

ROBOT CELL INTEGRATION BY MEANS OF APPLICATION-P’N’P

M. Naumann
Fraunhofer IPA
Germany

K. Wegener
Fraunhofer IPA
Germany

R. D. Schraft
Fraunhofer IPA
Germany

L. Lachello
Comau Robotics
Italy

**Speaker: Martin Naumann, Fraunhofer IPA, Nobelstr. 12, 70569 Stuttgart,
+49 711 / 970 12 91, naumann@ipa.fraunhofer.de**

Topic: Session B2 - Robots for SME manufacturing: SMErobot

Keywords: Plug’n’Produce, device descriptions, robot cell integration, robots for SMEs, intuitive programming

Abstract

For SMEs, which work typically with small to medium lot sizes, fast adaptability of robot and surrounding cell to new products and processes is much more important than small cycle times. To make this possible the programming of robots and especially robot cells and the integration of new devices into these robot cells has to be much easier than today. The approach of this paper is to offer the user of a robot cell in SME-environments an easy means of programming a cell without the need to do the configuration. Therefore, commands that represent the functionality of devices have to be offered. A concept and a first implementation are presented to describe the functionality of devices on the one hand and the required functionality of processes on the other hand. Based on this information a configuration module is able to determine what processes can be executed by the available set of devices and therefore what commands the user can choose from on a HMI.

1. Introduction

The main field of application for robots today is mass production [1]. The tasks robots have to fulfil in mass production are mostly highly repetitive and do not change over an extended period of time. Therefore, the main requirement of robots used in mass production is short cycle times. The goal of *SMErobot* [2] is to broaden the field of applications for robots from mass production to small lot size production, as it is typically encountered in small and medium sized enterprises (SMEs). Because of small lot sizes, for typical SMEs fast adaptability of robot and surrounding cell to new products and processes is much more important than small cycle times. To make this possible the programming of robots and especially robot cells and the integration of new devices into these robot cells has to be much easier than today.

2. Approach and Scope of this Paper

In the office world it is very easy to install and use new devices. For example, to install a printer to your PC, you just plug it in. The entire configuration is then done automatically and your application will offer you the service “print”. This automatic configuration is called “Plug’n’Play”. Carried forward to a production environment this would mean that you would connect e.g. a robot to a cell controller and a configuration module would offer you the service “move_to” on a HMI. Even more advanced, it could mean that you connect e.g. a robot and a gripper to a cell controller and the configuration module would recognize the synergy effect and offer you the service “pick and place”. To achieve this, the configuration module needs to know about the functionality of the connected devices and must be able to draw conclusions which services it can offer a user. Therefore, the approach pursued in this paper is based on device descriptions that are evaluated by the configuration module on order to offer services to a user. This approach can be seen in Figure 1.

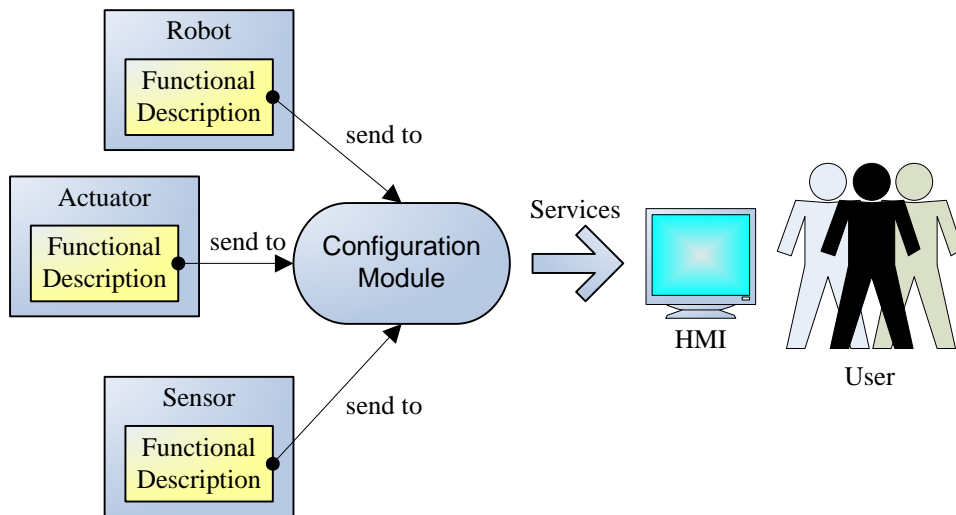


Figure 1: Approach of this paper

The ability - provided by the configuration module - to add devices to a robot cell and use these devices without the need of configuration is called “Plug’n’Produce”, according to “Plug’n’Play” in the office world. The functionality of the configuration module can be broken down into several layers that can be seen in figure 2:

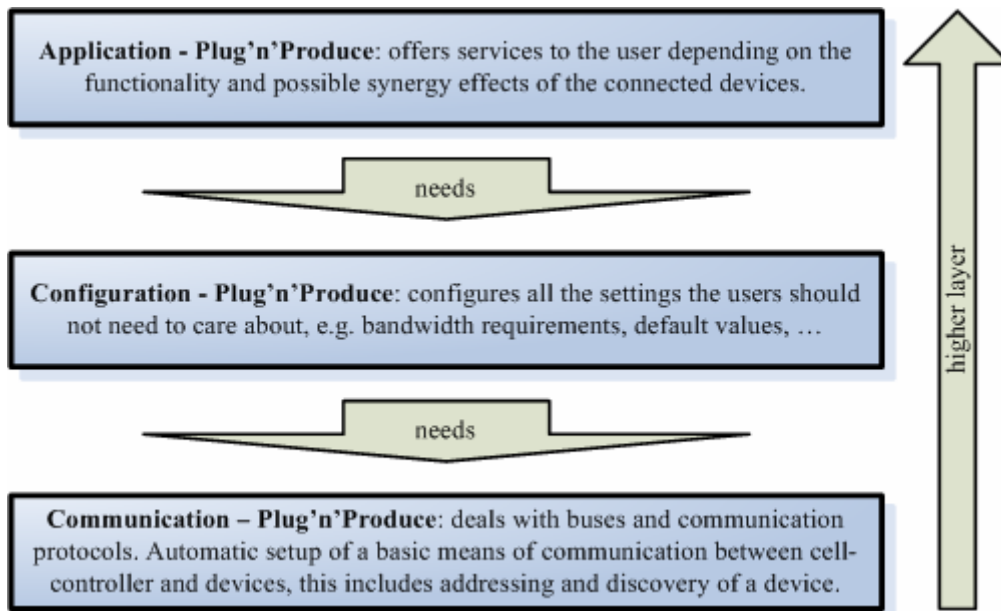


Figure 2: Plug’n’Produce-layers

To offer the user of a robot cell in SME-environments an easy means of programming a cell without the need to do the configuration, the HMI must offer the user ready-to-use commands that represent the functionality of one single or several combined devices. This corresponds to the functionality of the Application-Plug’n’Produce-layer. Therefore, the focus of this paper lies on the Application-layer. Of course, this layer depends on the Configuration- and the Communication-Plug’n’Produce-layer in order to get to know which devices are available, to communicate with these devices and to get to know the descriptions of these devices. However, the realization of the two lower layers will not be within the scope of this paper.

3. State of the Art

Studies on the state of the art of Plug’n’Produce have to cover all available technologies to integrate devices into a system. Since Plug’n’Play in the office world is long established one could think that achieving Plug’n’Produce just means using the Plug’n’Play mechanisms in an industrial environment. But because the requirements in an industrial environment differ

significantly from the ones in the office world this is not possible. In an industrial environment one has to deal with a lot of different devices from I/O-boxes to robots, bus systems, timing and bandwidth requirements and at the same time has to accomplish an industry-level robustness. Nevertheless, there exist some interesting approaches in the office world that are worth a more detailed examination. UPnP is one of them. In the industry, the challenge of integrating different devices from different vendors into one production systems has led to vendor-neutral device description languages. As one example, EDDL is examined.

Furthermore, XIRP, a XML interface for robots and periphery based on TCP/IP is worth an examination because of its intention to ease the integration of complex sensors and robots by means of a standardized communication protocol.

Universal Plug and Play (UPnP)

UPnP is a distributed, open networking architecture that leverages TCP/IP and the web technologies to support zero-configuration and automatic discovery for a breadth of device categories from a wide range of vendors. The foundation of UPnP is IP addressing. If a device has obtained an IP address the steps discovery, description, control, eventing and presentation follow. For each step dedicated protocols exist. A device that supports UPnP must support all of these protocols. Each device must contain a *device description* including vendor-specific, manufacturer information like model name and number and a list of services included into the device. For each service a separate *service description* including a list of commands and parameters must exist. Both kinds of descriptions use the XML syntax and are usually based on a standard *UPnP Device Template* according to the device category.

The main achievement of UPnP is to show a means to automatically integrate devices into a system. But, these devices must all communicate with one and the same protocol, namely non-deterministic TCP or UDP. Therefore timing-requirements cannot be guaranteed. Furthermore, the device controllers have to be able to cope with a large protocol-stack which could be a problem especially for simple devices like I/O-boxes. Nevertheless, UPnP offers good concepts to subdivide the integration of devices into steps and to describe devices and especially services a device can offer. Due to the concept of device categories, different devices of one category are exchangeable [3].

Electronic Device Description Language (EDDL)

The goal of EDDL is to describe ProfiBus-devices in a way that makes it possible to configure vendor-specific devices with one vendor-neutral software tool instead of using a different software tool for each device. Therefore, EDDL-files contain information about all variables that can be accessed via the ProfiBus communication interface, information about how to address these variables, information about how these variables should be grouped into intuitive menus by the software tool and methods to check values entered by the user of the software tool for plausibility.

The advantage of EDDL is the possibility to integrate devices with one vendor-neutral software-tool into a system instead of using different vendor-specific tools for each device, but it is not designed to allow for automatic integration of devices into one system [4, 5].

XML-based Interface for Robots and Peripherals (XIRP)

Purpose of the TCP/IP-based XIRP-interface is to reduce the effort needed to integrate peripherals from different vendors by means of standardizing the communication profile. The communication is based on a client-server architecture. The client, responsible for the sequence control, sends commands to a server, e.g. a sensor, which sends measured data in return. The communication is subdivided into three communication channels. The main communication channel is used for the above described client-server communication. A second channel is used for event-triggered messages and a third channel is used for the synchronous transmission of measured data. The commands, too, are subdivided into a group for connection-establishment, a device-profile-specific vendor-independent group of commands and a group for additional vendor-specific commands.

XIRP shows a good means to integrate TCP/IP-based devices automatically into a system and make the functionality of the devices available for the user. Strict timing requirements are fulfilled by means of an exclusive access to the Ethernet connection and the client-server architecture. Furthermore, the subdivision of the communication channel and the grouping of commands seem to be concepts worth keeping in mind.

Conclusion of State of the Art Analysis

As shown, several standards exist in the industrial environment as well as in the office world that reduce or eliminate the effort needed to integrate devices into a single system. But they rely either completely on TCP/IP or do only part of the job of integrating devices into a system up to a level that allows using the functionality of the devices without knowing much about the internal control structure of the devices. Furthermore, none of these concepts tries to represent the synergy effects possible if several devices are available. E.g., if only a robot is available it can “only” move, if only a gripper is available it can “only” fix and release. If a robot and a gripper are available, they can pick and place an object.

4. Requirements

In order to achieve Application P'n'P - that is: to offer services depending on the functionality and synergy effects of the available devices in a robot cell – the configuration module on the one hand needs to know about the functionality of these devices, on the other hand needs to know the functionality required to perform certain processes. Based on this information the configuration module must be able to determine which devices can be combined (e.g. robot and gripper) to offer new functionalities and what processes can be executed by the available set of devices. Therefore, it is required to have:

- a knowledge representation language to describe on the one hand the functionality of devices and on the other hand the required functionality of processes,
- a common semantic basis used in the device and process descriptions in order to make it possible to evaluate these descriptions,
- functions to work on these descriptions with the goal to determine devices that can be combined and to determine processes that can be executed by the available set of devices.

Predicate logic is the most general means of representing knowledge - that is: relations among objects - and therefore offers high flexibility for the device and process descriptions [8, 9]. Prolog is a logic programming language based on predicate logic that offers built-in methods to execute queries on the inserted knowledge (in this case: the device and process descriptions). Therefore Prolog was chosen to implement the Application-Plug'and'Produce-layer of the configuration module [10]. This does not mean that the other layers of the configuration module need to be implemented in Prolog as well because implementations of Prolog exist that offer interfaces to other programming languages.

5. Concept

In this chapter the structure of device descriptions and process descriptions as well as a method to determine if devices can be attached to each other and a method to determine if devices can execute certain processes are presented.

Device Descriptions

Device descriptions have the purpose of describing the functionality of devices. In order to describe the functionality of devices the concept of “*Skills*” is introduced. A *skill* is something a device can do, e.g. “MOVE_ATTACHED_DEVICE” or “ROTATE_TOOL”. Because a *skill* is a very basic and imprecise description of the functionality of a device, the concept of “*Features*” is introduced. A *feature* describes details. A *feature* of a device is e.g. “WEIGHT”. A *feature* of the skill “MOVE_ATTACHED_DEVICE” is e.g. “MOVE_WITH_6DOF”. *Features* can have “*Values*”, as it is appropriate e.g. for the feature “weight”. A *value* consists of the actual number and a unit. *Features* can be used to describe details of *skills*, of devices, but also to add details to *features* themselves. By means of this recursive mechanism the structure of device descriptions is at the same time very simple and very flexible and adaptable. Therefore, with the three concepts skill, feature, and value very basic functional descriptions as well as detailed and complex descriptions can be described.

Combination of Devices

The possibility of devices to attach them to other devices is a special functionality devices can have that can be represented by the special *skills* “CAN_ATTACH” and “CAN_BE_ATTACHED”. Two different skills are necessary because some devices offer to attach another device while other devices request to be attached to another device. For example, a robot has a flange that includes interfaces to offer power and air to an attached device while a gripper has the same flange, but needs power and air in order to operate. Therefore, the robot has the skill “CAN_ATTACH” while the gripper has the skill “CAN_BE_ATTACHED”.

The existence of these two *skills* in two devices means that these devices could be attached to each other if their flanges are compatible. Therefore, one *feature* of these *skills* must be a flange. This flange must be described in detail by means of further *features*. By matching the flange descriptions of all devices, all pairs of attachable devices can be determined. Out of these pairs of attachable devices combined devices with a new device description can be generated. These new device descriptions contain all the *skills* and *features* of the two original devices except for the *skills* “CAN_ATTACH”, “CAN_BE_ATTACHED” and all their features, because these functionalities are no longer available.

Process Descriptions

The counterparts of device descriptions are process descriptions. Process descriptions contain information about the functionality required to execute a process, the task that can be fulfilled by a process, and a sequence that describes how the process is executed. The functionality required by a process can be described by the above introduced *skills*. In process descriptions, however, *skills* do not describe functionalities offered by devices, but functionalities required to be available in order to execute a process.

Every process contains a parameterized “*Sequence*” that is executed every time the process is executed. This *sequence* consists of commands of single devices, so called “*Services*”. A *service* initiates an action in a device that corresponds to a

skill offered in the device description. It is mandatory that a device that offers a certain *skill* also has to support the corresponding *service/s*, e.g. if a robot offers the skill “MOVE_ATTACHED_DEVICE”, it also has to offer the service “MOVE_DEVICE_TO(x, y, z, yaw, pitch, roll, motype, speed ...)”.

The third element of a process description is a “**High-Level Service**”. This is the only element of a process description the user of a robot cell should ever see. It represents the command the user can choose in the cell programming environment to initiate the execution of a process. A *high-level service* contains *parameters* necessary to execute the process that are mapped to the process *sequence*.

Matching of Processes and Devices

In order to determine if a certain process can be executed by the available set of tools the functionality required to execute the process must be matched with the functionality of the available set of tools. That is, the skills of the process description must be matched with the skills of the device descriptions. If one or more device descriptions include all skills requested in the process description, the corresponding devices can execute the according process.

6. First Implementation

In figure 3 the device descriptions of a robot and a drilling tool can be seen. The robot has the *skill* “move” that corresponds to its ability to move his flange and the *skill* “can_attach” that corresponds to its ability to attach another device to its flange. The drilling tool has the *skill* “rotate_tool” that describes its ability to rotate a clamped drill and the *skill* “can_be_attached” that corresponds to its ability to be attached to another device. Both have the same flange “standard_flange”, meaning that both flanges fit geometrically. The *feature* “connections” means that the drilling tool needs to be supplied with air and 24 V and the robot can offer air, 24 V and 5 V. The robot can attach at most 20 kg and the drilling tool weighs 6 kg. Therefore, the drilling tool can be attached to the robot.

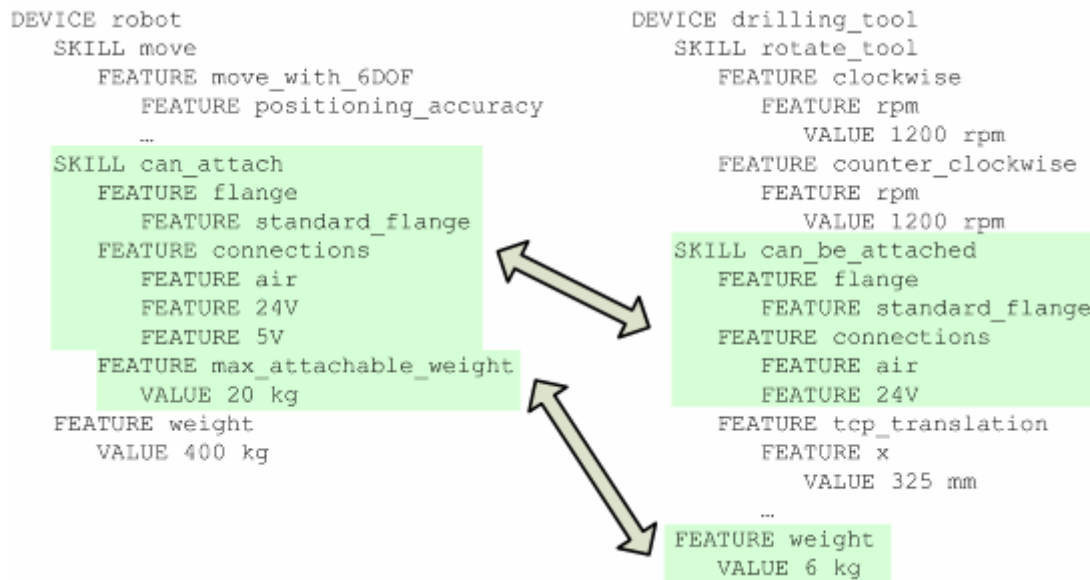


Figure 3: Device descriptions of a robot and a drilling tool

Figure 4 shows the device description of the robot and the drilling tool combined. This combined device has the *skill* “move”, inherited from the robot, and the *skill* “rotate_tool”, inherited from the drilling tool. But it has not inherited the *skills* “can_attach” and “can_be_attached” because robot and drilling tool combined can neither attach nor be attached to another device.

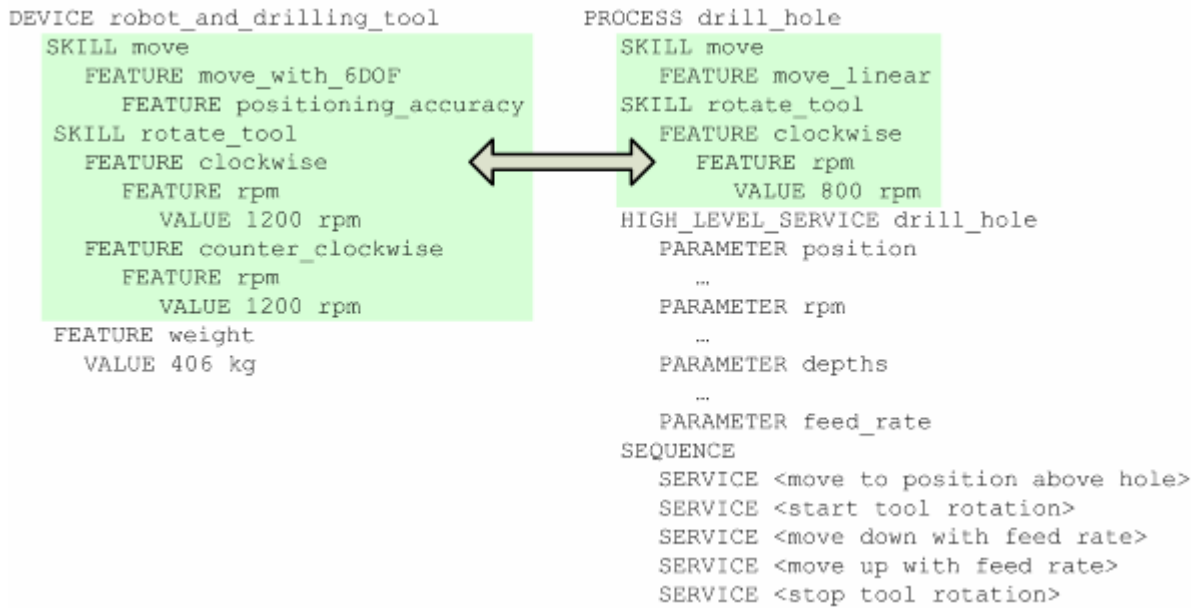


Figure 4: Generated device description of the combination of robot and drilling tool and description of the process “drill hole”

Furthermore, figure 4 shows the description of the process drill hole. This process requires the *skill* “move” with the *feature* “move_linear” and the *skill* “rotate_tool” with the *feature* “clockwise”. According to the *feature* “rpm” a tool rotation of 800 rpm is required.

If a device can offer these two *skills* and can satisfy all of the *features*, this device can execute the described *high-level-service* “drill_hole”. If a user actually calls this *high-level-service* with the *parameters* “position”, “rpm”, “depths” and “feed_rate”, the *sequence* of the process will be executed. This means, the *services* contained in the *sequence* will be called.

If you compare the skills offered by the combined device “robot_and_drilling_tool” with the skills required by the process “drill_hole”, you will see that the device offers the skills required by the process. Therefore, this - in this case combined - device can execute this process.

7. Test Bed Woodworking Cell

Fraunhofer IPA is currently under way to set up a woodworking robot cell to have a possibility to evaluate the above presented method to accomplish Application-P’n’P. This cell will consist of a robot, a cell controller including a HMI, a clamping system to fix wooden boards, a vacuum gripper to pick boards from a stack and transfer them to the clamping system, and at least a drilling and a milling tool. The gripper, drilling tool and milling tool will be equipped with tool controllers including an Ethernet-Interface. Therefore, it will be possible to implement Communication-P’n’P and Configuration-P’n’P and on top of both the above described Application-P’n’P. Thanks to the High-Level-Services offered by Application-P’n’P and an intuitive user interface the cell-user should be able to work with the cell on different applications without knowing much about the programming of the robot and the devices. Figure 5 shows a CAD-model of the cell including the above mentioned devices. Descriptions of the devices will be transferred to the configuration module residing in a PC. The configuration module evaluates the descriptions according to the concept and first implementation introduced in this paper. Finally, the HMI offers the user high-level services which represent the functionality of the available devices.

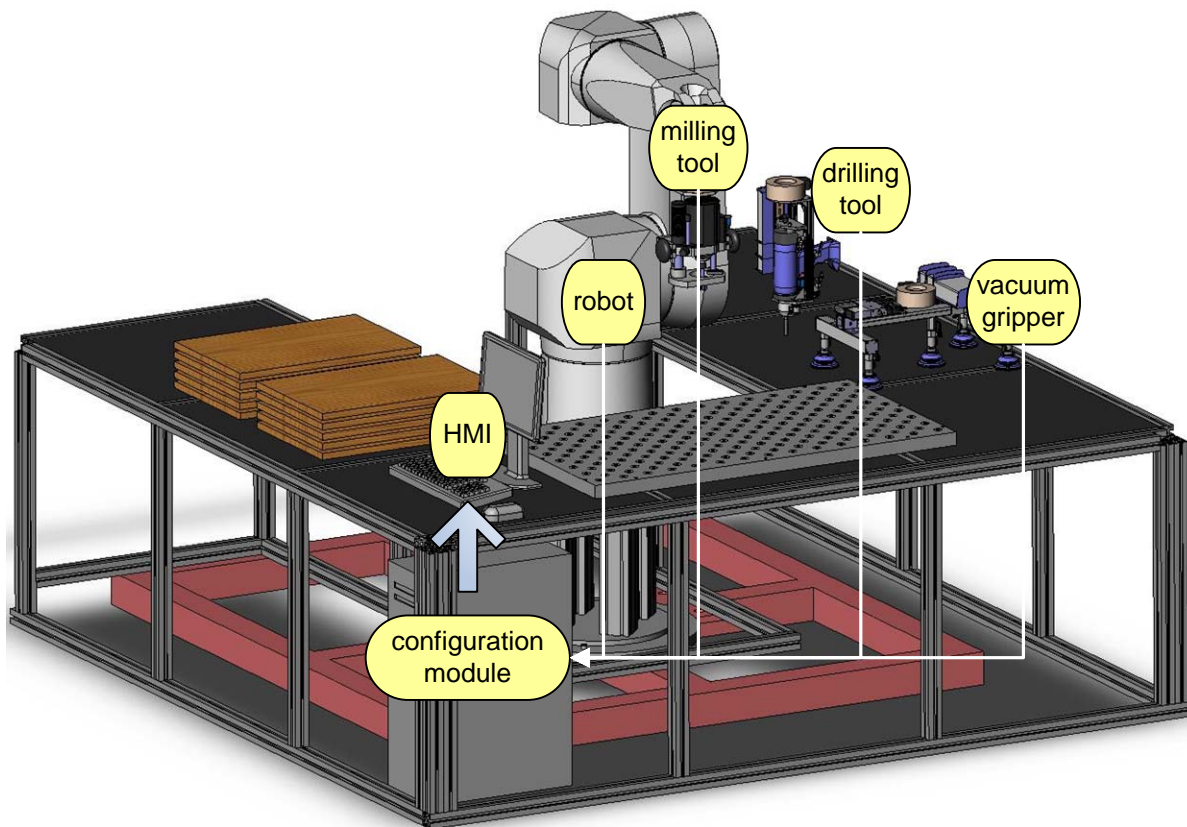


Figure 5: test bed woodworking cell

8. Conclusion

The concept presented in this paper shows a possibility to realize Application-P'n'P based on functional descriptions of devices and processes. Especially the ability to recognize synergy effects of devices distinguishes this concept from state of the art standards. This ability can make the programming of a robot cell in SMEs much easier because one command triggers a whole sequence, e.g. "pick and place", instead of only one single action, e.g. "move" or "fix". This reduces the effort and knowledge needed to program a robot cell.

At the moment the presented concept copes with not very detailed functionalities. Therefore, the next step will be to add more detail to the functional descriptions in order to better represent reality.

Acknowledgement

This work has been funded by the European Commission's Sixth Framework Programme under grant no. 011838 as part of the Integrated Project SMErobot.

References

- [1] Brodtmann, T.; Litzenberger, G.: presentation slides of "Pressegespräch VDMA Robotik + Automation", 2.3.2005.
- [2] SMErobot homepage: <http://www.smerobot.org>
- [3] UPnP Device Architecture; Version 1.0; 8.6.2000. Downloadable from the UPnP-Forum: <http://www.upnp.org>
- [4] Riedl, M.; Simon, R.; Thron, M.: EDDL – Electronic Device Description Language. München, Oldenburg Industrieverlag, 2002.
- [5] Augustin, M., Polzer, K., Ott, W.: Die Electronic Device Description Language – die Basis für die Plattformunabhängige Gerätebeschreibung. In: atp – Automatisierungstechnische Praxis 41 (1999), Heft 10, S. 24-32.
- [6] XML-basiertes Kommunikationsprotokoll für Industrieroboter und prozessorgestützte Peripheriegeräte, Stand 17.10.2005. information sheet downloadable from: <http://www.vdma.org/xirp>

- [7] Gauß, M.; Bürkle, A.; Längle, T.; Wörn, H.: An Architecture and Communication Protocol for Interaction of Industrial Robots and Vision Systems. In: Proceedings of ICAR 2003, Coimbra, Portugal, 30.6. - 3.7.2003, P. 625 – 630.
- [8] Russel, S.; Norvig, P.: Artificial Intelligence – A Modern Approach. Upper Saddle River, New Jersey, USA , Prentice Hall International, Inc., 1995.
- [9] Sowe, J.: Knowledge representation: logical, philosophical, and computational foundations. Pacific Grove, California, USA, Brooks/Cole, 2000.
- [10] Nilsson, U.; Maluszynski, J.: Logic, Programming and Prolog, 2nd Edition. Chichester, UK, John Wiley & Sons Ltd., 1995.

Curriculum Vitae of Authors

Martin Naumann, Fraunhofer IPA

Martin Naumann, born in 1978, studied Mechatronics for Production Systems at the University of Stuttgart. In 2005 he joined the department of robot systems at Fraunhofer IPA. He currently holds the position of research assistant and is involved in research in the fields of robot cell control architectures and industrial communication.

Kai Wegener, Fraunhofer IPA

Kai Wegener, born in 1974, studied Electrical Engineering at the University of applied science Giessen and Mechanical Engineering at the University of Stuttgart. Since 2000 he joined the department of robot systems at Fraunhofer IPA. His main research interests lie in the fields of distributed control architectures for intelligent end effectors and in mechanical design of flexible grippers. Since 2003 Kai Wegener is Group Manager for Handling- and Assembly systems. Currently he is working on his PhD-Thesis "Gripping of unknown objects with varying geometries".

Rolf Dieter Schraft, Fraunhofer IPA

Rolf Dieter Schraft (Prof. Dr.-Ing. Dr. h.c. mult.) is head of the Fraunhofer-Institute for Manufacturing Engineering and Automation (IPA), Stuttgart, Germany. Prof. Schraft studied Mechanical Engineering at the University of Stuttgart and graduated in 1969. In 1976 he got his Dr. degree. He started the research group for Material Handling and Assembly at the IPA which did numerous projects and developments for national and international research agencies and mainly for industry. Right now he is very much involved in promoting and developing service robots.

Luca Lachello, COMAU

Degree of of Electronic Engineer at Politecnico di Torino. He has been working in COMAU Robotics since 1989, employed in the Advanced Studies area developing software for application packages. From 1996 to 2003 he worked in the Marketing dept., where he was in contact with all main customers / robot consumers in the world. From 2004, he returned in the Engineering area, taking care of the Product Innovation & Development activities.