Smart factory equipment integration through standardised OPC UA communication with companion specifications and equipment specific information models

Fabian Stoop*

Institute of Machine Tools and Manufacturing (IWF),
ETH Zürich,
Zürich, Switzerland
Email: stoopf@ethz.ch
*Corresponding author

Gerald Ely, Robert Menna and Greg Charache

TRUMPF Photonics,
Cranbury, NJ, USA
Email: gerald.ely1@trumpf.com
Email: Robert.Menna@trumpf.com
Email: Greg.Charache@trumpf.com

Thomas Gittler and Konrad Wegener

Institute of Machine Tools and Manufacturing (IWF),
ETH Zürich,
Zürich, Switzerland
Email: gittler@inspire.ethz.ch
Email: wegener@iwf.mavt.ethz.ch

Abstract: Industry 4.0 promotes digitalisation of manufacturing organisations into smart factories. Communication and connectivity of an operational data collection are required principal. OPC Unified Architecture and standards like ISA95 are key resources to develop automated interfaces between equipment and manufacturing execution system (MES). The target is to integrate complex equipment in a convenient way through the use of a companion specification, a dynamic address space and a OPC UA client characterisation process. Challenges are the reconciliation of the right programming languages, the integration of generic, domain agnostic, domain specific and equipment specific information models. This study describes the generic approach and demonstrates its applicability in a complex manufacturing environment, serving as a blueprint for other environments. The outlined concept is based on C#/LabView source-code developed for semiconductor back-end equipment, which interacts with an MES. The proposed system is validated for a final-testing-unit in the production of kW-class high-power laser diodes.

Keywords: smart factory; Open Platform Communications Unified Architecture; OPC UA; companion specification; information model; ISA95; manufacturing execution system; MES; machine connectivity.

Reference to this paper should be made as follows: Stoop, F., Ely, G., Menna, R., Charache, G., Gittler, T. and Wegener, K. (2019) 'Smart factory equipment integration through standardised OPC UA communication with companion specifications and equipment specific information models', *Int. J. Mechatronics and Manufacturing Systems*, Vol. 12, Nos. 3/4, pp.344–364.

Biographical notes: Fabian Stoop is a student at the Institute of Machine Tools and Manufacturing (IWF) at the Swiss Federal Institute of Technology (ETHZ). During his master studies in Mechanical Engineering, he worked with TRUMPF Photonics USA on implementing OPC UA both in manufacturing as well as their OEM solutions. His research is focused on both mechanical as well as connectivity aspects of machine tools and their peripherals.

Gerald Ely received his BS in Computer Science from the Richard Stockton University, Pomona, NJ in 1987. He has had many roles as a Systems Engineer and Manager over past 25+ years. He joined TRUMPF Photonics in 2015. He is currently an Automation Engineer in Equipment Development and Automation group at TRUMPF Photonics. Gerald focuses on standardising and scaling final test systems. He is interested in all things robotics and automation.

Robert Menna joined TRUMPF Photonics as an Automation Engineer after graduating from the Rochester Institute of Technology in 2013 with a BS in Electrical Engineering. At TRUMPF, he builds and programs custom automated work cells for handling and testing laser diode packages. His work has touched on everything from visual inspection and optical character recognition systems, to PLC applications, to higher level programming in object-oriented languages.

Greg Charache is the Head of Equipment Development and Automation at TRUMPF Photonics Inc., Cranbury, NJ which develops/manufactures high-power laser diode modules used for industrial manufacturing. After earning his PhD EE at Rensselaer Polytechnic Institute in 1994, he joined Knolls Atomic Laboratory working on thermo-photovoltaic devices and systems. In 2000, he joined Princeton Lightwave Inc. as a Senior Researcher for high-power semiconductor laser diodes used in the telecom industry. In 2002, the company was acquired by the TRUMPF Group where he has worked in both development and production.He has published over 75 papers and has seven patents.

Thomas Gittler works on Industry 4.0 and machine learning in manufacturing during his current doctoral studies at the Institute for Machine Tools and Manufacturing (IWF) of the Swiss Federal Institute of Technology Zurich (ETHZ). His research is focused on both connectivity, as well as prognostics, health and monitoring (PHM) applications in machine tools. He also accompanies the digital transformation of Swiss machinery manufacturers.

Konrad Wegener is the Head of the Institute for Machine Tools and Manufacturing (IWF) at the Swiss Federal Institute of Technology Zurich (ETHZ). His research is focused on machine tools, manufacturing processes and process chains, as well as the development, evaluation and optimisation of production machinery.

1 Introduction

The current demand for data acquisition and controllability in the manufacturing industry is on a rise (Monostori et al., 2016). Automatic data collection, data analysis and optimisation of the in-process decision making are expected to become key factors for economic efficiency of entire manufacturing plants (Wegener et al., 2016; Deuse et al., 2015). Predictive maintenance, self-optimising, health management and remote control are only some aspects of this Industry 4.0 movement. The foundation for strategic movement like this is built from connected, adaptive and transparent technologies (Kagermann, 2014).

Innovative approaches for machine-to-machine interfaces are the solution for communication and connectivity. The fundamental task is the integration of physical equipment in a virtual representation of the manufacturing process. This integration consists in general of two main parts, connectivity and communication. The connectivity describes the problem to have consistent data that both sides can understand. In this context, there are industry specific standards that can be used to gain some standardisation through this connectivity problem. To fulfil requirements regarding communication, Open Platform Communications Unified Architecture (OPC UA) provides a key technology. The variations and flexibilities of OPC UA goes from sensor level data acquisition to complex business-related communication like an enterprise resource planning (ERP) system. The key aspect of this technology is a platform independent, service-oriented architecture (SOA) for communication between machines and services built into one extensible framework (OPC Foundation, 2017). A single manufacturing tool now becomes part of a network of machines. The implicit difficulty of reconciling different machine types with diverse, incoherent functions is amplified by different source code languages used for the programming of machinery equipment. Joint approaches by standardisation initiatives show interesting results in the form of domain agnostic and also domain specific companion specifications. However, the task of harmonising both syntax and semantics of companion specifications with the programming languages of equipment, middleware and superior control system remains an issue of amplitude, especially in heterogeneous, complex and diverse production systems (Kagermann et al., 2013; Gittler et al., 2018).

This holds especially true for the semiconductor manufacturing industry, in which one-of-a-kind equipment for specialised tasks in industry brings new challenges to manufacturers in view of asset integration. Front-end semiconductor wafer processing tools are often equipped with standardised semiconductor equipment communications standard/generic equipment model (SECS/GEM) interfaces that allow configurable connectivity to a network (Guo et al., 2017). In contrast, machines for back-end operations (e.g., intermediary inspection, handling and stocking units, tool and consumable handling, final testing, packaging) in this industry are mostly unique to the product, have non-standard connectivity capabilities and have a limited life cycle. In addition, these back-end machines are working on complex, manufacturer-specific tasks and therefore represent a paramount impact to the quality output and efficiency of the whole production network. Therefore, aspects like transparency, interoperability and interconnection are at the foundation of the integration into superior production management systems and need to be reflected in the process and in the equipment, in order to allow for efficiency and effectiveness improvements.

This study aims at defining an integration procedure allowing to consolidate standardised protocols, generic companion specifications, domain and equipment specific information models as well as programming languages proprietary to equipment, middleware and manufacturing management systems. It describes the results of a common methodology utilised to connect one-of-a-kind back-end equipment for production of high-power laser diodes to a manufacturing execution system (MES). The regarded equipment uses LabView and C# source code, as well as similar and specific functionalities and information outputs. The integration concept regards both, differences in informational content and structure, and describes the decision-making points in the integration process. As a result, the realisation of the connections allows for gaining data in real-time process control within the MES as well as valuable data for subsequent engineering analysis.

2 State of the art

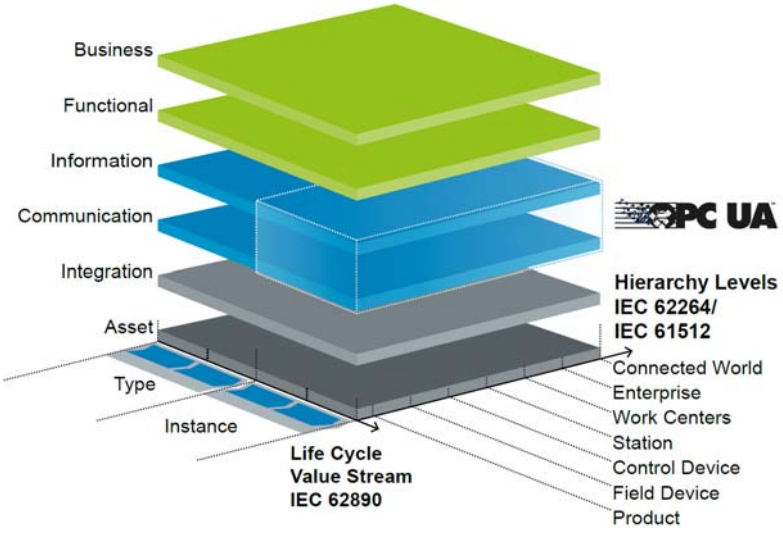
The realisation of the above-mentioned Industry 4.0 potentials in existing production lines and processes via connectivity and communication can be cumbersome. However, guidelines and reference architectures are helpful in outlining a method for this higher level of manufacturing operation and integration. The Reference Architecture Model for Industry 4.0 (RAMI 4.0) is a three-dimensional system to classify these problems and solutions in the integration task (Hankel, 2015). The model distinguishes between different views of maturities inside the life cycle and also distinguishes between a type and an instance. It also takes a look at the hierarchy levels from a product towards a connected world. The third view of stakeholders differentiates from an asset to the top business layer (Kolberg et al., 2017).

2.1 OPC unified architecture

One purpose of this model is to characterise solutions, architectures and problems in the information and communication layer. OPC UA provides solutions in the areas of communication and information, as shown in Figure 1. The communication protocol can be used over all hierarchy levels and provides functionalities for the operation of instances of components, machines and plants referring to Figure 1. The protocol is a further development of the Open Platform Communications (OPC) model. Some main features compared to the older OPC are: SOA, platform independency, scalability and many other distinctive features. This makes the OPC UA standard an enabler and a key resource in the application of the Industry 4.0 paradigm, also referred to as the transformation to the smart factory (Radziwon et al., 2014). As mentioned by Kolberg and Zühlke (2015), it is already widely accepted in different types of industries and is also still growing in its popularity compared to other protocols for specific industries. The bidirectional communication with built-in security based on TCP/IP communication are basic requirements for an open and freely available communication implementation. Based on the overall dynamics and recognition of the OPC UA standardisation ambitions, it appears a strategically sound approach to build new and complex smart factory integrations on this standard. In order to facilitate the technical implementation, software development kits (SDK) are provided by different standardisation initiatives and software integrators. These SDK are available for different programming languages like ANSI C,

.NET and Java. They provide the basic communication functions like transport, security, discovery and information access. The upper level of the stakeholders view from the OPC UA representation in RAMI4.0 (Figure 1) is the information part. This is integrated in the SDK in terms of basic information and as an extensible framework. One main challenge is to integrate this extensible information framework in the programming environment and the actual information providing/consuming application. Today this is mostly done in the language of the SDK itself and fixed in a model and the actual code. This approach is quick for the implementation of one piece of equipment. If the problem consists of a broader variety of equipment types with different source code types and information structures (Figure 2), the total integration time quickly increases manifold.

Figure 1 OPC UA in the reference architecture model Industry 4.0 (RAMI4.0) (see online version for colours)



Source: Pethig et al. (2017)

2.2 Manufacturing execution system

Another purpose of the RAMI4.0 model is to bring the functional and business layer to the level of Industry 4.0. The manufacturing control level in the functional and business part is done with an MES. This system builds the central information hub in a state of the art manufacturing company (Kletti, 2015). The tasks of the MES are described according to Verband Deutscher Ingenieure (VDI) guideline 5600 (Verein Deutscher Ingenieure e.V., 2016). The guideline defines ten tasks of an MES like order management, detailed scheduling and process control, equipment management, materials management, human resources management, data acquisition, performance analysis, quality management, information management, energy management. This overview of which task-oriented

components an MES actually has to include is also meant to compare different solutions which are available.

In order to fulfil a part of the entirety of these tasks, an MES is typically configured as a real-time system and must respond and react quicker than the actual processing time. Another requirement to achieve the smart factory attributes of the real-time data representation in the digital copy is found in the integration strategy. A common language for the dataflow from devices to the digital factory is necessary, in order to enable interoperability between the systems. A number of ontologies and information standards such as IEC 62264 and ISA95 have been developed, in order to accommodate these challenges (Shariatzadeh et al., 2016; Kolberg and Zühlke, 2015).

2.3 ISA95 companion specification

The OPC Foundation collaborates with different industries and standardisation organisations, in order to provide standardised information layers which can be used to communicate through OPC UA. An outcome from the collaboration with the International Society of Automation (ISA) is the ISA95 companion specification.

It defines a standardised interface between manufacturing equipment and enterprise control systems on an abstract level. IEC 62264 standardises the functionalities for enterprise-control system and is also based upon ANSI/ISA-95. The focus of the standard is to increase the interoperability and definition of the manufacturing control operations and the integration of enterprise level activities (Virta et al., 2010). These manufacturing operations management (MOM) systems handle a ubiquitous amount of information and are meant to cover all manufacturing processes.

The base characteristic is found in the ISA95 common object model, that is divided in four parts like personnel information, role-based equipment information, physical asset information, and material information (Ono et al., 2013).

One important part of MOM is the handling of material information. This part is also crucial in an MES system and contains actions to identify materials, check suitability for processes and recipes. The ISA95 standard provides basic types to handle materials in different MOM systems.

Another important aspect is the equipment information level. It should handle information about the equipment like identification, maintenance, capacity tracking and overall equipment effectiveness calculations. The equipment model is quite abstract and describes classes of logical devices. The physical equipment furthermore is defined by physical assets from the physical asset information model (Ono et al., 2013).

The specification is typically used to deal with information on business-to-manufacturing level and expand the basic OPC UA information model with types for this purpose. It builds the connection between the Information and functional/business like illustrated in Figure 1.

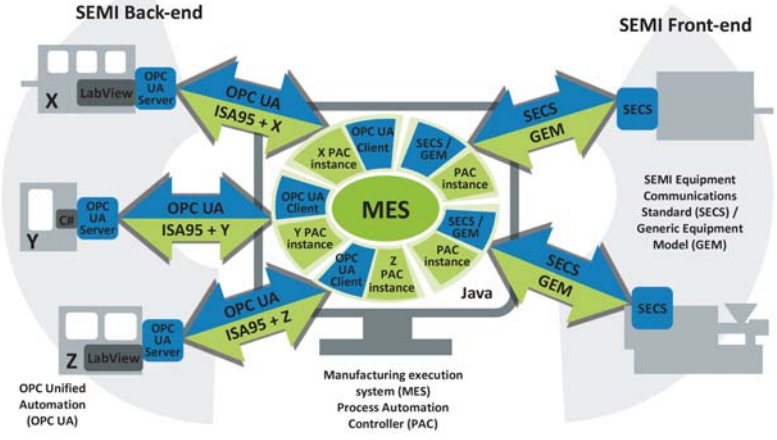
3 Equipment integration method

To achieve a sustainable and standardised communication on the business-to-Manufacturing level, some key aspects need to be considered. Initially, the implementation of connectivity on the equipment side must be easy to program at the beginning and easy to maintain during the lifecycle of the tool. Easy in this term means

no additional software development effort no a given level of implementation. On that level of implementation the dynamic address space modelling comes into effect and will act in a plug and play way. In addition, the communication itself must be reliable and sustainable for the future challenges of the plant. This reliability and sustainability is reflected in the choice of basic communication technologies together with the companion specification as mentioned in the section before.

OPC UA has the basic technical requirements to fulfil this integration. The protocol provides SOA opportunities to enhance the integration characteristics. This feature provides the foundation for task-driven intelligent equipment in an intelligent manufacturing system as pointed out by Chen et al. (2018).

Figure 2 General MES communication concept (see online version for colours)



The general MES communication concept for a generalised integration into an arbitrary production environment is illustrated in Figure 2. The physical equipment resource consists of an existing equipment application programmed in LabView, C# or any other suitable language. This application builds the transition from the real-world sensors and assets to the virtual parameters, variables and methods in the distinct source code. Another part is to include an OPC UA server in the software of the equipment. This is the virtual hook for the OPC UA communication protocol on one side and the ISA95 plus an equipment specific part on the other side. The connection is physically done over the factory network to a central MES server. This server, acting as the OPC UA client endpoint, consists of an outer middleware layer like the Process Automation Controller (PAC) and the actual MES application. The middleware layer itself consists of a generic OPC UA client and a specific PAC instance.

The PAC software also consists of other service adapters to communicate with factory floor equipment. One of these is the SECS / GEM adapter for other SEMI equipment. Equipment specially from the semiconductor front-end typically includes this adapter. TheSECS/GEM communication protocol is very industry specific and has less

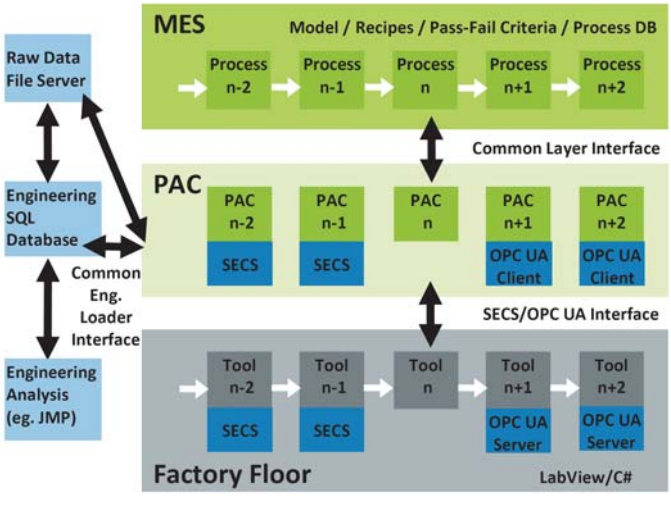
features in interoperability, open standards and the extended information model compared to OPC UA.

3.1 Shop floor integration in Industry 4.0

Smart factory integration in semiconductor manufacturing, and in particular high-power diode-laser manufacturing, must master many challenges, including: batch and single-piece flow, lot splitting and recombining, re-entrant flows, qualification lots, long cycle times, long material lead times and multiple communication platforms. Thus, implementations in this industry are likely to be applicable to others as well (Cemernek et al., 2017). Also, the back-end equipment is regarded as special purpose machinery or one-of-a-kind manufacturing units, bringing a considerable diversity to the communication integration requirements. If an equipment integration concept suits this broad variety of requirements, it is also applicable for easier tasks of machine-to-machine communication.

The integration needs to reflect the following aspects and focus on these main facets like Flexibility in data structure and accommodation of changes, Robustness and reliability of transport, Interoperability between different systems, Sustainability of used technologies, and scalability on plant level.

Figure 3 Overall MES integration concept for back- and front-end (see online version for colours)



The equipment on the factory floor level needs to form a communication network with the MES as shown in Figure 3. The material and products flow through the process as indicated with the white arrows. A communication path from the MES to every equipment as shown with the vertical arrows, is the general framework for this setup. For example, test data from the tools are routed to the MES for real-time process control and

are also archived in an Engineering SQL database or file server for subsequent engineering analysis. Examples of raw data include camera images and spectral data that are too large to store in an SQL database. In addition, tool recipes for each product can be stored within the MES and subsequently transferred to the production tool immediately prior to lot processing.

The connectivity in this network needs to fulfil numerous integration aspects, so that the interface contains nodes from business fields like production process planning, maintenance and development. Aspects of production process planning include: recipe download, decision making steps like pass/fail criteria, or scheduling and controlling of different production steps. Maintenance and development information should reflect basic information of the equipment which could be analysed with higher level systems like an MES.

The integration must consider that one-of-a-kind manufacturing units like some back-end equipment need to be included. These highly specialised units have different needs for connectivity but must also be consistent in terms of integration. Shop floor integration needs to fulfil both long- and short-term business requirements. To get the connectivity and communication ready for this task, standardised components and key technologies must be used whenever possible.

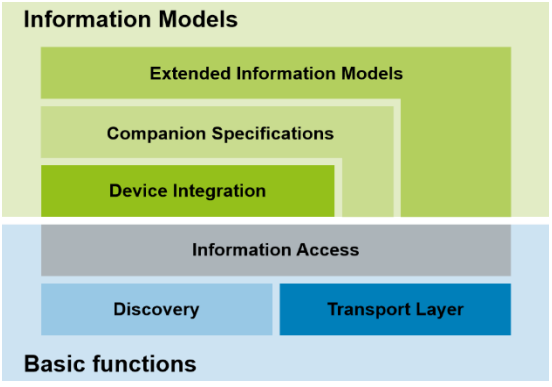
3.2 Dynamic address space modelling

OPC UA provides different ways to develop a service application. Some of these aspects, like the basic functions in Figure 4, are mostly handled in the SDK and are not implementation specifically. Others like the extended information model are very specific to an individual piece of equipment. In the end, the implementation on the equipment must be connected to the machine source code. The connectivity of the interface is reflected in this code and therefore in the associated programming language. If there is an OPC UA SDK available for the programming language of the equipment, it is highly recommended to make use of it in order to benefit from efficiency gains in the process. For example, in the case of the object-oriented language C# .NET there is a kit from Unified Automation available, which is used directly in the machine code.

A widely-used programming language for prototypes and one-of-a-kind machines is the graphical language LabView. The integration of OPC UA communication in LabView can be realised in different ways. For basic applications with limited requirements and less complex information structures, the LabView OPC UA Toolkit serves as a simple entry point. This toolkit provides all the basic functions of OPC UA and leads to short development and integration times. However, not all possibilities of the communication standard are currently supported. The object-oriented paradigm of type definition and object instances are not represented. However, this concept is useful to solve complex data structures and accomplish connectivity requirements like standardisation and interoperability. The encapsulation of this and other conceptual OPC UA functions is also intended for LabView applications. This problem is approached with a dynamic address space and an underlying companion specification. Since LabView follows the dataflow paradigm, the initialisation and maintaining of objects and its instances can be cumbersome. The main objects in an OPC UA communication are called nodes. These nodes are initialised in the SDK part like in Figure 5 of the software layer. The main challenge for the communication is the handling and administration of these nodes and objects through LabView. In order to cope with the LabView constraints

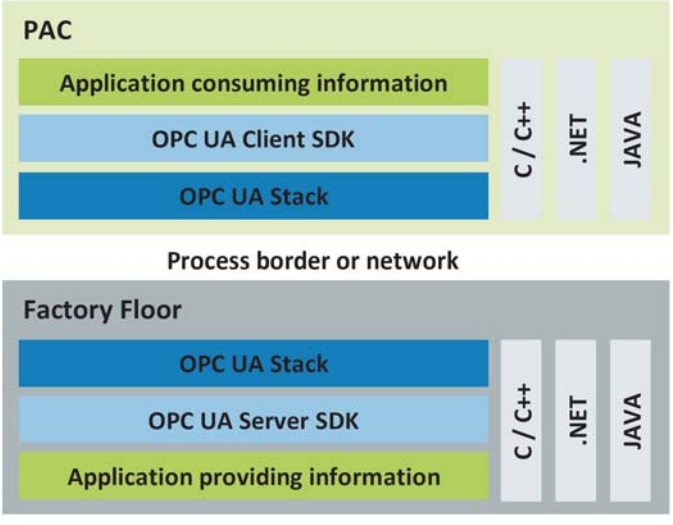
while allowing for simple and generic object integration, an equipment generic address space interface from the SDK was conceived and implemented. After this step, the SDK accommodates LabView data objects and publishes these combined with a set of parameters to the OPC UA address space. This allows the equipment developer to initialise, manipulate, maintain and delete complex data structures directly via the dataflow language.

Figure 4 Logical layer of OPC UA (see online version for colours)



Source: Pethig et al. (2017)

Figure 5 OPC UA software layers (see online version for colours)



This combination of setup and operational tasks within a single language and framework significantly reduces overall system complexity and administration efforts, whilst improving significantly asset interoperability and ease of integration. This dynamic behaviour, towards a proprietary, graphical programming language like LabView is also possible to be easily applied in other OPC UA setups. If it is not possible to use an OPC UA SDK directly in the programming environment, a generic application internal server interface is used to access the address space. This interface should maintain as much as possible from all the OPC UA functions and characteristics. However, it is also necessary to make certain restrictions to the new caller in another programming environment.

4 Implementation/integration

The implementation is split up in two main parts. One is the factory floor with the equipment side and the OPC UA server (Figure 3). The other is the MES middleware layer PAC with the OPC UA client. The PAC instance is also connected through a common layer interface to the actual process integration in MES. The general design goals for both sides are listed as automatic information model processing, use of companion specification layers, simple and secure access from/to the underlying system, maintenance and reconfiguration possibilities on both sides.

These goals need to meet the requirements from the dynamic environment in which back-end equipment is developed. The goals need to be realised and also be included in all the software layers like shown in Figure 5. The most inner layers like the OPC UA stack or the OPC UA server/client SDK are used for basic communication and for the implementation of the information models. The application specific layer is utilised to interact with the different objects of OPC UA and to connect them to the appropriate place in the application of the underlying system.

Maintenance is an important aspect of the whole communication setup. Especially the application layer and the information model should be able to adapt to changes fast, easy and economically. This is achieved with a flexible and dynamic interface for C# / LabView and a Model-driven Architecture (MDA) approach on the client side. This approach and especially the characterisation on the client side also highly satisfies the requirements of an easy to program and maintain connectivity structure. The concept is based on object oriented or LabView data types and provides this generic functions through the whole stream of connectivity.

4.1 MES and PAC

The Siemens CAMSTAR (Siemens Product Lifecycle Management Software Inc., 2019) MES with the Semiconductor Suite is designed and used for this front- and back-end manufacturing task. The MES provides on one side the interoperability with the higher ERP business systems like SAP and on the other side solutions for equipment integration like the process automation controller (PAC). The PAC is a middleware layer for this MES and integrated as shown in Figure 2. The PAC as the Middleware contains only communications and connection logics, but no business logic like the MES. It is designed to work with different communication technologies and provides data interfaces also suited for the SEMI industry. These are recipe management, equipment engineering and process control systems (znt Zentren für Neue Technologien GmbH, 2017). PAC also

consists of a broad variety of external service adapters like Camstar, SECS, file transfer protocol (FTP), simple mail transfer protocol (SMTP), transmission control protocol (TCP), Modbus, OPC data access (OPC DA), OPC UA, discovery and basic configuration protocol (DCP) and some others.

The MES itself contains the main process models, recipes and pass/fail criteria, are passed to the machine, or compared to the feedback data of the machines. The operators on the machines will track new products in on the MES interface, via a terminal GUI at the workplace. Based on the worker interaction or some other trigger, the MES hands over the recipe and other product specific information to the production equipment. After the physical process is completed, the accumulated data and other live information are sent back to MES. The outcome is a virtual replication of the physical process in the system. The centralised role of the MES enables it to assume a mastermind role in manufacturing process, covering all aspects of control, communication and traceability.

4.2 OPC UA server and LabView

The OPC UA server side on the equipment consist of two main parts, the Unified Automation OPC UA .NET Server SDK library and the LabView interface to this library.

The first part uses a design tool like the Unified Automation UaModeler, generic classes and a dynamic address space as mentioned in Palonen (2010). The design tool is used to integrate the ISA95 companion model on one side and the generic classes on the other side. The generic classes are subtypes of the companion specification and are related to the actual business of the plant. In the case of a semiconductor back-end plant, the implementation contains special types for material and equipment information. The output of the design tool is the code basis for the further development. The main advantages of the UaModeler design tool in this design phase are in the quick and simple implementation via the graphical designer, as well as the automated output of reliable and error-free code, which helps to accelerate overall integration.

The next step builds on this base OPC UA server code and connects the .NET application to a dynamic link library (dll) which afterwards is used in the LabView environment. The library should also include the methods and variables that create the information model instance. These instances are afterwards accessed in LabView and the actual OPC UA functions like read, write, method, event, etc. are used. In this sense, the dynamic link library consists of plant generic and companion specification information which are accessed in a predefined way from the actual equipment code like LabView.

This static implementation in the model and the .NET application needs to be realised in a careful manner. It should always be intended to include the largest possible amount of the generic equipment characteristics in this part, in order to obtain one single library covering the entirety of all the equipment. However, this ideal condition is not always suitable. In some cases, it is necessary to maintain OPC UA types in the static part. This should be used only in case of radical modifications. It also allows clients to retain the necessary agility to self-implement changes and reconfigurations in the structure (Harju, 2015).

The dynamic part of the integration is implemented in LabView code. The OPC UA server service is called through the dll in LabView. After the basic startup and the initialisation of the static part, the dynamic part is called. The control in LabView is done with simple data objects or with clusters like shown in Figure 10. A cluster contains a group of data elements from mixed types. This is useful to create a bigger data structure

in the OPC UA address space and afterwards maintain the information of these. The server is dynamically accessible from the underlying system and provides the benefits of a companion specification to the client.

4.3 OPC UA server and C#

The implementation of the OPC UA server is also possible directly in C# .NET equipment application. The source code and SDK are directly available in the application and accessible in the same programming language, which enables a larger range of integration options.

The first step in the UaModeler and the code generation with ISA95 remains the same as for LabView. After the code is generated, the output in form of C# classes and objects is immediately useable for example in the Visual Studio environment.

The server initialisation of the equipment specific part is similar to the integration in LabView. Because of the same programming language, the C# data objects are passed to the server with only little adjustment and get published. It is also easier to maintain these data objects directly from the main program itself. Also, the handling of other OPC UA functions like methods, events, alarms and historical access are possible in a generic way directly in the application. This makes the initial integration and the maintenance comparably simpler in regard to the LabView application. However, it is not always possible to choose this language and a generic interface to another language is necessary.

4.4 Implementation in the existing equipment application

Today, the equipment integration for smart factories is still at the starting ground of the journey. One of the major, and potentially the main challenge is in the integration of already existing equipment and their applications. The major requirement for this task is to have the full source code of the equipment app available. Furthermore, the existing inputs and outputs of the program need to be known for appropriate information provision. In most cases the main input and control comes from a graphical user interface (GUI). The main output is in most cases to a log file or a database.

The integration of the server in the existing application should consider one important aspect: Will the integrated equipment afterwards still require a proprietary GUI or will it be controlled remotely through an application programming interface (API)? If the GUI is not needed, because the user interaction is with the MES, the server becomes the main interface for the application, the OPC UA server becomes the main functional access point on the equipment. All actions and interactions will be initiated, controlled and supervised from there. For existing applications, existing GUI scan remain visible for information visualisation purposes, and simply put the equipment in a remote-control mode. For new equipment, it should be considered to construct nonetheless an on-machine GUI for offline, engineering and service tasks whilst transmitting the main input during production via the API.

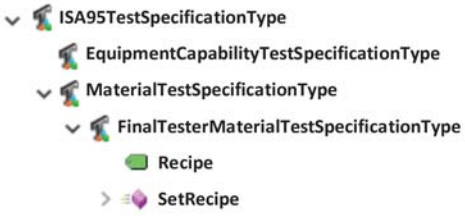
The output of existing applications is in most cases totally through the OPC UA interface. In this way, all the information are transferred to PAC and from there distributed to the e.g., database or MES. However, it may still is necessary to let the existing logging functions on an equipment run in the background for development and/or debugging purposes, to allow for quick interventions in exception cases. Security

related interactions like an emergency stop must always run on the machine and need to be able to also change the state through the API.

4.5 Implementation of the generic types and ISA95

The implementation of the generic types is done in the UaModeler. The first step, however, is to include the used companion specification to the modeler project, e.g., ISA95. After this step, new generic subtypes are implemented in a new namespace. This is illustrated in Figure 6 for the 'TestSpecificationType' which handles the recipe for the exemplary case of a final testing machine. The rather abstract ISA95 types are defined in a generic way for the actual implementation on the equipment.

Figure 6 Information model type definition example (see online version for colours)



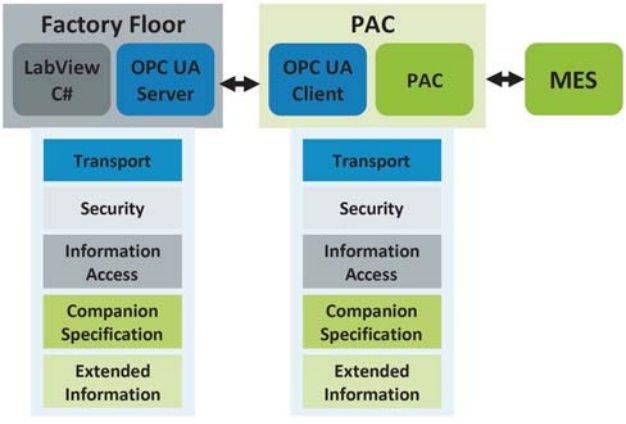
This generic definition of a OPC UA type and the actual representation of this in the server code fulfils the requirements of standardisation. The design process will follow the ISA95 specification down to the point where its design provides no further instructions on subsequent refinement. At this point, the extended information model in Figure 7 will define the needed generic type definition. These are the final types which are afterwards initialised in the dynamic part of the actual OPC UA server instance creation process. The definition of these types reflects the common equipment characteristics for the sum of all considered equipment.

4.6 OPC UA client and PAC software

The OPC UA client is on the other side of the communication and consists of two main parts as shown in Figure 7. The equipment from the factory floor as described in section 4.2 communicates to an OPC UA client built in PAC. PAC is an open platform for the vertical and horizontal implementation from shop floor equipment integration to the MES level. The platform is implemented in Java and built to be used in semiconductor and solar manufacturing industries. (znt Zentren für Neue Technologien GmbH, 2017)

PAC contains different external service adapters to communicate with external systems. One of these adapters is the SEMI equipment communications standard (SECS) and the Generic Equipment Model (GEM) to communicate with typical front-end equipment in the semiconductor industry. Another built-in service adapter is OPC UA. The adapter consists of the Prosys OPC UA Java Client. The client provides the basic communication and connectivity functions from OPC UA in the Java environment. These service adapters and PAC instances run in parallel as shown in Figure 3.

Figure 7 Overview of integration (see online version for colours)



The basic OPC UA communication properties are set directly in a configuration xml file. The address space and information model can be included in different ways. On one side, the kit provides a code generator. This is used to generate Java classes from OPC UA types defined in the information models (Prosys OPC Ltd, 2017). This generation is the equivalent step to the UaModeler for the development of the server development. The classes are then used to access elements from an actual instance of a server. On the other side, the address space information for instance is retrieved from browsing the actual address space of the server.

A characterisation process is used to get the actual PAC instance for an equipment specific OPC UA server. Like the SECS characterisation in PAC, the communication specific information is manually implemented for each instance. After the connection to the equipment is established, the characterisation process is executed. The process follows a Model-driven approach as described by Pauker et al. (2016).

4.7 Characterisation on the client side

The characterisation is a main feature implemented on the client side of the communication. It is part of the common PAC side and helps to create the actual PAC instance as shown in Figure 2. The characterisation browses the node tree from the main entry object and searches for objects, methods and variables. If a known structure is identified, an automated process takes the platform-independent model to derive the platform-specific implementation. This approach is very useful for larger address spaces with similar characteristics. These similarities are in this case the static part with the generic classes and the ISA95 specification. Another advantageous trait lies in the communication development process. In case of inconsistencies in the server of an actual equipment, the characterisation process acts like a compliance test tool and hands out an error to the developer, creating a fallback mechanism for the functional validation before transition to the fully operational state. Hence, this process yields an added possibility of

debugging the connectivity and increasing the quality of the software code within the integration process.

The main advantage of the model-driven approach is to reduce the effort of manual software development and the automatic adoption of changes every time the code is generated (Pauker et al., 2016). If the server is developed in accordance with the characterisation, the effort to reflect changes is brought down to an absolute minimum.

4.8 Proof of concept for final tester

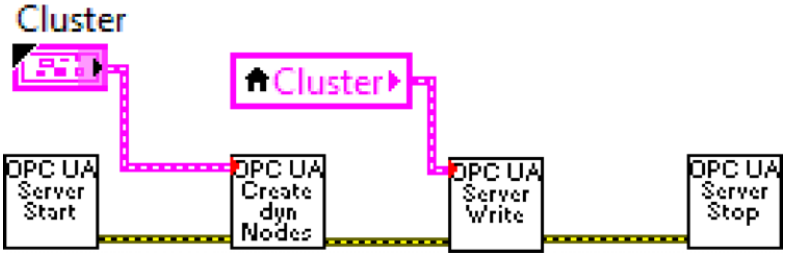
The developed approach to integrate equipment into smart factory applications has various use cases. The primary advantage of this procedure is realised for a manufacturing line with a high diversity of manufacturing assets and tools. The use case in the context of this paper is a proof of concept application for a final tester in a kW-class high-power laser diode production line. The final tester (Figure 8) is the last step in the back-end production line prior to shipping. The tool processes one laser module at a time, which can be of different type for each module, and gets the corresponding recipe information from the MES. The result of the test is stored as pass/fail criterion with the actual measurement values from the tool. The data consists of various electrical, thermal and optical parameters, including: power, efficiency, wavelength, and beam quality. A focus point of the overall measurement result is the traceability of the final product quality to the individual laser module. The data is additionally used for product and process development throughout the entire organisation.

Figure 8 Final Tester for kW-class high-power laser diode modules (see online version for colours)

Figure 9 Final tester example equipment specific information model (see online version for colours)

The ISA95 specification is implemented to handle 'Equipment', 'MaterialLot' and 'Personnel' information (Ono et al., 2013). For example, the 'MaterialTestSpecification' is used to handle recipe information and the according 'MaterialTestResult' to provide the data to PAC and the MES. Also, the 'EquipmentProperty' is used to handle main functionalities and states of the equipment through PAC and MES. These properties are also used to get general data about the health and condition of the equipment itself. This data builds the basis for the ERP, maintenance and product/process development. An example of this equipment specific information model is illustrated in Figure 9.

Figure 10 Sample call of OPC UA LabView SubVIs (see online version for colours)



The final tester itself is a one-of-a-kind unit developed for a specific diode laser module product. The source code of the equipment is based on LabView and running in a Windows environment. The integration of process data is done in LabView with the existing Clusters and equipment generic OPC UA SubVIs as shown in Figure 10. This dynamic integration of information in the address space is easy to handle for the individual equipment developer. The static OPC UA parts are implemented and predefined in the dll. Communication specific information like the IP addresses and security settings are defined in one xml file and should be set once at the beginning of the development.

5 Results

The integration of manufacturing equipment in a smart factory environment does not need to be cumbersome. The stated implementation shows different techniques and methods to reduce the variable effort for the integration of each equipment in an entire production plant, by the aid of the combination of abstract and domain agnostic companion specifications, their translation into domain specific applications with the extension of equipment specific information models in different programming languages. OPC UA fulfils all the needs for a basic communication technology. The implementation of this technology compared to SECS / GEM is more strategic in long term perspective and better supported in the short-term implementation. For example the SECS/GEM implementation on only one equipment can cost up to 50'000 USD. While the same LabView based equipment can be integrated with the mentioned method and implementation within some working week. If the integration also covers a whole line of manufacturing equipment, the initial effort of the method can also be shared over several

equipment and the effort per equipment even gets smaller. The open platform characteristics of OPC UA also enables the expansion and further development of other parts of the communication. This is, on the server side, the stated dynamic address space in terms of the use with another language like LabView. This possibility significantly leverages the effort for the implementation in whole manufacturing plants. The previously applied equipment-by-equipment integration is not reasonable in terms of the overall implementation cost and the long-term maintenance aspect. The implementation is even possible in more abstract programming languages like LabView and even easier in basic languages like C# .NET, which have been demonstrated and waged against one another.

On the client side also includes a number of distinguishing features. First is the client part of the whole MES side in general. Therefore, it is the first software layer where the incoming data from the equipment are processed and categorised. The output of the characterisation process during the development is the main feature on the client side. It works for a range of equipment servers and provides automated, reliable and sustainable output for further implementation. The result is favourable in terms of implementation time, costs and quality. The model driven approach also allows to abstract the different layers of information and already sort them for the further use. The effort in the initial setup of this characterisation is used for all the future equipment integrations.

The proof of concept provides a good reference integration and builds the knowledge foundation for the further implementation in a whole back-end production of a high-power laser diode factory. The main server and client features are already visible to bring all equipment to the state of the art MES level integration. The standardised components and automatic parts will help to apply the concept for over forty customised back-end machines. They consider also the maintenance and service aspects of future evolutions.

6 Conclusions and outlook

The field of equipment integration in terms of the digitalisation is as relevant as never before. MESs require valuable data input which are used for controlling product routing on the factory floor. Connectivity and communication are the fundamental requirements of these new needs in information technology (IT).

The validation of the interaction between a whole line of the back-end equipment and the MES needs to be done in the subsequent steps. The real-time process control within PAC and the Final Tester is validated in terms of a proof of concept. Also the gain of valuable data for subsequent engineering analysis needs to be validated but already has the right tools like a dynamic address space and LabView integration on board.

The long-term target is to have a reliable communication network and gain strategic advantages of it. This will be interoperability, traceability and finally the analysis of the collected data to improve products and processes.

A strategic integration of manufacturing equipment to smart factory applications like an MES is an organisational challenge. The interdisciplinary tasks combine knowledge from business logic, IT, process- and equipment-engineering, as well as adjacent departments. Reliable technologies combined with strategic implementations are necessary to make persistent and economically feasible integrations.

OPC UA provides an ideal framework for complex communication tasks and special needs. The combination of this protocol together with existing companion specification brings the necessary flexibility for one-of-a-kind manufacturing equipment and the required standardisation for these diverse units. This is especially needed in some back-end parts of the semiconductor industry.

The platform-independent machine-to-machine communication protocol together with established software development environments like C#/LabView results in short integration times. This also creates acceptance of the new technology within the existing equipment development process, even if the implementation causes important changes to existing applications.

The equipment integration is an important step towards a smart factory but should not be underestimated in terms of complexity and everyday impact. The framework presented here including the trade-offs between choice of source code language, ease of maintenance and integration, as well as reconciliation of information models for a complex manufacturing environment suggests an overall approach allowing to extract efficiency gains from smart factory applications in a straight-forward manner. Further research should be dedicated to the deployment of smart factory applications on the control layer, and a harmonisation of equipment or manufacturer specific information models which are not represented in industry-wide companion specifications today.

Acknowledgements

The authors would like to thank TRUMPF Photonics Inc. for the provided information and the financial support through the project.

References

Cemernek, D., Gursch, H. and Kern, R. (2017) 'Big data as a promoter of industry 4.0: lessons of the semiconductor industry', *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, Emden.

Chen, B. et al. (2018) 'Smart factory of industry 4.0: key technologies, application case, and challenges', *IEEE Access*, Vol. 6, pp.6505–6519.

Deuse, J., Weisner, K., Hengstebeck, A. and Busch, F. (2015) 'Gestaltung von Produktionssystemen im Kontext von Industrie 4.0', in *Zukunft der Arbeit in Industrie 4.0*, pp.99–109, Springer, Berlin.

Gittler, T., Gontarz, A., Weiss, L. and Wegener, K. (2018) 'A fundamental approach for data acquisition on machine tools as enabler for analytical Industrie 4.0 applications', *12th CIRP Conference on Intelligent Computation in Manufacturing Engineering*, Ischia.

Guo, N., Liu, Y-h. and Yan, M-h. (2017) 'Chip manufacturing, data integration and transmission', *Current Trends in Computer Science and Mechanical Automation*, Vol. 2, pp.41–51.

Hankel, M. (2015) *Industrie 4.0: Das Referenzarchitekturmodell Industrie 4.0 (RAMI 4.0)*, ZVEI, Frankfurt am Mein.

Harju, J. (2015) *Plant Information Models for OPC UA: Case Copper Refinery*, Tampere University of Technology, Nurmijärvi.

Kagermann, H. (2014) ‚Chancen von Industrie 4.0 nutzen', *Industrie 4.0 in Produktion, Automatisierung und Logistik*, Springer, Wiesbaden.

Kagermann, H., Wahlster, W. and Helbig, J. (2013) *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0: Abschlussbericht des Arbeitskreises Industrie 4.0*, acatech, Frankfurt am Main.

Kletti, J. (2015) *MES – Manufacturing Execution System*, Springer Verlag, Heidelberg.

Kolberg, D. and Zühlke, D. (2015) 'Lean automation enabled by Industry 4.0 technologies', *IFAC-PapersOnLine*, Vol. 28, No. 3, pp.1870–1875.

Kolberg, D., Knobloch, J. and Zühlke, D. (2017) ,Towards a lean automation interface for workstations', *International Journal of Production Research*, Vol. 55, No. 10, pp.2845–2856.

Monostori, L. et al. (2016) 'Manufacturing technology cyber-physical systems in manufacturing', *CIRP Annals – Manufacturing Technology*, Vol. 65, No. 2, pp.621–641.

Ono, T., Ali, S., Hunkar, P. and Brandl, D. (2013) *OPC Unified Architecture for ISA-95 Common Object Model*, OPC Foundation, Scottsdale (AZ).

OPC Foundation (2017) *OPC Unified Architecture Part 1: Overview and Concepts*, OPC Foundation, Scottsdale (AZ).

Palonen, O. (2010) *Object-Oriented Implementation of OPC UA Information Models in Java*, Aalto University School of Science and Technology, Espoo.

Pauker, F., Frühwirth, T., Kittl, B. and Kastner, W. (2016) ,A systematic approach to OPC UA information model design', *Proceedings of the 49th CIRP Conference on Manufacturing Systems*, Stuttgart.

Pethig, F. et al. (2017) *Industrie 4.0 Communication Guideline Based on OPC UA*, VDMA Verlag GmbH, Frankfurt am Main.

Prosys OPC Ltd (2017) *OPC UA Java SDK Code Generator Manual*, Prosys OPC Ltd, Espoo.

Radziwon, A., Bilberg, A., Bogers, M. and Madsen, E.S. (2014) ,The smart factory: Exploring adaptive and flexible manufacturing solutions', *Procedia Engineering*, Vol. 69, No. 69, pp.1184–1190.

Shariatzadeh, N., Lundholm, T., Lindberg, L. and Sivard, G. (2016) 'Integration of digital factory with smart factory based on internet of things', *26th CIRP Design Conference*, Stockholm.

Siemens Product Lifecycle Management Software Inc. (2019) *Camstar Semiconductor Suite* [online] https://www.plm.automation.siemens.com/global/en/products/manufacturing-operations-center/camstar-semiconductor-suite.html (accessed 12 February 2019).

Verein Deutscher Ingenieure e.V. (2016) *VDI 5600 Manufacturing Execution Systems (MES)*, Beuth Verlag GmbH, Berlin.

Virta, J., Seilonen, I., Tuomi, A. and Koskinen, K. (2010) 'SOA-based integration for batch process management with OPC UA and ISA-88/95', *IEEE 15th Conference on Emerging Technologies & Factory Automation*, Bilbao.

Wegener, K., Kunz, A., Bochmann, L. and Bänziger, T. (2016) 'Industrie 4.0 für den Maschinen- und Anlagenbau', in *Adaptive and Smart Manufacturing: Tagungsband*, Vol. 3, No. 1, pp.29–53.

znt Zentren für Neue Technologien GmbH (2017) *pac 14.x Documentation*, znt Zentren für Neue Technologien GmbH, Grünwald.