City Research Online

City St George's, University of London

Citation: Houyou, A. M., Huth, H-P., Kloukinas, C., Trsek, H. & Rotondi, D. (2012). Agile manufacturing: General challenges and an IoT@Work perspective. In: UNSPECIFIED (pp. 1-7). IEEE. ISBN 978-1-4673-4735-8 doi: 10.1109/ETFA.2012.6489653

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/12635/

Link to published version: https://doi.org/10.1109/ETFA.2012.6489653

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to City Research Online policy.

City Research Online: http://openaccess.city.ac.uk/ publications@citystgeorges.ac.uk

Agile Manufacturing

General Challenges and an IoT@Work Perspective

Amine M. Houyou, Hans-Peter Huth
Siemens AG, Corporate Technology
Munich, Germany
(amine.houyou/ hans-peter.huth)@siemens.com

Henning Trsek
inIT - Institut Industrial IT
Ostwestfalen-Lippe University of Applied Sciences
Lemgo, Germany
henning.trsek@hs-owl.de

Christos Kloukinas
Department of Computing
City University London
London, United Kingdom
C.Kloukinas@city.ac.uk

Domenico Rotondi
TXT e-Solutions SpA
Valenzano, Bari, Italy
Domenico.Rotondi@TXTGroup.Com

***Abstract*— This paper describes the potential impact of the Internet of Things (IoT) technologies and architecture on factory automation. Whereas, IoT use cases range from intelligent infrastructure and smart cities to health care and shopping assistants, it is important to note that factory automation could benefit as well from an IoT approach. In this paper, we argue that there will not be one IoT but many IoTs that could differ in the type of infrastructure they are running or applications they support. In IoT@Work we focus on the potential of making manufacturing environments more agile and flexible. We explain how the IoT-centric architecture for manufacturing also needs a deep understanding of the manufacturing system and its state today. We, therefore, do a reverse engineering based on the requirements and the description of the agility expected in the automation system itself.**

Keywords-component; Internet of Things; Automation; Agility; Modular factory; Auto-configuration; Plug-and-Play.

I. INTRODUCTION

The term agile manufacturing is often interpreted differently from the various stakeholders or experts managing production systems. To some extent machine and mechanical setups could be made agile by increasing the number of combinatory sets of mechanical, electrical and software parameters a machine can have. This should make the machine adaptable to many needs and many types of end products. However, agility could also be approached in making the production system as modular as possible, with independent and intelligent components or units cooperating with each other, to create the large number of combinatory configurations. The manufacturing setup is meant to be reusable, smart, and could be shared by many production processes at the same time.

We, therefore, present the needs for automation devices and computing systems to offer and support this flexibility and agility. For this we also see one enormous hurdle to achieve this, which is in the high cost and complexity of configuring and reconfiguring the IT and communication part of the automation system. One of the promising approaches that should ease this complexity and cost problems could be based on adapting the Internet of Things (IoT) to support automation systems, in general, and more specifically, manufacturing.

The IoT [1] has recently gained a lot of interest from numerous industries, where large numbers of devices, machines, sensors, or simple things should become integral part of the Internet, open to new distributed applications and connected with IT services or the cloud. The industrial interest in IoT arises from its promise to simplify initialization and re-configuration tasks, reduce the complexity of the tasks performed by humans and lead to faster response times for the adaptations required, while at the same time minimizing configuration errors and the associated system downtime. For example, an automation expert will no longer need to assign IP addresses for new devices manually and configure the QoS characteristics of the network that they use. The challenge is to achieve this within the strict safety critical and highly expensive manufacturing environment, instead of the non-critical environments that IoT has been targeting traditionally.

The vision of IoT@Work is to be able to configure large numbers of intelligent devices in an automatic manner similar to the way you plug and play a USB stick today. The device is just plugged-in physically in the system, and to the network. The latter takes care of the rest, until the device is made ready for operation. This is one of the ways to achieve system flexibility and agility. This paper provides an example of a pilot setup to understand the constraints and limitations of today's technologies. This analysis is also used to extract requirements and key performance indicators for IoT architecture.

II. IOT-ENABLED AGILITY

Configuration of automation systems relies today on discrete engineering steps, which are carried out at the design phase using a complex engineering environment. The latter tools offer different views of the same engineered project, each view presenting a given parameter set, such as the CAD layout of the manufacturing setup, the PLC programming, the network

design, etc. The resulting engineered project details are then transferred to the physical setup in several iterative steps triggered by the engineer manually. This may in some cases lead to errors, which are very hard to locate and debug given the amount of complexity involved. The approach also assumes a tightly coupled definition of the IT services and applications running on top of the field level. The applications interaction with the production is taking place through well established gateways and filtering points, often defined in PLCs. These gateways are the only entry points to richer IT appliances or services running at enterprise (ERP) and manufacturing execution (MES) levels, which deal with the logistics of the production or materials, for instance. This leads to the pyramid model for automation depicted in Figure 1. , which explains the separation between the production system and its IT.

An IoT-enabled agile manufacturing system, in contrast, is less rigid since it should rely on autonomous units offering different resources, such as production capabilities, computing power for complex PLC logic, or other type of services in a flat and networked way. The system is characterized by:

- Highly modular production: where each module can function in an isolated manner, can be replaced or turned off, and whose hardware or software components can be easily adapted and exchanged.
- Each module is automatically configured by interacting with the network autonomously.
- Each module can be booted and configured in two main phases; (1) basic connectivity and preparation for further configuration, which includes addressing and naming steps according to local conventions and to administrative rules, in a way that a module can be uniquely identified by the configuration tools or programming environment; and (2) further parameterization of modules, where modules can receive the parameters of the applications and services to be executed.

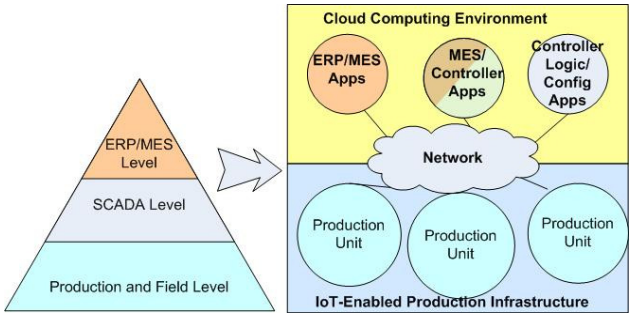

Figure 1. IoT enabled Automation

Thus the role of the network moves past from simply offering communication, to also incorporating some configuration intelligence in the form of configuration scripts, policies, discovery or negotiation protocols [4].

The following introduces some general concepts that are required for achieving agility, followed by a model factory at the Institut Industrial IT that is then used to discuss agility-related requirements and how these are met by the IoT@Work architecture.

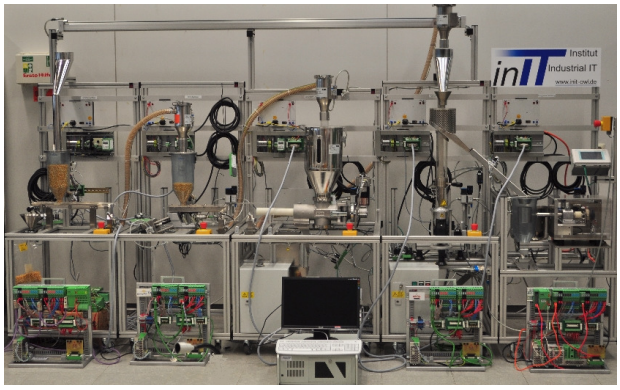

Figure 2. Setup of the LMF

A. *Example of an Agile Factory*

The Lemgo Model Factory (LMF) (see Figure 2.) is a hybrid manufacturing process, i.e. with process and factory automation elements, in a small scale. It is used at the inIT labs (Institut Industrial IT) to test new automation technologies in a real industrial manufacturing process. In its first configuration the complete LMF comprised five individual production cells. Each of these provided some functionality for processing corn seeds and producing at the end packaged pop-corn, illustrating a complete manufacturing process.

B. *Automation Aspects of Transformation*

A state of the art production facility is typically realized in a hierarchical structure. Every manufacturing cell is controlled by one main PLC that executes control software. The overall production process is controlled by a manufacturing execution system (MES). The connection between PLCs and IO devices in the manufacturing cell is realized with an industrial communication network, commonly referred to as a fieldbus or an industrial ethernet, which provides deterministic communication with real-time guarantees (See Figure 3.). In an agile manufacturing environment, the bus structure changes when new components are attached, or existing ones removed.

Such a change in the state of the production system requires two steps of engineering. These are (i) a new import of the modified fieldbus topology, including the assignment of variable names, and (ii) a modification of the PLC program to provide new sub-functionalities regarding the process to be controlled. This is caused by the fact that the software executed is always specialized to a particular production facility and its corresponding physical process.

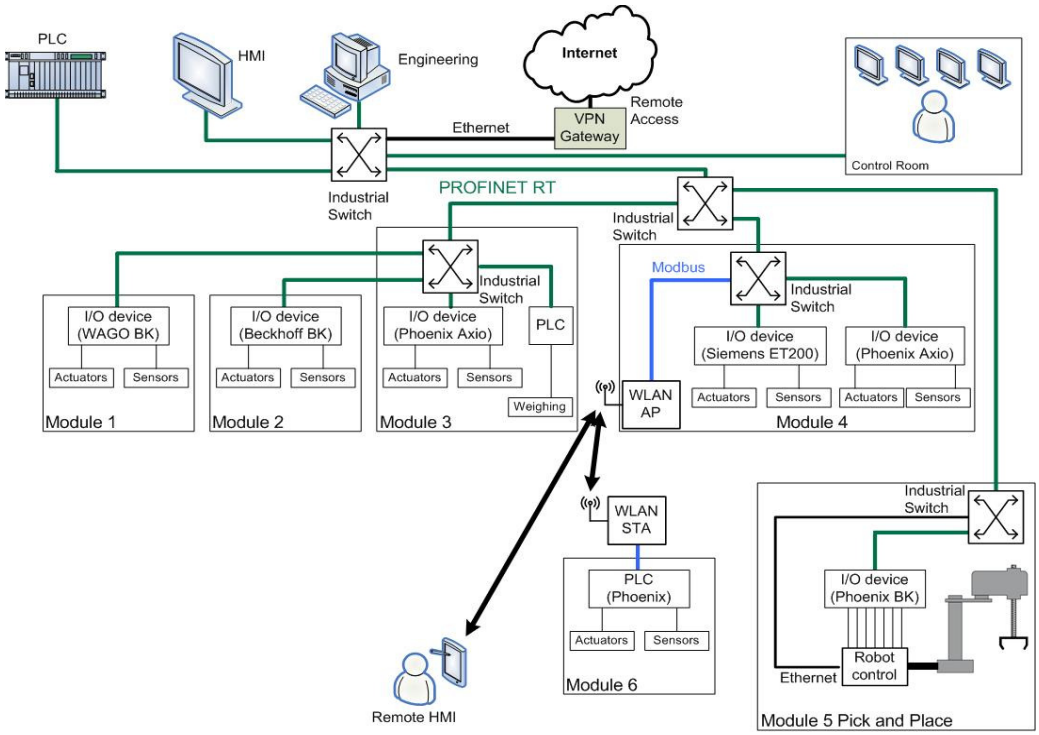

Figure 3. Network Architecture of the LMF

.

The software is implemented by an automation engineer or a technician, using hardware vendor specific tools. Typically these programming tools are based on the standard IEC 61131 [2]. This standard describes the programming methodologies of PLCs and the widely accepted usual workflow. For example, the available programming languages are defined in IEC 61131 part 3. Applying this standard helps to avoid very specialized programming languages and programming workflows that are different among PLC vendors. The typical workflow of automation software development is shown in the following list.

•Read bus configuration

•Assign IO ports to variable names

•Modify the current PLC program/project

•Compile the modified PLC program/project

•Download PLC program/project to PLC

In the first step of programming the control software of a plant, all available devices must be included in the programming tool. This is necessary to get detailed knowledge about the attached devices and to make the inputs and outputs of the all IO devices controlled by this PLC available in the programming tool. The device structure can either be automatically read by the engineering station or can be assembled manually by the programmer using graphical tools.

If all devices connected to the bus are available in the bus structure of the programming tool, variable names have to be assigned to the ports of each device. This is necessary to make data available to the PLC programs.

Since each available device provides only a general naming of its IO ports, a meaningful name is normally used to make programming easier and also easier to understand for later changes or maintenance. In addition to the variable names, also a description regarding the function of the variable can be provided. This step of engineering can take a long time, depending on the number of involved devices and the documentation of the plant structure. A detailed documentation helps associate devices with their functionality in the plant and avoids additional effort for figuring out device responsibility.

C. Analysis of the Lemgoer Modellfabrik Agility

The modules of the Lemgoer Modellfabrik have been analyzed with respect to their agility. The current results of this analysis are presented below. However, some of the items of interest are not yet clarified and have to be further investigated as this project is progressing.

1) inIT stakeholder view:

- Agility through modular process;
- Modules are moveable and can be added to an existing setup depending on production needs.

2) Technological view point:

- Module is connected through wireless network to main PLC;
- Module has its own PLC, which makes it independent of the rest of the setup;
- Agility is currently simulated and restricted through PLC/SPS programming, where variables and code are

pre-engineered and pre-configured with the exact setup of the mobile module. Currently once the new module is added and establishes its connections, the PLC program can call the process of the mobile module and trigger it to start working. This is achieved by introducing a library of function blocks for the expected component classes. A function block encompasses communication interfaces, services, data structures (object models), and can be dynamically instantiated. A lower end new component has to register to its virtual function block instance, in order to subscribe to certain services and publish its own ones. All existing components need to know the changes in the system in order to use available services and to ignore unavailable services. Initially, the manufacturing process of the modular "Lemgo Model Factory" achieved this by extending PCWorX (the utilized engineering software [7]) by a function block (AR). This function block encompasses the Profinet configuration of a single unit. Currently, a specialized firmware at the IO controller is required, that enables the PLC to sense continuously if the function block is in an active or inactive state. In case of a dynamic configuration change, this function block iss either enabled or disabled for the running project.

- Other data-centered applications like events or process monitoring have to keep track of the availability of the mobile module;
- How to trigger the event-driven application running in the local PLC and connected to a MES application? And how to link both automatically?

3) Network view point:

Wireless connection has to deliver the same reliability as wired Profinet, when used to connect an automation process.

- Real-time communication can be supported;
- Auto-configuration supported at network level, involving mapping variables of the PLC program to TCP connections. Currently the modules are hierarchical building blocks, whose complexity (variables, connections, controller/actuator/sensor interactions) is hidden from the main PLC. The number of variables that are manipulated by the central PLC is limited.
- Can mobility lead to IP domain change? For instance moving to another NAT subdomain?

4) Research view point:

- What is restricting full agility technologically?
- Can PLC programming be simplified by a more service oriented-architecture?
    - Can we encapsulate PLC software in a web service?
    - Can we enable invoking PLC services in a more service oriented manner? I.e., less detailed knowledge of the function blocks and variables of both sensors/actuators/controllers.
- Network services are pre-configured using engineering tools – does this result into configuration files that need to be made available to the mobile module in advance? Could this be replaced by an on-the-fly negotiation with the network manager that provides the communication services requested by higher layers?

III. SCENARIO-DRIVEN REQUIREMETS

Through the analysis of a number of different types of manufacturing systems the IoT@Work project [3], [8] has identified and is targeting eight high-importance general requirements.

- A-G1.Allow system modifications at run-time, if the remaining production subsystem is not affected.
- A-G2.Support fast re-initialization if the system modifications are performed at an offline system state.
- A-G3.Allow controlled initialization when modifying the system at run-time.
- A-G4.Support re-configuration of network paths in the case of path faults, through a redundancy protocol.
- A-G5.Provide an easy and autonomous system bootstrapping.
- A-G6.Support device migration through fast device re-configuration.
- A-G7.Decrease the manual effort required for configuration and re-configuration.
- A-G8.Enable the system to perform automated decision making in response to faults.

At the same time, the project has identified four high-importance requirements that are specific to automotive manufacturing.

- A-A1.Provide a graphical network configuration system, e.g., allowing drag & drop device configuration.
- A-A2.Provide a graphical network maintenance system, e.g., highlighting network faults and localizing them at the port and wire level.
- A-A3.Provide a fast and reliable semantic addressing, avoiding static IP addresses.
- A-A4.Support secure remote maintenance, potentially by external companies.

The general requirements for agility (A-G1 – A-G8) target the fast configuration of devices and network resources that is needed either when these are introduced into the system for the first time or when they need to be adapted to respond to system faults. This configuration needs to be fast when the system is offline, so as to decrease the overall system downtime, and it specifically needs to be fast when the device re-configuration is done during run-time. In the latter case it is crucial to be able to

guarantee that the production system itself will not be affected in any way – otherwise the system needs to be taken offline for safety reasons. To achieve fast configuration and to decrease the number of configuration errors, it is required that the amount of manual effort is minimized and that the system itself can automatically decide how to perform parts of the re-configuration required as a response to faults that are currently performed by human operators.

The automotive manufacturing specific agility requirements (A-A1 – A-A4) identify specific problems that the project has identified for this domain, which revolve around the current difficulties encountered with static IP addressing of devices (especially when changes need to be performed), the security of remote maintenance activities that are required to ensure the correct and safe functioning of the production machinery, and the usability of the configuration and diagnosis aspects of the system, so as to reduce the strain on the factory operators.

These requirements have an effect on a number of Key Performance Indicators (KPI) that have been established in the IoT@Work project [6]. The most important ones that are used to evaluate the proposed solutions are the following three KPIs:

- KPI-1 Downtime costs: Costs occurring due to the need of stopping the system for a period of time that is not negligible.
- KPI-2 Manpower costs: Costs of the manpower required both for the installation, configuration, and maintenance (fixed costs) and for the operation (variable costs).
- KPI-3 Number of manual steps: Number of the individual elementary manual operations that are needed to implement the feature dealt with.

By reducing KPI-3 we are able to simultaneously reduce KPI-1 and KPI-2 at the same time. This is because manual steps take longer to complete than automated steps and thus increase the system downtime required for installation, configuration, and maintenance, while also increasing the (fixed) manpower cost. By introducing automated decision making (A-G4, A-G8) all three KPIs are improved. This is because the system is able to respond quickly to situations and thus reduce the downtime, the maintenance manpower costs, and the number of manual steps required. Finally, the requirements that are specific to automotive manufacturing aid in improving KPI-2 and in particular the variable manpower costs associated with the system operation.

IV. IoT@Work Architecture Approach

The IoT@Work architecture follows a layered pattern. The layers are defined as abstractions and function groups managing the IoT infrastructure at the lowest layer, i.e. devices and networks, and the IoT applications running on top. In between these two, the function groups include management and orchestration functions that deal with configuration and execution of applications on top of resources and services offered in the IoT infrastructure. The functional grouping has resulted into separating three functional layers (shown in Figure 4.):

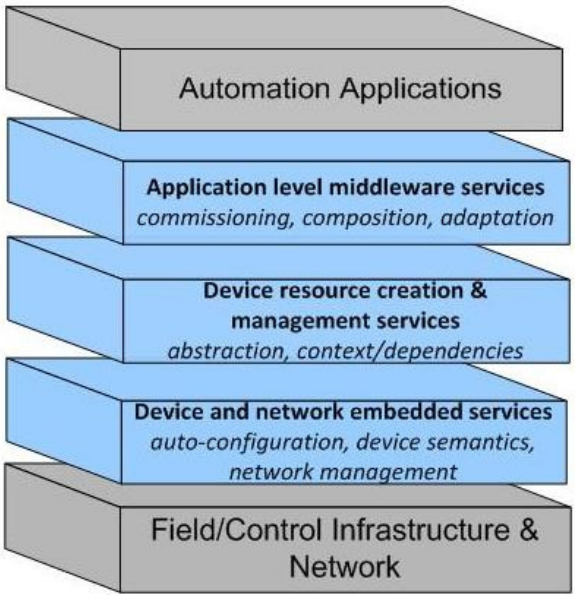

Figure 4. IoT@Work Layered Architecture

i. The device and network embedded services, which is the lowest abstraction layer of the architecture, and its associated management functions. These functions include assigning identifiers, collecting device semantics and context, managing communication interfaces, securing physical components, etc.

ii. The resource creation and management abstraction layer deals with managing embedded resources and services in an aggregated manner, hiding some of the details of single components or devices. The functions here include service directories, network abstractions, and low-level system monitoring and security management.

iii. The top layer of abstraction supports directly the application through specific middleware services targeting IoT scenarios. In the automation field, these functions include a messaging bus, application resource descriptions (e.g. requesting reliable communication or security context is interpreted here). At this layer, the application logic is interpreted at configuration or runtime, and the interfaces to the different IoT management components are defined here. Semantic reasoning functions, as well as other supporting functions can be placed here.

These functional groups roughly gather at each abstraction layer a group of technologies targeted in the project to meet functional requirements, which are detailed in the IoT@Work deliverable D1.2 [5]. This IoT-centered architecture, however, is defined within the context of automation systems. Therefore, there is a focus on those functional parts that should deliver reliable and secure communication for instance, which is required by some automation applications.

The IoT approach to embedded systems is based on the model that virtual and physical are interlinked and supported by self-organizing properties of the Internet protocols. Several functions and resources offered by embedded devices (which are a subset of smart objects or things), can be encapsulated into virtual objects that are invoked or made available to a

variety of applications and services, which contend to access and use the things, i.e., their physical and virtual resources. The IoT approach of purposing distributed embedded systems can be adapted to automation systems as well. The difference to a traditional automation system lies in the multitude of applications and their scope (they could run remotely) that can interact with the same production system. Also the way the system is configured and managed in a flexible and agile manner, offering more data and context information.

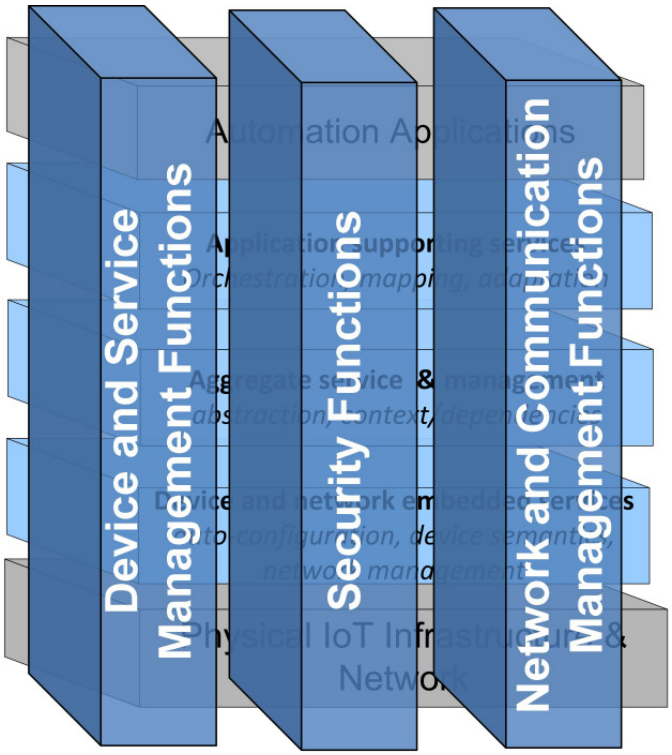

Figure 5. IoT@Work Three Cross-cutting Concerns

An important constraint of automation systems today is that they are heavily engineered and rely on detailed pre-configured physical components and protocols, making the system very rigid to changes, let alone allowing multiple services or applications to be adapted or added on the fly. An example of such detailed planning includes designing exact layout of both physical networks and logical connections or protocol parameters in the offline engineering phase. The network parameters are obtained from the communication needs of the pre-defined automated production applications. This top-down engineering of the network also prerequisites an exact choice of networking technology and protocols, so that their behaviour can be predicted and well dimensioned. This example of pre-engineering of communication solves the cross-cutting problem that involves knowing all about the infrastructure and its physical layout, but also the applications, their connectivity and non-functional requirements of reliable and deterministic communications. In a layered and well-decoupled IoT-based architecture, these cross-cutting concerns still exist and are also important for its successful adoption for critical infrastructure in general. An IoT-centered architecture has to deliver reliable communication and guarantee security, the way automation systems need it. It is, therefore, our task in the project to address these cross cutting concerns at each involved layer. These vertical cross-cutting concerns can be pictured as planes vertical to the layered architectural model presented before, as shown in Figure 5. . They include:

i. Managing networks and communication so that multiple applications can share the same network infrastructure, while delivering reliable guarantees for those applications that have high Quality-of-Service (QoS) needs.

ii. Managing system security, by making sure that no security loopholes exist in the architecture, while making sure that different management functions at each layer include some security checks and supporting mechanisms.

iii. Supporting service orchestration on top of an IoT infrastructure, while managing devices and their configurations, and linking them to applications and services running on top.

The planes specify the three technology focuses of IoT@Work. The plane demonstrates the cross-cutting concern that needs to be resolved through a specific interaction between the layers. This helps defining the interfaces of the different functions and function groups separated in the layers.

As an example, if we take the networking plane separately, it defines the way communication is realized automatically, without heavy engineering by interpreting applications' needs, on the one hand, and structuring the existing physical networks on the other hand. The lowest functions that are both used to gather the networking and QoS capabilities of underlying networking technologies, like Profinet, Ethernet or IP, are then orchestrated in network resources provided to each application according to its needs. We call this slicing the network resources. The application then sees a slice of the network rather than requiring to define each QoS parameter of a given topology such as the case in engineered Profinet. Instead, application needs could be described in profiles that classify some well understood application requirements for communication. The slices are created dynamically and on demand to fulfill a set of profile instances.

The result is that the same physical resource (i.e. the physical network) is partitioned among different application groups. Depending on the application needs and where these applications are run (e.g. distributed agent-based service points exchanging data according to some controller application logic), logical networks are created. The network slice is the mapping of that logical network to a real network infrastructure using both network virtualization and network control mechanisms that are able to protect and separate traffic of a slice from other slices, even though they might share the same or part of a physical infrastructure.

The other planes are closely related since security and device/application support is managed in parallel to communication and networking, but still closely to each other. The types of security mechanisms offered at each layer could be invoked to secure a service, a virtual network, or an end-device.

IoT@Work is not claiming to solve all issues related to IoT such as semantic interoperability or creating ontologies for all

applications or devices that exist in automation systems. However, some management functions and engineering steps could be put in sets of ontologies and described in semantics that help each configurations or error detection for instance. The project also chooses an evolutionary approach to enable some self-organization, while keeping some control over the way physical and virtual resources and services are interlinked with each other. This control is a key to trust the things to achieve not only their function but also to guarantee reliability and resilience needed in automation systems.

V. CONCLUSIONS

The Internet of Things is a concept that is hard to delimit or to define concretely. Among other things, an IoT approach to embedded systems could allow a better support for integrating semantics, context-awareness, and pieces of intelligence together, to allow self-organization when defining the interactions between things, devices, and users. We believe that this approach would allow a better integration of automation components and devices in any type of production system. It is the goal of the IoT@Work project to explore the possibilities to ease the configuration and reduce the setup cost and effort for both engineered applications and those resulting from adding or removing pieces of the production whether hardware or software. Facilitating (re-)configuration in this way is a key enabler for agile and flexible manufacturing, as is the case in the Lemgo Model Factory. The project main result is the IoT-centered architecture, whose functions and components are defined in a reverse-engineered approach. The role of the architecture is to manage the devices and services in way that complies to both IoT philosophy and to the requirements and constraints of manufacturing environments.

ACKNOWLEDGMENTS

This paper is partially supported by the EU project IoT@Work under the grant number FP7-ICT-Call5 - Contract Number ICT-257367. The authors would like to thank the whole of the project team that has worked towards these results.

REFERENCES

[1] D. Bandyopadhyay, J. Sen. Internet of Things: Applications and Challenges in Technology and Standardization. Springer International Journal of Wireless Personal Communications, Vol. 58, No. 1, pp. 49 - 69, May 2011

[2] K.H. John & M. Tiegelkamp, IEC 61131-3: Programming Industrial Automation Systems; Concepts and Programming Languages, Requirements for Programming Systems, Decision-Making Tools; Edition: 12. February 2001 - ISBN: 3-540-67752-6

[3] D. Rotondi et. al.; D1.1 State of the art and functional requirements in manufacturing and automation; IoT@Work public deliverable, 30th December 2010 https://www.iot-at-work.eu/downloads.html.

[4] A. M. Houyou et. al.; D2.2 ; General Bootstrapping Architecture , IoT@Work public deliverable, 1st June 2011 https://www.iot-at-work.eu/downloads.html.

[5] D. Rotondi et. al.; D1.2 Architecture requirements, assumptions and threats; IoT@Work restricted deliverable; 27th September 2011. https://www.iot-at-work.eu/downloads.html

[6] J. Mascolo et. al.; D4.1: Framework for economic, social, and technical validation of selected pilot; IoT@Work restricted deliverable. 20th December 2011. https://www.iot-at-work.eu/downloads.html

[7] PCWorx Engineering tool from Phoenix; http://www.phoenixcontact.com/.

[8] Amine Houyou, Hans-Peter Huth. “Internet of things at work European Project: Enabling Plug&Work in Automation Networks,” Embedded World Conference 2011, Nuremberg, Germany, February 2011.