Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 81 (2019) 606–611

www.elsevier.com/locate/procedia

52nd CIRP Conference on Manufacturing Systems

Implementation of the MIALinx User Interface for Future Manufacturing Environments

Dominik Lucke[a,c,*], Frank Steimle[b], Emir Cuk[a], Michael Luckert[a], Matthias Schneider[a], Daniel Schel[a]

[a]*Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Nobelstraße 12, 70569 Stuttgart, Germany,*
[b] *University of Stuttgart, Institute for Parallel and Distributed Systems IPVS, Universitätsstraße 38, 70569 Stuttgart, Germany*
[c]*Hochschule Reutlingen, ESB Business School, Alteburgstr. 150, 72762 Reutlingen, Germany*

* Corresponding author. Tel.: +49-711-970-1897; fax: +49-711-970-3603. *E-mail address:* dominik.lucke@ipa.fraunhofer.de

Abstract

The flexible and easy-to-use integration of production equipment and IT systems on the shop floor becomes more and more a success factor for manufacturing to adapt rapidly to changing situations. The approach of the Manufacturing Integration Assistant (MIALinx) is to simplify this challenge. The integration steps range from integrating sensors over collecting and rule-based processing of sensor information to the execution of required actions. This paper presents the implementation of MIALinx to retrofit legacy machines for Industry 4.0 in a manufacturing environment and focus on the concept and implementation of the easy-to-use user interface as a key element.

Peer-review under responsibility of the scientific committee of the 52nd CIRP Conference on Manufacturing Systems.

Keywords: manufacturing; smart factory; Industrie 4.0; manufacturing service bus; rules; integration; user interface

1. Introduction

The increasing personalization of products and shorter product life cycles confront manufacturing companies with new challenges. To remain competitive, they must constantly change and adapt themselves.

Nomenclature

CMMS	Computerized Maintenance Management Systems
ERP	Enterprise Resource Planning
HTTP	Hyper Text Transfer Protocol
IoT	Internet of Things
IT	Information Technology
MES	Manufacturing Execution Systems
MSB	Manufacturing Service Bus
UI	User Interface

Here, the availability of up-to-date information at all levels of production is essential to implement a highly effective manufacturing. This includes technical processes, individual machine components as well as the entire production and associated business processes. Using the gathered information, the production can be controlled and optimized by workers or by self-organizing manufacturing systems. This leads to a continuous optimization over the entire life cycle of products, factories and processes.

To turn this vision into reality, the flexible and easy linking of information is essential. In the field of production, this typically includes various information sources such as MES or ERP-systems as well as a wide variety of up-to-date and legacy machines and sensors on the shop floor. Especially on the shop floor, more and more mobile sensors and sensor networks have to be included that are used to retrofit machines e.g. to gather more information for machine maintenance or to measure the

Peer-review under responsibility of the scientific committee of the 52nd CIRP Conference on Manufacturing Systems.
10.1016/j.procir.2019.03.163

product quality inline. This integration of all the different information sources can be implemented e.g. by point-to-point connections or workflow technology. It always requires an extensive knowledge of manufacturing workers on the one hand and skilled programmers able to adjust the existing solutions on the other. This means, once workers identify the need to change the environment, they depend on the IT department to implement this change. In most cases, this integration process is time consuming, expensive and impedes a rapid adaption of the manufacturing to the current situation. To overcome this challenge, a simplified and flexible integration of sensor-equipped physical assets (machines, equipment and tools) on the shop floor with the IT systems is required.

Therefore, we developed MIALinx, a lightweight and easy-to-use IT integration solution, which we introduced in previous works [1, 2, 3]. In this paper, we focus on the implementation of the user interface of MIALinx. Furthermore, we describe the application in a manufacturing environment where MIALinx enables worker to easily link and optimize manufacturing processes without special programming knowledge. The paper is structured as follows. The first sections are dedicated to the relevant context of the paper starting with the overall MIALinx concept, the implemented use case and the related work. The following sections present the aspects related to the development of the MIALinx UI.

2. MIALinx Concept

In the daily operation within a manufacturing company, many situations exist that follow an IF-THEN logic, such as “IF the last part of a production order is finished THEN report it to the ERP system” or “IF the temperature is higher than 50°C THEN turn the ventilator on.” Many parts and systems of manufacturing are controlled this way with more or less complex IF-THEN rules. The workers, foremen and managers are used to think in that logic.

In order to solve the integration challenge, we apply the IF-THEN logic to connect flexibly different IT systems, machines, sensors and via interfaces, humans. Therefore, all these assets are clustered into sensors, providing information about something, and actuators, executing something. The sensors can be physical such as sensors for electrical current and voltage or temperatures measurement but also virtual like database events or production orders. Everything that produces information in a manufacturing environment can be used as a sensor. Actuators can be physical (e.g. linear axis or signal lights) but also virtual (e.g. emails or maintenance demand message generator for CMMS or ERP systems). In general, an actuator can be everything that can consume information.

An IF-THEN rule links events from the sensors to corresponding actions of actuators. Rules can be saved, reused and executed automatically. A set of conditions is required to specify the circumstances under which the actuator is triggered and executed. For the technical implementation a so-called action configuration is required It specifies the action that should be executed by the actuator when the rule is executed, consisting of key value pairs that are send to the involved actuator. These four parts can be combined to a rule saying: “If the involved sensor sends new information and all conditions are met, then send the action configuration to the given actuator” [1]. One advantage of these rules is that they can be saved and reused and that they can be used to automatically react to occurring events. Another advantage of this approach is that people are enabled to program new rules or to adapt existing ones without any deep programming knowledge.

In the development of such a flexible and easy-to-use integration solution MIALinx follows the service-oriented architecture principle. MIALinx is split into a back-end and a front-end. The back-end comprises multiple functions realizing the technical integration of the sensors and actuators, the rule configuration, storage and execution. In order to simplify the integration and communication the implemented architecture uses a manufacturing service bus (MSB), presented in [1, 2, 3]. The MSB is a scalable service-oriented integration layer connecting the sensors and actuators via a multitude of protocol interfaces and a so-called integration flow. Simplified, the integration flow stores the connected sensors and actuators and the transforming operations. To program the integration flows a deep programming knowledge is required. To simplify this, we introduced a rule configurator, comprising the functions to create and edit a rule in a technical abstract form. It transforms the rules automatically into executable integration flows. Further functions are dedicated to the rule management (activation, deactivation, storing), rule checking for conflicts and a role-based user management. Technically, the functions are encapsulated into a HTTP interface that can be controlled by the front-end.

The front-end is the presentation layer for the rule configurator. It implements rule-modeling functions and allows the interaction with the user, which is the focus of this paper. The challenge is to find the balance between simplicity and functionality, hiding the complexity of IT systems on the one hand and providing enough functionality to cover most problems in manufacturing on the other. Here the UI is key to provide guidance and flexibility for users in the rule creation, editing and management processes.

3. Use case of retrofitting of legacy machines

The advantages of MIALinx are especially on retrofitting and upgrading of legacy machines functions, as we have here the situation to connect different IT systems and machines with different interfaces and protocols. In our use case we focus on the maintenance of air filters of machines and equipment in electrical cabinets that are often neglected. The electrical components inside a cabinet require a sufficient cooling, which is implemented with an air-cooling system. With predetermined maintenance strategies, the air filters will be usually replaced too early to avoid the risk of a failure. The potential of reducing unnecessary maintenance work and spare parts is high, as on one single shop floor often several hundred air filters are installed within the machines. Usually, they have no integrated condition monitoring system. Therefore, in this use case we upgrade the fan with a low-cost wireless condition monitoring system, measuring the air filter condition. The applied wireless sensor is IoT-capable, meaning that apart from measuring the filter condition, it has capabilities to perform simple signal processing techniques for information reduction,

and for transmitting the processed values to MIALinx. After the physical installation of the sensor, the worker can create a rule directly at the machine's location using a mobile device, for example his tablet. This rule sends, on a defined level, an automatic generated notification email to the maintenance planner. Another MIALinx rule creates a maintenance demand message for the CMMS.

4. Related work

In the area of IoT, many concepts and tools for integrating different devices have been developed, which could be used in future manufacturing environments to simplify further the integration of different systems. Web-automation-platforms like IFTTT[1] or Zapier[2] enjoy increasing popularity. These platforms can be used for easy and simple definition of IF-THEN rules. An IF-THEN rule links events to corresponding actions. Using these platforms users are able to integrate different IT systems and IoT devices without any programming skills using a graphical user interface. Some existing platforms, like Zapier, are able to integrate, e.g., Customer Relationship Management Systems. Nevertheless, they cannot be used on a shop floor as they are closed systems, which do not allow integrating arbitrary services or devices. Also, the Sensor management platforms like OpenMTC[3] or FIWARE[4] could be used for integrating arbitrary devices on the shop floor with any IT system. Although these platforms can be used to realize IF-THEN rules, they are not as flexible or easy-to-use like web-automation-platforms as they require the rules to be implemented either in source code or using a query language. While these approaches simplify the integration and programming of the immediate IF-THEN rules, they have drawbacks in the visualization and simple selection within a large number of sensors or actuators as it is required in manufacturing environments.

Several research projects have dealt with creating easy-to-use platforms that enable users with little programming knowledge to create IF-THEN rules to customize the behavior of environments. One of the first approaches was iCAP [4], which supports personalization of home appliances. They present a user interface where the user can create if-then rules by dragging items from a repository for user-defined objects, (activities, locations, people, time) into the situation area, which consists of a situation-part (if) and an action-part. iCAP does not support the user in picking a sensor or actuator based on his context. It also does not show the relations between existing rules. CharIoT is an end-user programming environment for the Internet of Things [5]. It allows the definition of so-called virtual sensors that can be used to define higher-level events over raw data. Virtual sensors can then be used in an IFTTT-like user interface to define if-then rules. The combination of the IFTTT-like interface and the hidden complexity of the virtual sensors should enable any user to easily create if-then rules. TARE is framework that allows end-users to customize the context-dependent behavior through the specification of trigger-action rules [6]. It splits the development process of rule-based environments in three phases. In the first phase professional developers set up the infrastructure of the environment. The second step consists of the definition of available triggers and actions based on the available environment on the one hand and the requirements of the end users on the other hand. Domain experts who are familiar with the application domain perform this step. In the third phase end-users use a web-based authoring tool to create trigger-action-rules. EFESTO is a tool to enable users to express rules for smart object composition [7]. The main idea is that users who want to specify customized behavior can do so by answering five questions: Who did it? What happened? When did it take place? Where did it take place? Why did it happen? (5W model). The authors of EFESTO also build a web-based authoring tool to guide the user through the rule creation process based on their model. HomeBlock is a tool created to use an Event-Condition-Action language to create intelligent scenarios, and constraints that prevent scenarios with undesirable behaviors to be applied [8]. Although this project mainly focuses on verification of scenarios, the authors also build an authoring tool to verify their approach.

After examining all the presented user interfaces, we find that none of them provides the combination of a simple rule creation process, an overview of rules and relations between them, and a user interface that can be used on every device.

5. Requirements to the MIALinx user interface

Based on the main ideas of MIALinx presented in section 2 the following derived requirements need to be addressed in the development of the MIALinx UI. The requirements to the UI have been captured during a workshop with experts of our target user group and during other industrial projects. The target user groups of the MIALinx system are skilled workers, foremen and plant managers with a high range of age, technical and social backgrounds. This defined target user group has also been used for the development of the so-called mental model of the MIALinx UI, presented in the following section. The mental model represents the logic of the UI. The captured requirements range from basic functionality over usability to organizational categories.

5.1. Basic functions

The basic functions are essential functions for a working MIALinx system comprising the creation, editing, deleting as well as activation and deactivation of rules. Also, a basic requirement is to provide an overview of existing sensors, actuators and rules.

5.2. Usability requirements

One of the main ideas of MIALinx and therefore one major requirement is to abstract the technically complex integration

[1] http://www.ifttt.com
[2] http://www.zapier.com
[3] http://www.openmtc.org
[4] https://www.fiware.org

processes of sensors and actuators. It is the key for simplifying the programming of the rules. The reason is that the target user group, such as workers, foremen or plant managers, usually has knowledge about their machines and sensors on a natural language level, such as “machine 5532”, “temperature sensor 34”. However, they do not want to deal with the technical details such as the communication protocol or data format that are required to build an integration process.

For the acceptance of a software in a shop floor context the UI shall provide a flexibility in the rule creation and editing process, because there are existing situations in which it is easier to start with the configuration of the actuator instead of the sensor. In addition, this flexibility in programming gives the user the feeling that MIALinx adapts and assists to him and not vice versa. In parallel, the created rules have to be checked for validity and the user has to be guided if something is missing or not valid. Another derived requirement of the acceptance is that already created rules have to be understandable within a short time. Implementation examples can be an IF-THEN natural language sentence or in a clear visualization.

Other requirements for the UI result from the intended use of MIALinx. The MIALinx system shall be accessible everywhere in the shop floor e.g. in front of a machine as well as in the office. Therefore, the MIALinx UI shall be usable on different devices such as PCs, tablets or smart phones. As these devices have usual different operation systems, the UI should be accessible without the installation of additional software. In addition, the UI should automatically adapt to different display sizes and resolutions (responsive user interface). Here, further requirements have to be considered: best practice of interface design, such as a sufficient contrast of fonts and elements, and different sizes of font and UI elements.

5.3. Organizational requirements

Organizational requirements result from the typical organizational structures or processes within a factory. One basic rule of security is to restrict the access of information to a minimum necessary to perform the work. Therefore, the MIALinx UI shall be able to restrict the access to sensors, actuators and rules according to the user’s role. In same context for security reasons, a reviewing process of created rules has to be established. Therefore, newly created rules are at first in a disabled condition, by default. They have to be activated as the last step of a revision process.

6. Concept of the MIALinx user interface

The UI is one of the key elements in the MIALinx concept to implement the simplicity feature resulting in the requirement of a high abstraction of programming from technically complex integration processes. In the overall concept we mainly use the idea of IF-THEN rules to solve this in the direct user interaction, but in the development of the UI we also need to address the other previously stated requirements.

For the development of the UI we use the approach of mental models. Mental models are not focused on modelling details and the complex mechanisms behind something, they model things or processes more from an overall end user perspective [9] and based on incomplete facts and simplification [10]. The mental model in our case represents the logic and the user interaction of the UI. With this approach, we enable envisioned users in the development phase to focus on the logic and interaction of the UI and not the technical details behind it.

Later, the mental model and the technical conceptual decisions such as the implementation as a web application, the programming language or frameworks are merged in the implementation phase in a so-called represented model. This final represented model is implemented using so-called UI design patterns (see section 7).

Based on the analysis of the captured demands and requirements of our envisioned target user group we identified three mental models for our MIALinx UI. The first model represents the hierarchy and principle layout of rules, sensors, actuators and parameters. The second one represents the interactions on creating a rule. The third mental model represents the interactions to search and select the desired sensor or actuator.

6.1. Rule, sensor and actuator model

Each rule has two underlying main components, a sensor and an actuator. Both have the same structure and contain one or more parameters of different types [2]. Therefore, we propose a clear layout with sensors on the left column and actuators on the right column (Figure 1). This has the reason, that usually in western culture people read from the left to the right and therefore we decided to arrange the sensor selection and configuration on the left as it is usually the first step of a rule creation process.

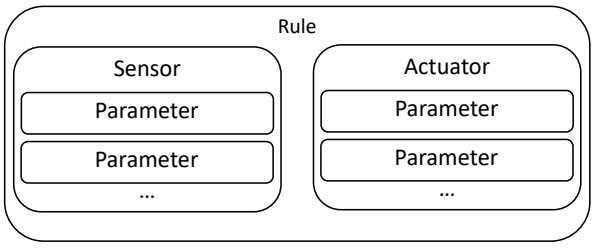
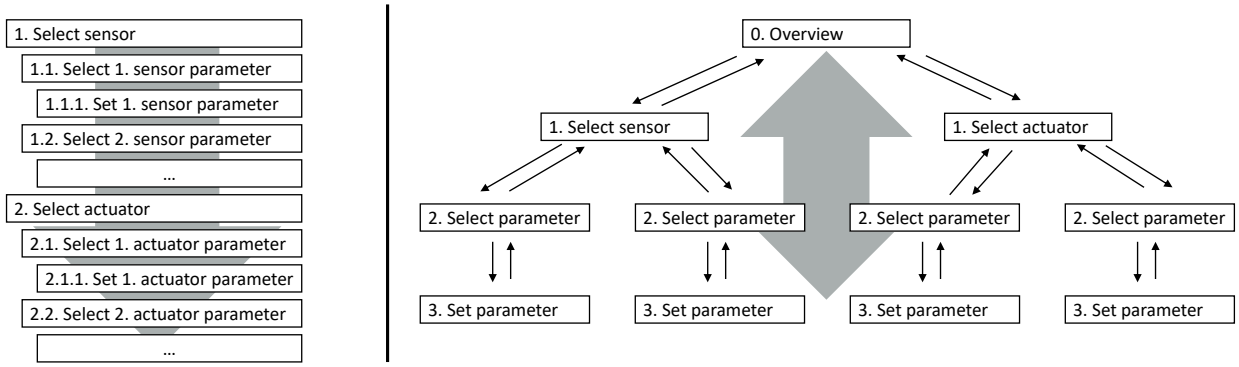


Fig. 1. Mental model of rule, sensor, actuator and parameter nesting.

In addition, to address further the requirement for an easy understanding of the rules, our mental model contains a clear hierarchy, showing the direct dependencies of involved sensors, actuators and parameter in a rule.

6.2. Rule creation model

The user starts with prior knowledge of the idea to connect sensors and actuators to one IF-THEN rule with certain conditions. He has also the mental model of the rule in mind. This is the moment where the user has to interact with the MIALinx UI. Often assistants are implemented following a static process flow providing a good guidance through the process (Figure 2a). In our case the user has to select the sensor at first, set all the sensors parameters and then do the same for the actuator. During our development phase in tests and talks



a) Static flow b) Dynamic flow

Fig. 2. Static and dynamic flow of the rule creation model.

with experts of our target user group, the demand to a more flexible way of programming arose, resulting in the corresponding requirement. To meet this requirement, we use a rule creation model with a dynamic process flow (Figure 2b). In this dynamic process flow, the creation of the rule can start with choosing either a sensor or an actuator. Once a sensor or actuator has been selected, any parameter can be configured without a fixed order. Combined with the mental model of the rule, a progress or status of the creation process can be realized. Overall, with that concept the user is encouraged to play and interact with the system instead of following rigidly the steps.

6.3. Search sensors and actuators model

Another challenging part of creating a rule is the search for the right sensor and the matching actuator. A usual method is to use a plain list, which is either alphabetically sorted or a full text search function is provided. During our development phase, in tests and talks with experts of our target user group, the wish to a better searching process arose, to provide also a hierarchical and location-based searching possibility, as the search and selection out of a plain list is to be uncluttered. Therefore, in our mental model for searching and selecting of sensors, actuators and rules, we use the approach of a hierarchical factory structure model behind, a free text search and a faceted search. The factory structure model can be visualized in a nested view and give a first orientation. A faceted search uses attributes to filter the overall list or the result list of a free text search in several steps. Examples for possible attributes are the machine type, the sensor or actuator type, or their location. A spatial filtering as a major filtering ability as a special feature of a faceted search can be achieved using a graphical map-based filter and selection as another view. An example of the top-down location structure could be the following: from building to floor to room to machine to process. Also, this searching process follows a dynamic process flow. The user can start with the nested view, free text search or faceted search and switch to the other views during the search and selection process to refine the search criteria further.

7. Implementation of the MIALinx user interface

For the implementation of the MIALinx UI concept we use the approach of UI design patterns. They are widely in use in UI design and already consider the usability requirements related to form, size, contrast and layout of UI elements and fonts. On a technical level the UI is realized as a web-based application developed using Angular.io[5]. By using Angular, we meet the requirement of device and operating system independency. The implemented design is focused but not limited to a landscape tablet format. The earlier presented mental model is the blueprint for the design of the UI. The interaction derives from the other two mental models.

In our implementation the sensors and actuators (in the following referred to as devices) are initially displayed in a list view next to each other. The user is free to change the view to a nested view of the factory structure, or to a map, which shows the location of the devices. Additionally, the user can use a text field to filter the devices e.g. by name or type. This gives the user a high degree of freedom when searching for the right device. Once the right device is selected, the selection interface disappears and the devices' parameters get visible. At this point the user still has all interaction options available. He can select the missing device, or deselect the device again, or set a parameter, as shown in the rule creation process in Figure 2b. The UI also supports user and role management in order to meet the requirements to restrict the access on information.

Applied to our use case, a typical rule creation process presents as follows. As stated earlier, the worker wants to create a rule, which automatically sends an email if a filter gets clogged. The worker decides to start by selecting the actuator first. Therefore, he uses the list, which shows devices based on the workers privileges, to find the "Send Email" Actuator, selects it and configures its parameters (address, subject and content). After selecting and configuring the actuator, the worker has to select the right sensor. Since there are many monitored filters in the factory and he does not know the name of the sensor, he switches to the map view to select the sensor (background of Figure 3). In order to reduce the items visible

[5] http://www.angular.io

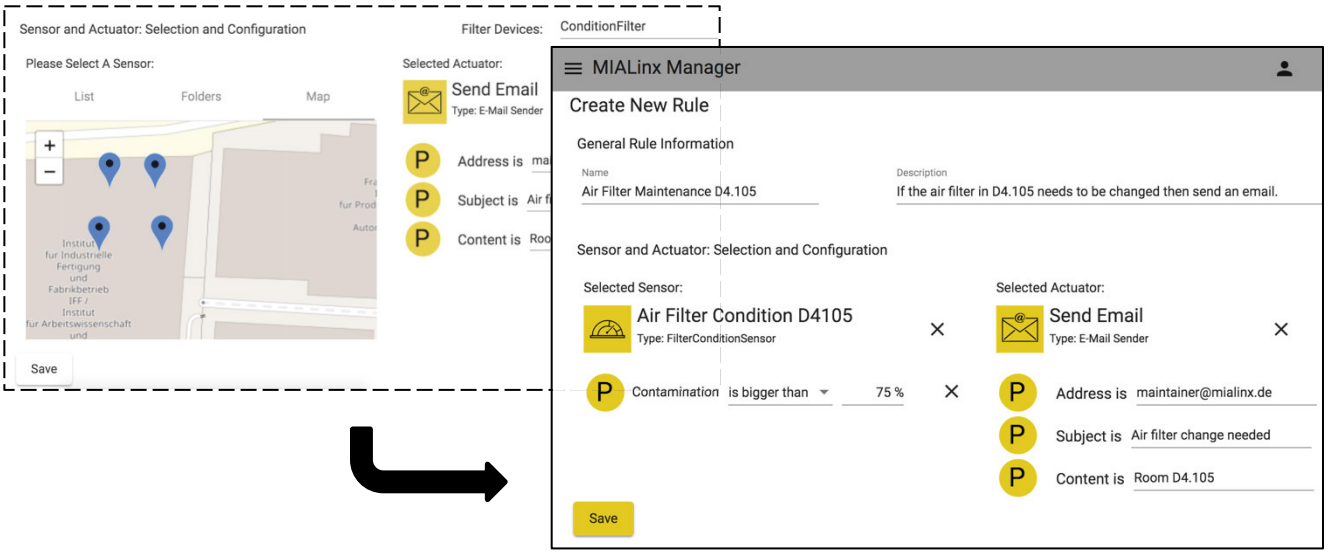


Fig. 3. Rule creation user interface.

in the map, he uses the provided filter function and filters by "Condition Filter" to find the right sensor. After the sensor selection using the map, the worker has to set the sensor parameter. He wants so specify that the rule should be executed, if the contamination of the sensor is bigger than 75%. Therefore, he has to select the parameter named "Contamination". The "bigger than" is selected by a drop-down menu and the "75%" has to be entered in a text field. If sensor and actuator are selected and configured, the user has to provide a name and a description for the rule in order to save it (Figure 3). The rule is then visible in the rule overview and can be activated. The activation is done by a different person, who has a supervising role and accounts for the rule consistency.

The MIALinx system is currently installed and tested in a plant of an industrial partner as well as in a permanent demonstrator in the future work lab, which is dedicated to the future of working places, has high visitor frequency, and the visitors themselves can get hands-on experience with future manufacturing environments. First results of the ongoing tests there show that it usually takes less than 30 seconds to create a rule after a short introduction of less than 5 minutes.

8. Conclusion and future work

In this paper, we present the concept and implementation of the user interface of the MIALinx system. MIALinx in general is a lightweight and easy-to-use IT integration solution. The user interface is one of the key elements to realize the simplicity feature. The user interface is designed for a target group of skilled workers, foremen and plant managers with a high range in age and technical and social background. In three so-called mental models we describe the logic concepts of the user interface. Theses mental models comprise the main aspects of visualization of the if-then rules, the processes to rule creation and the search and selection of sensors and actuators. The user interface is implemented as a web-based application user interface. As future work, we plan to deploy MIALinx in the domain of production information acquisition. Furthermore, the goal in future is to provide a catalog of different sets of rules for different use-cases and domains.

Acknowledgements

MIALinx is a joined research and implementation project between the Fraunhofer Institute for Manufacturing Engineering and Automation IPA and Institute for Parallel and Distributed Systems of the University of Stuttgart. It is funded by the Baden-Württemberg Stiftung gGmbH.

References

[1] Wieland M, Hirmer P, Steimle F, Gröger C, Mitschang B, Rehder E, et al. Towards a Rule-based Manufacturing Integration Assistant. Procedia CIRP 2016; 57: 213–218.
[2] Wieland M, Steimle F, Mitschang B, Lucke D, Einberger P, Schel D, et al. Rule-Based Integration of Smart Services Using the Manufacturing Service Bus. Proc. 14th IEEE Int. Conf. Ubiquitous Intell. Comput. UIC2017, Fremont, USA: 2017, p. 1–8.
[3] Lucke D, Einberger P, Schel D, Luckert M, Schneider M, Bauernhansl T, et al. Implementation of the MIALinx Integration Concept for Future Manufacturing Environments to Enable Retrofitting of Machines. In Proceedings of the CIRP-ICME 2018, Gulf of Naples, Italy, 2018.
[4] Dey AK, Sohn T, Streng S, KodamaJ. iCAP: Interactive Prototyping of Context-Aware Applications. In: Fishkin KP, Schiele B, Nixon P, Quigley A (eds) Pervasive Computing. Pervasive 2006. Lecture Notes in Computer Science 2006; 3968: 254-271.
[5] Tomlein M, Boovaraghavan S, Agarwal Y, Dey AK. CharIoT: An End-user Programming Environment for the IoT. In proceedings of the Seventh International Conference on the Internet of Things IoT 2017, New York, USA: ACM; 2017.
[6] Ghiani G, Manca M, Paterno F, Santoro C. Personalization of Context-Dependent Applications Through Trigger-Action Rules. In ACM Trans. Comput.-Hum. Interaction 2017: 24 (2):14:1-14:33.
[7] Desolda G, Ardito C, Matera M. End-User Development for the Internet of Things: EFESTO and the 5W Composition Paradigm. In: Daniel F., Gaedke M. (eds) Rapid Mashup Development Tools. RMC 2016. Communications in Computer and Information Science 2016; 696: 74-93.
[8] Guilly TL, Smedegård JH, Pedersen T, Skou A. To Do and Not to Do: Constrained Scenarios for Safe Smart House. In proceedings of International Conference on Intelligent Environments 2015, Prague, 2015, p. 17-24.
[9] Cooper A, Reimann R, Cronin D, Noessel C. About Face: The Essentials of Interaction Design, Indianapolis, USA: Wiley; 2014.
[10] Johnson-Laird PN. Mental models and human reasoning. Proceedings of the National Academy of Sciences 2010; 107 (43): 18243-18250.