend has also been observed among automation component manufacturers toward this style of customer orientation for mass products, with a stark increase in the range of variants available over the past few years. One example here includes robot manufacturers, who are not only offering ever-smaller grades of payload and range, but are also expanding their portfolios to include collaborative robots.

For the field of production automation, these trends are presenting new challenges with regard to flexibility and the adaptability of production lines (NYHUIS 2008, KOREN & SHPITALNI 2010). Requirements for future production environments are therefore the ability to produce a wide range of products through to custom manufacturing (flexibility) and the ability to modify lines so that they can produce newly developed products (adaptability). Since a number of variants are often produced almost in parallel, the total number of production stages and stations required also rises.

When combined with constant technical advances, which increase the complexity of products, these trends make it much more difficult to plan and commission automated production environments. One of the primary reasons for the high outlay during initial setup and modifications in line with new product variants is the heterogeneity within production automation. For instance, there are significant differences between programming interfaces, control commands and in the configuration of components. Furthermore, there is a large number of control interfaces and protocols that are incompatible with one another. Software development and project design have evolved into key cost factors, particularly in the field of mechanical and plant engineering. And in this process, individual components are still controlled in bits and bytes instead of in a service-oriented approach — as recommended in the I4.0 vision (KAGERMANN ET AL. 2013).
In the field of assembly technology in particular, this high outlay means that automated assembly lines only turn out to be economical for mass production. For low-quantity production batches, it is much more economical and quicker to adapt a manual assembly line to new product variants, particularly in countries with lower wages.

So, to protect Germany’s future status as a production location, the engineering of assembly systems needs to be simplified so that automated production remains more economical than manual production abroad, even with frequent product changes.
The concept of “skills and capabilities” is one approach for reducing the described engineering costs. These “skills and capabilities” are abstract functions of arbitrary automated resources, such as individual devices, stations, cells, machines or entire plants. For example, both a robot system with a gripper and a pick & place machine can provide a “Move” skill in relation to a workpiece. With regard to the planning of automation systems, the goal is to be able to place a greater focus on what needs to be done instead of how it can be achieved. As such, automation processes and their requirements can initially be described in a “resource-neutral” way, without having to be fixed on one or more specific manufacturers from the outset. To enable suitable resources to be found, manufacturers need to assign appropriate capabilities and skills to their products. At the moment, a great deal of effort is involved as customers first need to use component data sheets from a range of manufacturers to find out “what” a component can do, i.e., which function it offers. For many components, the process of simply assigning capabilities and skills can be quite clear; for example, the aforementioned Move skill is assigned to a six-axis articulated robot. However, there is also the option to further refine skills or to combine several capabilities to form a skill of a higher quality. A linear drive unit, for instance, is only able to offer the “Linear movement” skill, while a network of several linear units in a pick & place machine can also provide a multi-dimensional “Move” skill.

However, in addition to the actual skill, the constraints (guarantees) within which the skill can be performed are also of decisive importance. In the case of Move skills, for example, there are limitations in terms of payload, accuracy or clearance. If an arbitrary automation problem is now described on the basis of capabilities and skills and these constraints are defined as requirements, manufacturer-neutral assignment (matching) can take place between the capability-based requirements and the resources’ guarantees. In this process, the manufacturer and the actual resource type have no direct influence on the matching; the only relevant aspects are purely functional ones. In addition to significantly speeding up the process for selecting possible suitable resources, it could also allow for optimizations in terms of finding the most suitable resource. In practice, however, the components selected are often those already “familiar” to the engineers because analyzing all of the resources available on the market for every new problem is too time-consuming.

If capabilities have been used to find suitable resources for implementing an automation problem, the commissioning process still involves a great deal of time and effort. It is this process in particular where the full potential of skills and capabilities can be exploited, provided that the right technology is used. The importance of small and medium batch sizes rises, particularly in manufacturing environments (HERMANN 2021, HERMANN ET AL. 2019).

Typically, automation systems are programmed by hand in order to deal with any resource-specific differences. As such, manufacturers offer their own specific software interfaces, for example, for programming different programmable logic controllers (PLCs) or robot controllers. What is more, the actual commands needed to control the individual resources also differ. Here, “drivers” often have to be loaded (e.g., in the form of device description files) so that automation resources can be addressed correctly. One of the primary reasons for this is that components are still controlled on a bit-wise and byte-wise basis. This means that individual bits represent certain resource commands, which either have to be identified by the driver or looked up in the documentation by the user. Certain components may also require configuration directly on the device or using proprietary software (e.g., for configuring the operating mode or completing a
reference run). And last but not least, the heterogeneous communication interfaces used in industry are another key factor that make the commissioning process more difficult. Even though automated resources should really be selected exclusively on the basis of what they are capable of, other factors that do not relate to the actual function often come into play in practice. One example of this type of criterion is a suitable communication interface; this does not directly influence the functionality but has a major impact on the resource selection. Automation manufacturers are also able to use these dependencies to effectively tie customers to their own ecosystems.

So, how can describing skills and capabilities help with this problem? If we describe the functions of automation resources on the basis of skills and capabilities, many manufacturer-specific properties can be harmonized. If, for example, different manufacturers of pick & place machines were to offer a Move skill, within which different movement parameters could be configured, the individual programming interface and the manufacturer-specific control commands and configurations would no longer be required. However, this still raises the question as to “how” this skill is offered to the outside world, particularly in relation to the possible communication interfaces. As mentioned above, skills and capabilities have to be linked to a suitable piece of technology, which allows for manufacturer-independent control. The Open Platform Communications Unified Architecture (OPC UA), for example, is proving promising here, as it offers both a pure communication interface as well as an overall architecture system in which descriptive information for skills and capabilities can be modeled, for instance. As such, it is possible to set up an information model on an OPC UA server, which can be used to browse, configure and execute the skills provided using any OPC UA client. As a result, any automation components can be commissioned at the same time and the outlay for commissioning can be significantly reduced. Describing these skills and capabilities in OPC UA requires a high degree of standardization, which can be achieved using OPC UA companion specifications. The exact process for implementing skills and capabilities in OPC UA is described in section 3.5 and a sample application is set out in section 4.
1.4 Skills or capabilities?

In German, the word “Fähigkeiten” is used to describe the concept discussed here (see HAMMERSTINGL 2020), while English distinguishes between “skills” and “capabilities” (MOTSCH ET AL. 2021). While both words can be translated with “Fähigkeit” in German, the German term still describes two different concepts, just as the German term “Sicherheit” can be translated as both “safety” and “security”. The following section provides a brief description of the differences between the two concepts.

Capabilities are abstract and resource-independent abilities, which can be used to describe requirements from the side of the production process and the resources’ guarantees (see figure 2). The Product-Process-Resource model (PPR model) can be used as a basic concept for describing production systems and the processes performed by them; this model can also be used to explain the use of capabilities (Backhaus 2016; SPUR & KRAUSE 1997; DRATH 2010; HOLLMANN 2013). The required process is generally specified by the product or product specification. For instance, the product determines how the assembly process needs to proceed. Several assembly steps tend to be needed, which also include process steps from the field of joining in accordance with DIN 8593. Standards like this can be used as a basis for defining capabilities and skills.

Figure 2: Relationship between capabilities and skills
The requirements for the insertion joining process as per DIN 8593-1 can, for example, be described with a Move capability, in which the required range of movement, level of accuracy or mass to be moved is specified. Resources can also offer the Move capability and guarantee corresponding motion ranges, levels of accuracy or movable masses (payload) via this capability. This could apply, for instance, to a pick & place machine or a milling machine.

A process known as matching can then take place on the basis of the required and guaranteed capabilities; this process can be used to find resources suitable for performing the procedure. For matching to be feasible at all, it is important that comparable capabilities are available. To achieve this, both the requirements and the guarantees side need to use the same capability scheme, which can be described using an ontology, for example. Section 3.3 provides further information on how an ontology like this can be set up and used with the aid of the Web Ontology Language. The capability scheme enables the capability to be refined using additional properties. Here, it is important to note that there can be different levels of abstraction, e.g., a Handling capability, which can be achieved more specifically through movement or linear movement. Furthermore, capabilities can be combined to achieve higher-quality functionalities. For instance, a Pick & place capability can be achieved through multiple individual linear movements combined with a Grip capability. In this case, multiple individual resources connected to one another would also provide the capability, while a robot system with a gripper would not require any further breakdown into individual linear movements. So, during the matching process, it must be possible to compare these different capability levels to one another so that all suitable resources/resource combinations can be identified. In general, capabilities support the early planning stages of the engineering first and foremost — up until the selection of suitable resources. They therefore enable the planning time to be reduced and also facilitate quicker and more frequent adjustments in the event of changes.

Skills are the concrete implementation of capabilities within a certain resource. They can offer the same functionalities of different resource types and manufacturers in a service-oriented and standardized manner.

In this process, skills are accessed over a skill interface, which can be achieved with OPC UA — as described above. The program code required for the skill can still be implemented on a resource-specific basis. For example, a pick & place machine can offer a Move skill in a standardized manner over OPC UA, while the motion coordination and axis control processes required to execute the skill are implemented using proprietary commands within the respective system controllers made available by the OPC UA server. Skills are used primarily during the commissioning and operating phases. Providing a standardized skill interface for configuring the device and executing the required processes across all resources during runtime can save a great deal of time during commissioning. Like capabilities, skills can also be orchestrated. Looking at the example of the pick & place machine described above, the overarching controller is able to offer the Pick & place skill directly; when this skill is accessed, a subordinate sequence is called up, made of individual Linear movement and Grip skills.

Overall, the use of capabilities and skills is therefore able to support the entire engineering of automation systems — from modeling the requirements through to operating the automated resources. The concept can be applied both when creating new systems (greenfield) and reconfiguring existing ones (brownfield), meaning it can also be used to establish adaptable production.
2 Generating added value through the use of capabilities and skills

Capabilities and skills can be used in a variety of ways and offer different types of potential for different user groups. The following section sets out objectives for the use of capabilities and skills for mechanical and plant engineers and for the end users of automation systems.

2.1 Objectives for mechanical and plant engineers:

Capabilities and skills can be used at different points. For instance, component manufacturers are able to offer resources that directly provide capabilities and the associated implemented skills. In turn, system integration engineers can use them to set up machines and plants in a much shorter period of time. As a result, it is possible to establish an ongoing transition between planning, commissioning and operation using a standardized model (capability-based continuous engineering and operation).

The following section aims to distinguish between the planning phase — in which requirements are defined and a suitable automation solution is identified — and the implementation phase — in which the system is developed, set up, commissioned, operated or re-engineered. Even though these traditional planning phases become blurred in the Industrie 4.0 vision (and the concept of skills and capabilities can make an important contribution to fulfilling this vision), the separation of phases has been maintained in order to aid understanding.

2.1.1 Planning phase

Different providers may offer different solutions delivering the same functions. Manufacturers tend to describe their components in intricate detail in data sheets and often offer countless options when it comes to size and performance class. However, the resource’s actual functionality and possible areas of application are not always immediately obvious. Instead, a customer has to analyze each individual component to determine the possible area of application and study the data sheets to extract the limits that affect how the resource can be used.

Meanwhile, system integration engineers and end customers are expected to be very familiar with each resource manufacturer’s landscape and to search directly for the resource that fits their application perfectly. This calls for a number of resource experts — both on the customer’s side and on the resource manufacturer’s sales team — who can work together in consulting meetings to find suitable resources for the customer’s application. These meetings often involve many iterations as the requirements are not always clear, making it difficult to identify what the customer really needs. Furthermore, customers tend to already have very concrete ideas of what could solve the problem at hand, which makes the search for the perfect solution — covering all suitable resources — even more difficult. This can also be attributed to the fact that, when faced with new problems, customers want to fall back on the same automation solutions that they are already familiar with, even though there may be more suitable resources available for the specific automation problem.
Objectives when using capabilities and skills

In principle, employing capability-based descriptions during the planning phase can significantly reduce the level of expertise required by the customer in relation to the products offered by different resource manufacturers. Capabilities can be described in relation to the process or the product. A Move or Drill capability is an example of a process-related description, while a product-related capability expresses the task directly using the product, e.g., Mounting the hood or Producing a hole. Both styles of description can be beneficial depending on the application. As a general rule, a product-oriented description does not yet specify which specific sub-processes should be used to execute the capability. The hood can be mounted using a range of joining techniques; likewise, the hole can be produced using an array of manufacturing processes. As such, a supplier, system integration engineer or operator can define the specific suitable processes and the resources required for them at a later point in time. In contrast, a process-related description already identifies the specific processes; suitable resources can then be found for these processes on the basis of capabilities. Process-related descriptions often make sense in the field of handling tasks, while product-related descriptions are beneficial for the area of workpiece machining. In the case of workpieces, the required manufacturing processes (e.g., milling, turning, bending) are already defined indirectly by way of the geometry and material specifications. Geometric features to be created on the product, such as pockets or holes, can therefore be used to identify existing capabilities and skills for producing a geometric feature. In the case of capability-based resource selection, it is important that the description of the required capabilities matches the guaranteed capabilities (see figure 2).

While product-related descriptions and feasibility tests on resources already in use are the recommended course of action for the field of manufacturing (due to tool wear), this type of capability can only be applied to a limited extent when selecting resources in the area of assembly. The Mounting the hood capability, for example, is highly specialist but can still be built up from very generic Move and Grip capabilities. For resource manufacturers, it would be very difficult to list all possible products that could be assembled with their components for comparison purposes. Instead, it makes sense to break down the assembly process into resource-related capabilities — Move and Grip in this case — which are so generic that they can be offered by a range of resources. However, if a specialist solution is needed that should be designed precisely to assemble a particular product in high quantities, a product-based description can definitely make sense.
requirements up with half-finished ideas for solutions. This therefore ensures that a customer requests, for example, a solution for “moving component X weighing Y from position A to B” and not a “2D handling system in size X with payload Y”. With the help of solution-neutral requirements descriptions, a component manufacturer’s resource experts are able to select the right automation solutions much more efficiently. And on the manufacturer’s side, too, the use of skills and capabilities helps to significantly reduce the expertise required and, as a result, the time needed for developing possible solutions. When manufacturers describe their resources’ functionalities and application restrictions with the help of \textit{guaranteed capabilities}, solutions can be pre-selected quickly using the matching process (see figure 2). As in the field of manufacturing, feasibility tests can be performed at the level of the selected resources for certain resource combinations (for example, a robot with particular attachments that could reduce its payload), though they are not mandatory for the pre-selection process.

If the \textit{guaranteed capabilities} are provided to the customer by all manufacturers by way of \textit{asset administration shells}, the customer can also perform the matching process itself and gain an insight into which manufacturers actually offer suitable solutions. This opens up the market and ensures that the customer ends up with the “ideal solution” and not “the solution they are familiar with”. For customers looking for new solutions, this can also significantly reduce the number of manufacturer inquiries required as only suitable resources are listed.

The models needed could be offered over the portals used by automation component manufacturers. Content Management Systems (CMS) and Product Information Management systems (PIM) already exist for this purpose and can be accessed via online portals. In the future, capabilities and their guarantees could be requested automatically via these portals in order to carry out the matching process.

On the one hand, this style of generic, cross-manufacturer matching process can be regarded as a risk from the component manufacturer’s perspective, as customers are able to discover new, more suitable solutions offered by their competitors. On the other hand, manufacturers also have the chance to attract new customers who previously always used the same solutions from other manufacturers, including market leaders.

From a component manufacturer’s perspective, the chance to attract new customers (if their product quality is good enough) is significantly higher than the risk of losing existing customers, all the more so as a lack of awareness of a competitor’s potentially more suitable product should never be a market strategy. From the perspective of the component user (e.g., a system integration engineer or machine manufacturer), the concept can significantly reduce the time and effort involved in planning. The systems and machines developed can also be enhanced further with regard to the resources because automated matching enables a larger range of resources to be considered. For this reason, from the user’s perspective, preference should be given to component manufacturers who offer capabilities for their resources.
2.1.2 Commissioning

Once the capability matching process has identified suitable resources, these can be procured and work can start on setting up and commissioning the automation system. While capabilities are the focus during the planning phase, the spotlight is now shifted to the skills implemented in the resources, which can be accessed over interfaces. In the field of mechanical and plant engineering, programmable logic controllers (PLCs) or other resource-specific controllers (e.g., a robot control system) are normally used to control automation systems. Setting up and programming these controllers takes a great deal of time and effort, primarily due to inhomogeneity in four sub-areas:

1. **Programming interfaces:**
   A wide array of programming interfaces are used to program the relevant controllers. Well-known representatives in this area include, for example, TIA (Siemens), TwinCAT (Beckhoff) and CODESYS. Even though the same programming language can be used in principle (mostly structured text based on IEC 61131-3), the various interfaces and overall configurations are set up very differently. While there is the option of using large parameterizable PLC projects for machines and plants made up of recurring resources with similar scopes of functions, the process of programming new plants or integrating new resources is very complex. This exacerbates the aforementioned trend toward always using identical resources.

   The situation is intensified further by machines and plants containing devices that are equipped with their own controllers with proprietary programming languages. Typical examples here include image processing systems or robot controllers. Experts in each of these interfaces and languages are required for commissioning; certain systems may even be ruled out during the resource selection phase due to a lack of experts.

2. **Resource-specific application code:**
   Even devices without their own controllers often require very unique application code for configuration and operation. Regardless of the communication interface used, communication generally takes place using bits and bytes, the individual assignment of which can normally be checked in the resource’s documentation. Even within the same interface (e.g., IO-Link) and same type of resource (e.g., gripper), the configuration of commands at bit level can differ completely. Bit-wise transfer also means that information about its configuration cannot be taken out of the interface. In general, there have been advancements in this field, such as the use of function blocks (FB) or PLCopen approaches; however, these control blocks are not as interchangeable as suggested. The individual manufacturers keep offering extra functions to differentiate themselves from their competitors, which is why adjustments are needed all the time and direct reusability cannot be guaranteed. Furthermore, FBs are often too granular and have to be combined to even reproduce a resource’s full functionality at all.

3. **Communication interfaces for control purposes:**
   Different communication standards at field level also increase the time and effort needed for integration because the controllers need to have modules corresponding to every interface. Alternatively, a dominant interface can of course be defined for the system, which restricts the range of resources for selection accordingly. Overall, this can lead to a reduction in integration outlay, but also means that the resource offering the best possible functionality and economic efficiency may not be selected.

4. **Component configuration:**
   While some components can be configured directly over their control interface using highly complicated operations on an overarching controller, it is often the case that manufacturer-specific software is required for configuration. Customers often have to use the manufacturer’s own configuration.
interface. This is used to configure all settings for subsequent operation, such as the operating mode or reference runs.

In addition to these specific problems, the general separation of the planning and commissioning phases increases the time and effort involved. Even more modern standards, such as IEC 61499, provide no help here; because commissioning engineers are not brought into the overall planning process until a very late stage, they are always required to transfer the sequence created during the planning stage into control code.

Objectives when using capabilities and skills

The application of skills can help to counteract a number of these challenges. When using resources, it is important to note that a standardized description of capabilities and skills must always be accompanied by a physical interface suitable for this purpose. While it can be helpful to use the same exchange format for capabilities to allow the matching process to be automated as much as possible, it is in fact easier to translate these formats across one another during the planning phase provided that the structure and content (= semantics) are identical. Looking exclusively at the commissioning and operating phase in the mechanical and plant engineering sector, the standardized semantic access to a resource’s skills also requires a standardized communication interface.

As figure 2 shows, there must be a clear distinction between the actual executable skills and the skill interface. The goal is to implement capabilities into all resources in a way that enables them to be executed directly. They can be implemented on a resource-specific basis, particularly when the resource offers the skill directly. When movement is regarded as a skill, the actual control code for executing the movement in the case of a pick & place machine consisting of linear drive units and grippers is implemented by an overarching controller. The FBs from the PLCopen Motion Control library could be used for this purpose, for example, but it is also possible to use fully proprietary programming that directly addresses the bits and bytes required for the axes and grippers. In contrast, when robots and grippers are combined, the Move skill may need to be implemented into the corresponding robot-specific programming language, although most robot manufacturers also offer PLCopen Motion Control nowadays.

These implemented skills are accessed over the skill interface. To achieve the goal of significantly simplifying the commissioning process, this interface must have certain features. For instance, it must offer the option to search for skills (browsing), access a description of the skill, access configuration and runtime parameters, and also execute the skill. A suitable communication standard must be selected to reach these objectives. The OPC UA communication standard meets the aforementioned requirements and is already widely used in the mechanical and plant engineering sector. OPC UA servers can be integrated directly into resources’ controllers and are able to provide information models. In turn, these models can provide descriptions of skills, their configuration and runtime parameters, and method calls for executing them. A generic OPC UA client permits access to these servers and enables their content to be browsed.
This allows skills from a wide variety of resources to be configured and called at the same time without the need to use manufacturer-specific programming interfaces, control commands, communication interfaces or configuration tools. This means that a single generic tool can be used for programming, even when completely different controllers are in use. Overall, this would address all of the challenges mentioned and guarantee a significant reduction in the time and effort required and an increase in quality during commissioning due to fewer errors at the bit and byte level.

### 2.2 Objectives for end users

End users also benefit from machines and plants that have been developed on the basis of capabilities and skills. At the moment, the production sequence is determined by master production controllers. In this approach, signals have to be passed across the individual levels of the traditional automation level and often have to be converted. In the field of vehicle production, for example, variants are created using RFID tags in the product, which determine which program is executed. The primary goal when applying capabilities and skills would be to reduce the complexity of systems from the end user’s perspective — during both the planning and operating phases. A further goal would be to allow for more direct communication between the master controllers and the machines and plants that perform the processes, which would increase both process transparency and production flexibility. Here, skills could also be addressed directly via ERP and MES, insofar as they are provided with suitably standardized interfaces.

When specifying requirements for their production lines, end users could also use capabilities and skills to directly model the process and its individual stages and, as a result, document the requirements in a uniform way. As already discussed, capabilities and skills can be modeled on a product-related basis to begin with and then broken down into ever-broader sub-steps, e.g., into individual joining operations. These can then be modeled as “requirement capabilities” and used by system integration engineers or resource manufacturers for resource planning or commissioning, as described in section 2.1.

### Determining the course of production

One scenario that is particularly relevant for end users when applying capabilities and skills is the simplified modification of the production sequence. If the sequence controller for a production sequence were to be based on skills, it would be possible to achieve a high degree of transparency in relation to each of the process stages. Under the conventional approach of fully manual programming, these process stages are often “hidden” behind cryptic control variables. Intervention in the process sequence has to be explicitly integrated into the system by means of corresponding variables.

Skills make it easier to access economic aspects and enable these to be incorporated seamlessly into the automation process. To provide a concrete example: Users are able to check whether certain “switch points” in the production process can be activated depending on the order or material master configurations. The activation of these “switch points” can then also be triggered directly on the basis of skills. Here, the term “switch point” refers generally to any point where a decision is made. It covers both “real” switch points as well as adjustable processes, formulas or states (e.g., based on sensors).
The goal is to identify priorities — stemming from the order itself — which can be transferred into skill-based process adjustment. Here, customer priorities need to be known not only at ERP level, but also across the entire automation pyramid. Priorities can be assigned to every order in the MES and integrated into the automation technology in a flexible manner. Of course, a master computer could also do this job as well. With the “breakdown” of the classic automation pyramid in the context of I4.0, the flow of information could, however, be shortened here. The goal is to make information about an order’s priorities known up until a switch point. For instance, if the individual machines have access to this information before starting to process an order, it is still possible to make last-minute decisions. On an overarching level, skills can help to reduce the complexity between ERP and master computers and allow production to be managed holistically.

Reconfiguration

As well as helping to set the course of the production process, skills also provide support when reconfiguring a production line for a different product variant or a brand new product. The option of parameterizing skills enables modifications for new products to be implemented quickly, provided that all the skills required are already present within the production line. If a product with brand new requirements is introduced and potentially calls for skills beyond those offered by existing resources, any new resources needed can be identified quickly. Skill-based commissioning is able to minimize the amount of manual reconfiguration work needed to adapt a line to a new product.

When implementing the concept of capabilities and skills, a certain degree of standardization must be achieved as this is the only way that end users can benefit from the concept. This covers both the modeling of capabilities and skills and the technology for implementation at skill level. In general, a large number of software applications are already classified according to their skills and OPC UA is already widely used among end users. Due to its manufacturer-neutral approach and its option to reproduce information models directly on a server, OPC UA is the preferred communication technology for the skill interface, while the skills enable capabilities to be implemented in a manufacturer-independent way. However, it is mainly used for data provision at present; when OPC UA is used as a skill interface, control and configuration data is primarily sent to resources. End users still lack the tools needed for the concept to be employed seamlessly.

In addition to the scenarios discussed and the cases that are particularly relevant to the field of mechanical and plant engineering, there are a number of other possible scenarios for using capabilities and skills, particularly in the operating phase, such as rapid response to faults, production monitoring or optimization of the production process (DIETRICH ET AL. 2022).
3 Models for describing capabilities and skills

While section 2 set out different types of added value that using capabilities and skills can offer to stakeholders, the following section outlines models and approaches for modeling and applying capabilities and skills.

3.1 Categorizing capability and skill models

As described in section 1.4, a distinction must be drawn between resource-related skills and implementation-neutral capabilities to enable planning to take place on the basis of capabilities and skills, even before any concrete production resources are in place.

For requirement(s) and guarantee(s) to be compared, capabilities and skills must be described in a “neutral” language. Both the language and the content must be defined accordingly, independently of concrete processes and resources. In contrast to natural language, the language must be largely formalized so that it can be interpreted by machines and so that corresponding automated matching algorithms can be applied to it. The descriptions of how these models are set up, i.e., how capabilities and skills need to be modeled, are known as metamodels.
In principle, three different metamodels are needed, which describe (1) the cross-resource and cross-process description of capabilities, (2) the general description of the skill interface and (3) the allocation of capabilities and skills to a concrete process or resource. These metamodels and the models based on them can be presented in a variety of formal languages. A decisive factor for the complexity and efficiency of the representation in a particular language is the language's expressive power and its resulting ability to complement the content to be presented (here: capabilities, skills and their matching with resources and processes).

Ontologies are a suitable language for describing the capability metamodel (1) and the respective concrete capabilities (capability catalogs), with the Web Ontology Language (OWL) being especially suited to this task. Its standardization by way of W3C with corresponding tooling and its link to a formal logic that can be used as a basis for formal, verifiable and explainable capability matching are two beneficial features. The capability metamodel is implemented as an OWL ontology, which defines the fundamental vocabulary for describing capabilities, their relationships and their attributes.

The description of the skill interface metamodel (2) is provided by a standardized OPC UA information model (e.g., as a companion specification). This describes how skill interfaces need to be modeled.

Capabilities and skills are matched to specific types of resources and processes (3) by way of asset administration shells or their submodels, a process which is specified in the asset administration shell metamodel.

An appropriate capability submodel is currently being developed to describe the capabilities offered by the resource type (IDTA 2022A). The capability submodel establishes a link between the resource/process and the ontologies, which are outside of the asset administration shells. Along with the reference to the required or guaranteed capabilities, the concrete data values or data value ranges are also developed at this point. In precise terms, this means that the capability submodel must contain all information to enable it to be translated into an OWL class description, as outlined in section 3.3. In line with the PPR model in figure 2, a process needs to be allocated to a formal description of the capabilities it requires. Resources, on the other hand, need to be allocated to a formal description of the capabilities and skills that they offer. In addition to the capability submodel, a “control component submodel” is also being developed, which will generate a link to the offered skill interfaces (IDTA 2022B). This submodel is able to abstract the various resource control options in a standardized manner, including the concept of skills.

It is important to emphasize that the capability and skill descriptions and the allocation to resources/processes relate to the resource/process type. The linked capabilities/skills apply to all specific forms (= instances) of the types in question. As such, as well being used during actual operation, application is also possible during the planning phase, i.e., at a point at which the concrete process/resource instances do not yet exist.

Figure 3 shows the classification of capability and skill models for the guarantee side (resources). For capability matching, the resource instance in question contains the same information as the resource type. In contrast, the skill interface is offered via a specific OPC UA server model instead of a pure node set description.
3.2 General metamodel for capabilities and skills

Figure 4 illustrates an overarching metamodel for capabilities and skills based on DIETRICH ET AL. (2022). The metamodel describes the way in which capabilities and skills should be described. A standard metamodel is essential for interoperability, i.e., for the ability to interpret the specific capability and skill descriptions in a uniform way. It is important to note that the standard metamodel for capabilities and skills can be expressed using different language models (for capabilities and skills respectively) and conforms to the metamodel defined in DIETRICH ET AL. (2022).

The metamodel specifies that a capability can be refined in more detail by inputting properties. The applicability of a capability can also be restricted by constraints. In terms of their description, constraints relate back to properties.

3.3 Implementing capabilities as ontology in OWL

As well as representing capabilities in an implementation-neutral manner, the metamodel also describes an optional link between a capability and its skill. Skills can have parameters (specifically input and output parameters), which can correspond to the capability-describing properties on the implementation side. Alongside its parameters, each skill must at the very least be enclosed by a skill interface and is described internally by a finite automaton.

In accordance with the arguments set out above, the metamodel for capabilities can be implemented using an ontology model in Web Ontology Language (OWL). Figure 5 is a sample depiction of the relationship between the general metamodel described in section 3.2 and the representation in OWL with examples.
In addition to the elements depicted in the metamodel, the OWL representation contains the following options for describing more specific capabilities:

<table>
<thead>
<tr>
<th>OWL language element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL sub-class axioms for modeling a class hierarchy for the purpose of capability specialization.</td>
<td>Development of a capability hierarchy in the sense of specialization (not composition). Example: with reference to standards for manufacturing methods, such as DIN 8580 or VDA2860.</td>
</tr>
<tr>
<td>OWL object properties (hasCapability, associatedWithCapability) for describing relationships between processes/resources and capabilities.</td>
<td>Shifting to the level of assets (process or resource) enables non-hierarchical relationships between capabilities to be described, e.g., as part of a composition.</td>
</tr>
</tbody>
</table>
The general metamodel can be used by various organizations in order to describe concrete capabilities. This description consists, in particular, of the development of sub-classes for the root element and of the description of compositions. Any form of specialization hierarchy is possible here. For instance, ontologies for generic capabilities can be published on the basis of standardized guidelines (e.g., handling technology) or on a sector-specific basis (e.g., robotics). Where necessary, these can be specified further, right down to the level of particular companies and their products. This formalization and the modular approach therefore facilitate automated capability matching. For instance, a requirement to move something (Move capability, generally specified for handling technology) can be offered by a robot from a particular company (Move linear 6D capability). Through the use of a common language and specialization patterns, the robot can be classified as a provider of the required capability. Similarly, there is also the option to compare value ranges for properties. To achieve this, a capability constraint for a guarantee can be formulated using a property restriction in a way that means the capability only fulfills properties within a certain value range. And vice versa, a capability constraint for a requirement can be formulated using a property restriction in a way that means the capability has to fulfill properties within a certain value range. To give an example: A guarantee for a robot can be described as has-Capability some (C2MoveLin6D and hasPayloadKg only [<= 100]) i.e., the class for all things (resources in this case) that “possess” the capability (in the sense of “guarantee” here) to move something linearly in 6D and specifically only payloads of less than 100 kg.

In this sample description, C2MoveLin6D stands for the capability, hasPayloadKg for a property and hasPayloadKg only [<=100] for a constraint in accordance with the metamodel.

A sample requirement can be represented accordingly as hasCapability some (C2Move and hasPayloadKg some [>= 90]) i.e., the class for all things (processes in this case) that “possess” the capability (in the sense of “require” here) to move something, specifically payloads over 90 kg.

Based on the modeling pattern described in these examples, Boolean operators (AND, OR, NOT) can be used to construct any number of complex capability descriptions, which contain both numerical and symbolic property forms. For instance, a constraint for a “Joining” capability can be applied to permit only adherends which can be assigned to a certain material class that can be handled by an equivalent kind of robot/gripper — e.g., a “housing component” — and which include a geometric feature required for interlocking gripping — e.g., a “notch” with a minimum width. This example shows that — depending on the desired degree of description accuracy for the matching scenarios — there may be a need for much more complex object models that go beyond a simple list of numerical parameters. In this example, different domain model elements would be introduced in the form of additional classes and properties, for instance, in order to distinguish adherends from fixed components, geometric features with their own properties and relationships between these. In general, semantic techniques like OWL ontologies are a very suitable tool for providing a very accurate representation of domain models.

1 The representation of an OWL class description used here is a simplified version of the OWL Manchester Syntax..
Modeling in accordance with the metamodel and applying a given capability model as a basis (C2MoveLin6D as a sub-class of C2Move, itself a sub-class of capability, plus hasPayloadKg as a data sub-property of hasProperty) enables capability matching to be carried out. Existing approaches to ontology-based matchmaking can be applied here, e.g., relating to the ability of class conjunctions to be fulfilled or subsumption between classes using the techniques described in LI & HORROCKS (2003). In the first instance, the open world semantics of OWL enable a negative verdict to be drawn regarding the incompatibility of requirement and guarantee, thereby enabling the resources to be included in a selection process to be narrowed down significantly — any logical discrepancies identified when comparing requirements allow resources to be clearly ruled out. A (positive) verdict to the contrary is not immediately possible unless it can be ensured that the capability modeling process has reached a certain level of completeness. This corresponds to the belief that no guarantee can be provided for the feasibility of a capability requirement without full knowledge of all relevant underlying conditions. (Key features of OWL’s open world semantics in relation to matchmaking were discussed in detail in GRIMM (2009).)

Assuming knowledge of all the relevant factors required for matching, the proposed modeling process can, however, be expanded to include modeling patterns, such as closure axioms, so that positive matching verdicts can be achieved. Knowledge of the required property forms could also be assured, for example, by the Shapes Constraint Language (SHACL) semantic web technology, which is compatible with OWL.

The approach could also be expanded through the use of SPARQL queries for implementing numerical preliminary calculations — e.g., as part of a pre-processing procedure — as additional technology from the Semantic Web Stack. If the focus is shifted to more complex numerical calculations, the integration of constraint solvers can also be beneficial. In general, the exact method for applying the technologies suggested here requires more in-depth research for the purposes of capability matching and the combination of these technologies in particular needs further development. Nevertheless, the framework from RDF and OWL is a good starting point for directly implementing the capability section of the abstract concept model using modern knowledge representation tools and automated reasoning and then testing or expanding it in practice.
3.4 Implementing skills

Over the last few years, a range of research projects, such as DEVEKOS, BaSys4.0/4.2 and AKOMI, have looked into the options for implementing skills (MALAKUTI ET AL. 2018, DOROFEEV & ZOITL 2018, ZIMMERMANN ET AL. 2019, HAMMERSTINGL 2020, VOLKMANN ET AL. 2021). While the actual implementation of skills has to take place on a resource-specific basis using proprietary control code, the OPC UA communication standard has emerged as a promising approach for implementing skill interfaces. The core idea behind skills is to enable resources to be controlled on a cross-manufacturer basis, whereby the communication interface is directly linked to the successful implementation of the concept of skills. OPC UA is now widely used, particularly in the field of control technology, and is employed as manufacturer-neutral and resource-neutral information and communication technology. Descriptions of skills, including all properties and parameters, can also be mapped directly within the information model on the OPC UA server. Furthermore, OPC UA offers a wide range of interaction mechanisms (e.g., read, write, method call, eventing or even Pub/Sub), not only enabling skills to be monitored but also providing access to these skills for control purposes.

The VDMA has worked with the OPC Foundation to develop companion specifications for various domains within the field of mechanical and plant engineering. The majority of these specifications focus on the pure description of manufacturer and state information for applications like asset management or condition monitoring, i.e., mainly read-only access to this information. However, this type of access does not fully meet the requirements for full machine interoperability in the sense of the Industrie 4.0 vision, which is why it is important to provide write and control access to machines and plants. Initial companion specifications, such as the PackML state machine (OPC 30050) or OPC UA programs (OPC 10000-10), set out approaches for this type of access. Nevertheless, a uniform and cross-domain concept has yet to be established.

At present, it is generally necessary to establish communication for data acquisition in parallel to the classic, network-based, real-time communication interfaces at field level. This either requires additional outlay since a second network has to be set up for OPC UA or the TCP/IP-compatible part of real-time communication is used for OPC UA, which is the more common solution in practice. The latter option can lead to significant restrictions to the data throughput rate as OPC UA and real-time communication often have to share networks with maximum data transfer rates of just 100 Mbit/s. As such, the full potential of OPC UA can only be exploited if the provision of information is facilitated along with the implementation of tasks and services and direct integration in the overarching production planning and control system (MES). It is therefore important to transfer the concept of skills and capabilities to OPC UA in the form of controllable skills.
Implementierung von Skills auf der Steuerungsebene


**Figure 6:**
**Skill metamodel in OPC UA**

For this purpose, a separate OPC UA ObjectType is created with the name “SkillType”, which holds the key elements for the skill interface:
ParameterSet of the FeasibilityCheck and this is launched via a method call (start()). Tools such as simulations, decision trees or knowledge graphs are used to check whether and how the implementation of a skill is possible. After the FeasibilityCheck, the results are fed back as output parameters. The output parameters can include, for example, the time needed to execute the skill, the predicted energy consumption or the predicted manufacturing costs.

- **PreconditionCheck:**
  Shortly before a skill is executed, the optional PreconditionCheck checks whether the required resource meets all conditions. This is particularly relevant when the execution depends on a number of other factors. In the field of assembly, this process could include checking the fill level of the components store; in the field of machine tools, it could involve checking the availability of tools and their degree of wear.

- **ParameterSet:**
  The ParameterSet stores the parameters needed to execute a skill:
  
  - The LocalRuntimeID provides a client with a numerical value for identifying the skill to be executed. This can be defined by the client during the configuration of the control sequence.
  
  - The placeholder <InputParameter> is used to define any input parameters needed to execute or configure the skill. These could be position or speed specifications, which are defined by a number of individual parameters. Each <InputParameter> is organized via the FunctionalGroup InputParameters.

  To check that a skill can be executed, the requisite input parameters are written to the
The placeholder `<OutputParameter>` is used to define any output parameters received as the skill's return value. These could be current values (rotational speed, speed) from the skill execution process, which are needed, for example, for synchronization with other skills. Furthermore, it is also feasible for sensor-based skills to be created, e.g., for quality assurance, the results of which are also represented by the `<OutputParameter>` (HAMMERSTINGL 2020). Each `<OutputParameter>` is organized via the `FunctionalGroup OutputParameters`.

There are a number of different options for creating the skill interface, which differ depending on the controller manufacturer. Some controllers already enable "Nodeset2.xml" files to be input and thus allow the OPC UA server to be configured. Using this approach, instance-specific OPC UA server information models can be created for the respective machines and plants. In turn, the variables and methods set up on the OPC UA server can be linked to corresponding PLC variables by way of a binding or callback and, as a result, can be linked to the actual skill implementation.

Not all manufacturers offer the option to import XML files. Here, the PLC program structure is normally mirrored directly on the OPC UA server. Attributes or similar can be used to release the PLC's variables explicitly for OPC UA access. Here, the structure of the skill interface must be mapped by the structure of the PLC programs, the structure of the function blocks and structured data types.

To allow skills to be controlled and monitored during operation, their state must be known at all times. For this reason, a state machine is defined within the OPC UA information model; figure 7 shows the states and state transitions of this machine in simplified form. A skill can exhibit one of four basic states, which change either when a respective method call is triggered or by means of an automatic state change (SC):
• **Locked:**
The skill cannot be executed. This state is the initial state before a device has been set up, for example. Furthermore, the state is used to indicate an error in an executing resource in order to stop further action. The *Locked* state can be reached from any other state. A state transition is either triggered automatically by a machine fault (*Error*) but can also be initiated by the user using the *lock()* command. To leave the *Locked* state, a *reset()* is needed to reset the skill back to *Idle*.

• **Idle:**
The skill is ready to be executed. It has therefore been initialized and there are no errors in the machine that would prevent it being executed. *Idle* is the default state and every skill should reach this state as soon as the associated resource has been

• **Suspended:**
The execution of the skill has been temporarily interrupted. This can be triggered automatically, for example, due to process-related waiting times or can also be triggered manually by the user with *suspend()* during *Executing*. The process can be resumed by executing *unsuspend()* or can also be resumed automatically if the state was implemented due to process-related waiting. A material shortage or a jam at the output also automatically results in the *Suspended* state. If the skill execution process needs to be terminated, it can be stopped using *cleanup()* and the state switches straight back to *Idle*.

• **Executing:**
The skill is being executed. The change of state from *Idle* to *Executing* is executed by calling up *start()* and the *stop()* command ends the
skill execution process completely and the resource returns to its initial state. In this case, it is not possible to directly resume the skill.

The resource providing the skill may initially have to complete some preparations from the Idle state to start the execution process. Equally, the resource may need to be "shut down" after the skill execution process. To implement these processes, sub-states of the Executing state can be used:

- **Starting**: Describes the direct preparations for executing the skill. In the case of machines, this can involve starting up drives, for example.

- **Execute**: Describes the actual productive execution of the skill within the sub-state machine.

- **Completing**: Describes the state that exists directly after productive execution, in which the machine prepares for the subsequent Complete sub-state or ultimately for its return to the main Idle state. In the case of machines, this can involve shutting down drives, for example.

- **Complete**: The skill execution process has been completed correctly. Depending on the application, the state change from Complete to Idle can either take place via an automatic state change (SC) or a manual reset(). Maintaining the Complete state can be used, for example, to monitor the correct execution of the skill on a state basis.

This can be particularly useful for processes with long cycle times (e.g., batch processes) and if further follow-up steps are required after a successful skill execution before the skill is ready again (Idle). These steps could include, for example, manual product removal processes.

Because the individual states could mean different things depending on whether they are triggered automatically or by the client, the cause of a state change must be monitored. If, for instance, a client did not send a signal and the skill’s state is Suspended, this may mean that there is a lack of material or a jam at the output. If several clients have access to the state machine, state changes can also be monitored using events; alternatively, additional sub-states can be added to distinguish between automatic state changes and those triggered by the user or client.

A proposed implementation of a state machine is set out in figure 8 as a SkillStateMachineType. A SkillStateMachine is then created within the SkillType from figure 6.
State mapping
Since skills relate only to the actual execution of production processes, the corresponding state machine also relates solely to processes. However, disregarding the actual process execution, resources may have a number of different states and application modes. These are described in the companion specification OPC UA for Machinery Part 1 — Basic Building Blocks (OPC 40001-1).

Figure 9 shows the machine states defined in the OPC UA for Machinery.

In addition to these MachineryItemStates, a MachineryOperationMode (see figure 10) has also been defined, which mainly specifies which operating mode the machine has been set to by the user.
CAPABILITIES AND SKILLS IN PRODUCTION AUTOMATION

Figure 9: MachineryItemState from OPC UA for Machinery Part 1 — Basic Building Blocks

Figure 10: MachineryOperationMode from OPC UA for Machinery Part 1

- **None**: There is currently no MachineryOperationMode available for the MachineryItem.
- **Maintenance**: MachineryItem is set into maintenance mode with the intention to carry out maintenance or servicing activities of the MachineryItem.
- **Setup**: MachineryItem is set into setup mode with the intention to carry out setup, preparation or postprocessing activities of a production process.
- **Processing**: MachineryItem is set into processing mode with the intention to carry out the value adding activities.

Figure 11: Mapping between machine state, application mode and skill state

<table>
<thead>
<tr>
<th>Example</th>
<th>Machinery Item State</th>
<th>Machinery Operation Mode</th>
<th>Skill A State</th>
<th>Skill X State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non Scheduled Time</td>
<td>any</td>
<td>None</td>
<td>Locked</td>
<td>Locked</td>
</tr>
<tr>
<td>Unscheduled Downtime</td>
<td>Out of Service</td>
<td>Processing</td>
<td>Locked</td>
<td>Locked</td>
</tr>
<tr>
<td>Scheduled Downtime</td>
<td>Out of Service</td>
<td>Maintenance</td>
<td>Locked</td>
<td>Locked</td>
</tr>
<tr>
<td>Engineering Time</td>
<td>any</td>
<td>Setup</td>
<td>Locked</td>
<td>Locked</td>
</tr>
<tr>
<td>Standby Time</td>
<td>Not Executing</td>
<td>Processing</td>
<td>Idle / Suspended</td>
<td>Idle / Suspended / Locked</td>
</tr>
<tr>
<td>Productive Time</td>
<td>Executing</td>
<td>Processing</td>
<td>Executing</td>
<td>Locked / Idle / Suspended / Executing</td>
</tr>
</tbody>
</table>
The MachineryItemState, MachineryOperationMode and SkillState are directly dependent on one another, as the example in figure 11 demonstrates. Skill A state in figure 11 represents the state of a particular skill executed by a resource and Skill X stands for other skills provided by a resource. If the resource’s MachineryItemState is “Out of Service”, for example, no skills can be executed — regardless of the MachineryOperationMode — which is why the state of all skills must be Locked. As a prerequisite for executing any skills, the resource’s MachineryOperationMode must be “Processing” and there must be no fault (“Not Executing”). In this case, the state of all skills is either Idle or Suspended as no processes will be executed here either. If the resource has several skills that cannot be executed at the same time, the state of all other skills must be Locked as soon as one skill switches to the Executing state (or a sub-state thereof) or Suspended state.

The dependencies between the three state machines must be set up in the application accordingly and reflected in OPC UA.

Executing skills using the OPC UA skill interface

Here are a number of different ways to execute skills using the OPC UA skill interface depending on the requirements related to time response and determinism. In principle, skills can be started using a (start()) method call in the SkillStateMachine. The method call must be linked to the skill implemented in the controller, e.g., as an FB or in a proprietary programming language. Equally, it is important that the SkillStateMachine in the OPC UA skill interface always reflects the state of the skills actually implemented (see figure 4). However, since the TC/IP-based client/server connection to OPC UA does not behave in a deterministic way, this call is not suitable for synchronized processes. For this reason, concepts from the OPC UA FX (Field eXchange) task force could be used instead, in particular functional entities, which are based on OPC UA Pub/Sub and are suitable for controller-to-controller communication at field level. They are defined by the fundamental data types InputData, OutputData and ConfigurationData and can possess a corresponding “organizes” reference to other objects in the information model (see figure 12).
Figure 12: Example of using FunctionalEntities
As in the "MySubObject" shown in figure 12, a skill's input and output parameters can be linked to a functional entity to enable controller-to-controller communication to be established with OPC UA Pub/Sub on the basis of skills. Figure 13 outlines a proposal for how this could look. The functional entity's inputs and outputs are linked to the component application's ins and outs, meaning they are directly linked to the skill implemented as an FB, for example. Functional entities are also implemented on the side of the controller responsible for sequence control and are linked to the sequence program accordingly. The relevant skills can be found and linked using the skill information model and the functional entity can be used to establish direct communication between the controllers with corresponding links in the programs. The connection between Pub/Sub and the functional entity allows for much leaner and more direct communication than a method call via the client/server connection and could even facilitate real-time communication when combined with Time-Sensitive Networking (TSN). However, a skill call using a functional entity is only a proposal at this time; it was developed by a committee of experts but has yet to be validated in practice.
3.5 Approach for using capabilities and skills

Using the models and technologies set out in sections 3.1 to 3.4 for support, a number of different roles are now able to apply capabilities and skills. The following sections provide a recommended approach for how capabilities and skills can be applied in practice from the perspective of resource manufacturers, system integration engineers or mechanical engineers, and end users.

Approach from the perspective of resource manufacturers:

For resource manufacturers, it is of central importance that potential customers are provided with capability descriptions and their guarantees. It is equally important to make the skills implemented in the resources available over corresponding interfaces. The following section looks in particular at the manufacturers of individual components, such as manufacturers of axes, grippers and components of a similar size. When using capabilities and skills, manufacturers of entire machines and plants can be put on a similar level to system integration engineers. The process for these manufacturers is described in the next section.

1. In the first step, a resource manufacturer determines the capabilities its resource must fulfill. The ontology described in section 3.3 could be used for this purpose. If the resource manufacturer also offers combined resources (e.g., a pick & place machine consisting of several axes), the capabilities of both the individual resources and combined resources should be linked to the ontology. Furthermore, customers may request special capabilities that do not yet exist in the ontology. Capabilities like this can also be assigned to the resources at this point.

2. To enable suitable resources to be identified on a manufacturer-neutral basis, the next step involves the resource’s specific guarantees being extracted from the associated data sheets, for example. Normally, certain resource classes have identical capabilities (a linear drive train offers “linear movement”, for example) but differ in terms of their guarantees depending on their specific type (e.g., with regard to their payload, clearance/stroke). These guarantees are defined at type level and are offered as a guaranteed capability via the manufacturer’s product portal.

3. To now ensure a seamless transition to the commissioning phase, capabilities have to be implemented as skills. Skills are implemented on the basis of the device-specific controller — provided that the component has its own controller. Otherwise, appropriate program blocks, e.g., PLC code in the form of FBs, have to be provided to implement the skills.
4. In the final step, the skill interface is provided within OPC UA. As in the skill implementation process, the skill information model in question has to be made available directly on the component as an OPC UA server and linked to the implemented skill. Alternatively, the model is available as a node set and can be imported into the overarching controller and then linked to the standardized control code (see section 3.4).

Approach from the perspective of system integration engineers:
In the case of system integration engineers or machine manufacturers, it is important that they act with both their own customers and resource suppliers in mind. Here, the main goal is to translate customer requirements into a possible automation solution and to describe this solution on the basis of capabilities, which can then be used for resource selection and skill-based commissioning. As the intention here is to only look at differences to the traditional approach to planning, the approach will start straight from the capabilities.

1. In the first step, the potential outlined solutions or the production process required by the customer must be formulated in the form of resource-neutral required capabilities (see figure 2). Here, it is necessary to distinguish between whether the customer has used product-related or process-related capabilities. As already described in section 2.1.1, this step may require — depending on the domain — product-related required capabilities to be translated into process-related ones. One of a system integration engineer’s main tasks here is to break down and compose capabilities on the basis of the ontology. Particularly in the case of highly product-related capabilities, a number of successive or parallel capabilities may be needed to achieve the overarching task. The ontology can be used to identify which capabilities can be combined with one another (see section 3.3); an example of this would be the aforementioned Pick & place capability, which consists of several Move capabilities and a Grip capability.

2. To ensure that these capabilities actually generate added value, the next step demands that the requirements be described as accurately as possible with the help of corresponding properties. This is the only way to narrow down the selection of resources efficiently. If, for instance, a Move capability is required, a number of possible resources can be used; these can then be refined further by describing concrete requirements in the form of properties, e.g., the required motion range, levels of freedom, travel speed and accuracy.
3. In the third step, the resource-neutral required capabilities can now be used to find suitable guaranteed capabilities and their underlying resources (see figure 2) in the matching process. A pre-selection process can also take place with the help of tools — provided that the resource manufacturers offer suitable tools — or conventional requests can be used instead. One advantage here is that the resource manufacturers receive a uniform degree of detail in the requirements and also receive a solution-neutral description, which does not limit them in terms of the resources to be offered. Instead of specifications that are already solution-oriented and call, for example, for a specific type-X drive train delivering Y forces, the required capability only specifies the movement and its properties. This can be achieved by any kinematic system with any operating principle. However, to ensure that this approach works on the whole, uniform access is required to the modeled guaranteed capabilities from all resources offered on the market.

4. Once suitable resources have been identified, the specific system setup is planned and the resources are procured and then commissioned. At this point, capabilities become skills that are implemented in the resources, provided that these skills are made available over a corresponding skill interface. A cross-resource control application can then be created. As the resources have uniform interfaces, the amount of resource-specific expert knowledge required is greatly reduced, which helps to save time during commissioning. Concrete examples of commissioning resources on a skills basis are provided in section 4.

5. The final step involves summarizing the skills and aggregating them into combined skills, which can then also be provided to end users over the generic skill interface. The depth to which the end user penetrates the lower levels of the system depends on the application in question. There is the possibility that they restrict themselves exclusively to the top level of the combined skills. Equally, individual Move skills, for example, can be offered to end users at individual resource level. A suitable OPC UA information model can be used to browse these levels and address the associated skills.

**Approach from the perspective of end users:**
For end users in the manufacturing sector, capabilities and skills can mainly be used in the requirements modeling phase in their work with suppliers (e.g., system integration engineers, machine manufacturers or the automated resource manufacturers directly) or for the purposes of production control during operation. The following approach is recommended for achieving the goals set out in section 2.2:

1. **Determination of the required Capabilities based on customer requirements**
2. **Detailing the required Capabilities with their properties**
3. **Matching if Capabilities for resource selection**
4. **Starting up using Skills**
5. **Provision of Skills via a Skill Interface for usage by the end user**
1. In the first step, the process steps needed to manufacture the product must be identified and ideally transferred directly into required capabilities. To begin with, these can be at a high level and are therefore generally highly product-related (e.g., the aforementioned Mounting the hood). However, if more specific individual processes already exist from the product manufacturing process, it can make sense to describe the required capabilities at the lowest level possible on the basis of the ontology. Otherwise, it is the system integration engineer’s job to work with the end user to prepare this description as mentioned above. System integration engineers or machine manufacturers may also directly offer machines and plants with very product-related guaranteed capabilities (e.g., bespoke machines to manufacture a specific product); in this case, the capabilities do not need to be broken down any further. Equally, system integration engineers or even end users may already have templates for breaking down complex capabilities into their individual components or the information required for this purpose can be taken from the ontology.

2. In the second step, the required capabilities are handed over to the system integration engineer, who can then break these down into their individual lower-level capabilities on the basis of customer requirements, particularly in the case of larger plants (as described in the section above). The goal is to achieve direct matching with resources that are already known and in use or with new resources from manufacturers.

3. In the third step, the relevant resources are procured, which are commissioned as a bespoke solution from a system integration engineer or ordered directly from manufacturers based on successful matches.

4. In the final step, the procured resources have to be integrated into the end customer’s system. To achieve this, the implemented skills are addressed over their skill interface and incorporated into the overarching production control process. As a result, the time and effort required for integration is greatly reduced for end users as well, and production switch-points can be set in all overarching systems during production — as described in section 2.2.
4 Examples of implementing skills using OPC UA

4.1 Implementing skills using assembly technology as an example

The concept of capabilities and skills was already implemented for Automatica 2018 as part of the VDMA R+A OPC UA demonstrator (Zimmermann et al. 2019). The goal was to replicate an entire controller architecture — from the individual components to the entire machine — using skills on the basis of OPC UA. This project involved different automation component manufacturers working together to demonstrate the cross-manufacturer concept of capabilities and skills. The demonstrator depicts an assembly cell for fidget spinners, which assembles these from the basic body, ball bearings and caps.
The controller architecture with the various skill aggregation levels is shown in figure 15 and depicts the station for pressing the caps onto the fidget spinner’s ball bearings. This provides a good overview of the transition from product-specific to product-neutral skills and the consolidation of individual skills into combined skills. The entire cell’s overarching skill is “Fidget spinner assembly”. If the cell were to be sold to an end user for manufacturing fidget spinners, this skill would be addressed by the end user. The lowest level (component level) contains individual components such as axes and grippers, which offer Move, Rotate or Grip skills accordingly.

Two types of device were used here: devices with a directly integrated OPC UA server on the one hand and devices with classic digital field buses or digital I/O interfaces on the other. In the case of the latter, this meant that both skills and the skill interface in the form of an OPC UA server had to be generated in the next controller up with the aid of CODESYS. The requisite modules and a description of the offered skills with their parameters to be represented in the skill interface were provided directly by the manufacturers of the individual resources involved in this demonstrator project.
The system integration engineer’s task was then to combine the skills offered by the individual components in a way that ultimately enabled the *Fidget spinner assembly* skill to be provided. For this reason, “sub-stations” were formed first, which combined the elementary process-related skills into higher quality ones. One example here is the *Positioning* skill, which is made up of several *Linear movement* skills and the *Grip* skill. This combination of basic skills can provide various higher quality skills, which can be taken from the ontology. These skills include the *Guide* skill and the *Joining by pressing in* skill.

Together with the *Convey* skill from another sub-station, they can now be combined into the product-related *Press cap in body* skill, which is part of the top-level *Fidget spinner assembly* skill. The skills are aggregated in controllers, which address the underlying components via an OPC UA client and provide the levels above with a server with the aggregated skill. The sequence logic for ensuring skills are executed in the right order can be stored in each aggregating controller. However, individual overarching controllers can also assume responsibility for complete aggregation across all levels. While this approach would reduce the system’s modularity, it would also require far fewer controllers. Due to the client-server interaction principle, the path through the chain of clients and servers causes delays during execution.

On the whole, the demonstrator project was able to showcase a range of application scenarios:

- **Simplified planning and commissioning:** The use of capabilities in the planning phase enabled the overarching “Fidget spinner assembly” capability to be broken down further until it was possible to match them to individual resources. Once the resources were procured, the appropriate implemented skills and the OPC UA skill interface enabled the commissioning process to be shortened significantly as no proprietary control code or manufacturer-specific control interfaces had to be used.

- **Flexibility and adaptability:** Through the use of skills, it was possible to change resources quickly and adapt them to new products. This could be achieved on the basis of skill parameters on the one hand — provided that no design changes are needed for the new product variant or provided that these changes apply only to attachments (e.g., changing the gripper jaws and then re-parameterizing the Grip skill). However, this only works within the flexibility margins of the resources in question as the gripper in the example above has a finite opening width or force. On the other hand, the straightforward replacement of entire resources was also demonstrated using the example of grippers. Here, it was possible to both use grippers from different manufacturers for the same assembly task and also change grippers when, for example, a new product required a greater opening width or higher forces during the gripping process. The basic sequence in which the skills are called up remains identical; only the skill parameters have to be adjusted over the generic skill interface. The prerequisites for this are mechanical compatibility and an appropriate media supply.
4.2 Implementing skills using manufacturing technology as an example

For the example of manufacturing technology, a scenario is described in which a customer designs a unique component and manufactures it with the aid of skills. The CAD model and associated metadata are sent to a company as a quotation request. To describe the specific product, information about the quantity, material, external dimensions, form and geometric features (e.g., slots, pockets, holes, bevels) is provided. Each feature is specified by several parameters and is provided with a geometrically unique description. With the help of automated feature extraction, the product requirements can be created from the CAD file and mapped to the capabilities present in the production environment. So, a capability for producing pockets is required for the geometric feature "pocket". To simplify the matching process, it makes sense to orientate the definition of capabilities around geometric features on a product-related basis. The capabilities required to produce a feature refer to one or more skills, which have to be executed in sequence for the manufacturing process. In contrast to the capability description in the engineering process, the capability in a resource's usage life cycle is not necessarily static. The capability can be refined further and updated using historical data from production. This enables the complexity of production — which often depends on the combination of machine, tool and clamping — to be mapped.

The feasibility of a skill with the parameters dictated by the feature is checked prior to production. The FeasibilityCheck is used for this purpose; for skills in production, this check uses methods such as calculations and simulations to assess the feasibility of the manufacturing process with regard to collisions, the tools required and quality standards. The result of the FeasibilityCheck is a confirmation of an achievable trajectory, the tool and the selection of the clamping mechanism.

Further results from the FeasibilityCheck, such as the time required, expense or resource consumption, can be used as a basis for decisions when selecting a suitable resource for the later manufacturing process as well as for the creation of an automated quote. As such, the FeasibilityCheck is normally performed during the planning phase, which takes place a long time before the skill is actually executed. The FeasibilityCheck should therefore be regarded as a long-term, general assessment of feasibility. The results should be stored on an interim basis for reuse later on. Once a possible manufacturing process consisting of one or more skills has been identified, the customer receives a quote.

After the order is received, a specific manufacturing plan and timeline is generated. Before a skill is executed, the PreConditionCheck is used to query the readiness of the skills required for manufacturing. To do this, the resources check the results from a previous FeasibilityCheck for validity. A check is also carried out to see whether the correct clamping mechanism and tool are fitted in the machine, for example. If the result of the PreConditionCheck is positive, the workpiece can be loaded into the cell. When executing the skill, the resource automatically checks the data calculated in the FeasibilityCheck and PreConditionCheck for validity again and then uses this data for execution.
For implementation, a robotic arm with a mounted milling spindle is used as a machine tool. In addition, a PLC is connected as an adapter upstream of the robot controller; this PLC provides the skill interface via an OPC UA server and is responsible for various calculations and peripheral control tasks. A field bus is used for the exchange of data between the PLC and robot controller. The skills can be called up by a controller over an OPC UA client.

Multiple skills are implemented on the PLC as function blocks (milling a rectangular pocket, drilling a hole, milling a circular pocket, milling a slot). The relevant internal variables (parameters, current state, result data, etc.) are explicitly released for OPC UA access. The methods associated with the function blocks can also be called up via the OPC UA server. The FB contains a state machine for the FeasibilityCheck and a further state machine for the execution of the skill, enabling the FeasibilityCheck and skill execution to take place in parallel and independently of one another. The PreconditionCheck is implemented as a method. The state machines can be controlled using the skill interface’s OPC UA methods, whereby the state machines are also influenced by the state of the (entire) machine (e.g., if there is a fault in the machine, the skill automatically changes to the Locked state).

If the skill is in the Idle state and a client calls up the Start method, the PLC transfers the necessary parameters calculated in advance to the robotic arm. As soon as execution starts, the skill changes to the Executing state. A parameterized program for a circular pocket is run on the robot controller, similar to a CNC cycle on standard CNC machine tools. If it becomes possible to synchronize skills in real time in future (with the OPC UA FX specification), suitable skills to move the axes could also be used here.
5 Outlook

On the whole, the concept of capabilities and skills offers a great deal of potential to simplify a number of processes — from the planning phase and commissioning through to the flexible execution of production processes. Even though the concepts are already fully developed and have been validated in several demonstrators and research projects, there are still several issues to be settled. These relate particularly to the standardization of the corresponding data models for capabilities and skills, and the interaction mechanisms for controlling skills over the OPC UA skill interface. Some of these data models are already being developed, such as the capability submodel for the AAS, which is being developed as part of IDTA. At the same time, OPC UA FX is working on final concepts, such as OPC UA, which can be used for controller-to-controller communication.

However, this still raises questions relating to the latencies achievable, particularly in connection with TSN. This will have a major impact on applicability for control at field level. At the same time, ontologies need to be established that enable resource manufacturers, system integration engineers and also end users to link capabilities and skills correctly. Equally, the OPC UA information models behind the skills need to be standardized, though this publication has provided some suggestions for how to do this. If the modules mentioned can be standardized and validated at an industrial level, nothing stands in the way of the wide-scale, successful application of capabilities and skills.
Project partners / Publishing notes

VDMA e.V.
Lyoner Str. 18
60528 Frankfurt am Main

Fraunhofer-Institut für
Gießerei-, Composite- und
Verarbeitungstechnik IGCV
Am Technologiezentrum 10
86159 Augsburg

Forschungskuratorium
Maschinenbau e.V. (FKM)
Lyoner Str. 18
60528

Project leadership
Johannes Olbort, VDMA
Patrick Zimmermann, Fraunhofer IGCV

Companies, universities and associations involved
Jürgen Bock, Technische Hochschule Ingolstadt
Christian Dietrich, Otto von Guericke Universität Magdeburg
Rüdiger Fritz, SAP SE
Stephan Grimm, Siemens AG
Jesko Hermann, Technologie-Initiative SmartFactory KL e.V.
Johannes Hoos, Festo SE & Co. KG
Anna Kernspecht, Volkswagen AG
Jonathan Nussbaum, TU Kaiserslautern, WSKL
Magnus Volkmann, TU Kaiserslautern, WSKL
Andreas Wagner, TU Kaiserslautern, WSKL
Etienne Axmann, VDMA
Jakob Albert, VDMA

Layout
Gabriela Neugebauer, VDMA DesignStudio

Images
Titelbild: Shutterstock

Diagrams
VDMA
Fraunhofer IGCV

Year of publication
2022

Copyright
Fraunhofer IGCV
VDMA

Bibliography
DIETRICH ET AL. 2022

DOROFEEV & ZOITL 2018

GRIMM 2009

HAMMERSTINGL 2020

HERMANN 2021

HERMANN ET AL. 2019

IDTA 2022A

IDTA 2022B
The "Interoperable Interfaces for Intelligent Production — II4IP" project is financed with funds from the German Federal Ministry for Economic Affairs and Climate Action (BMWK).

The project is run by Forschungskuratorium Maschinenbau e.V. (FKM) in conjunction with Verband Deutscher Maschinen- und Anlagenbau e.V. (VDMA) in the period from February 2020 until January 2023.