Domain Specific Assets: A software platform for use case driven human friendly factory interaction

Felix Brandt
Faculty of Informatics / Mathematics
University of Applied Sciences Dresden
Dresden, Germany
felix.brandt@htw-dresden.de

Eric Brandt
Faculty of Informatics / Mathematics
University of Applied Sciences Dresden
Dresden, Germany
eric.brandt@htw-dresden.de

Javad Ghofrani
Faculty of Informatics / Mathematics
University of Applied Sciences Dresden
Dresden, Germany
javad.ghofrani@gmail.com

David Heik
Faculty of Informatics / Mathematics
University of Applied Sciences Dresden
Dresden, Germany
david.heik@htw-dresden.de

Dirk Reichelt
Faculty of Informatics / Mathematics
University of Applied Sciences Dresden
Dresden, Germany
dirk.reichelt@htw-dresden.de

***Abstract*—In current efforts to digitize manufacturing and move it into the fourth stage of the industrial revolution, a wide range of integration solutions is being considered to enable manufacturing to adapt to change. In transforming a factory into a self-organized, autonomous factory, companies are currently struggling with rapidly changing requirements and production factors, among other things. This is a particular problem for the human being as an actor within the factory, as the amount of new technologies and protocols increases the training effort. Proprietary interfaces of the control providers, a wide range of different communication protocols, complicate the understanding of the production processes, the evaluation and testability of new use cases and increase the danger of creating silos of knowledge as well as building collaboration barriers. As a solution to these problems, we propose an open software platform and define a way to model use case driven domain specific asset representation (*DSA*) that focuses on the human being and his needs for representing the factory in a way that it meets his requirements for the current production needs. We therefore conducted research on google scholar on human factors in industry 4.0 and used technologies as well as already existing platforms and their architecture.**

***Index Terms*—cyber physical systems, industry 4.0, human machine interaction, sustainable production**

I. Introduction

Sustainable integration of resources through automation and digitization is still a challenge for companies. Recent research works discovered various aspects of digital transformation in social and technical levels [1]–[3]. The loss of employees with special knowledge is a particular problem for small and medium-sized enterprises (SMEs). This goes hand in hand with the high training effort required for new employees. [1]. Similar problems exist in the software development as well. Conventional software development in particular has encouraged the creation of silos that encapsulate relevant domain knowledge. However, collaboration and sustainable knowledge transfer are neglected in this methods. New approaches, which are summarized under the term *social innovation* [1], counter this problem by offering services, business models or technology that make a positive contribution to current social problems [3]. In this paper, We introduce our approach to overcome the problem of knowledge transfer in digitization of SMEs. Our approach is based on applying DevOps principles and agile process models, in combination with information models based on the communication standard OPC-UA [7]. DevOps describes the improvement of processes in software development and follows a strategy to close gaps between development and operations [9]. For the industry, theoretical frameworks or approaches already exist, but there is a lack of reference architectures that demonstrate the practical usability. Particular attention is given to the use of software platforms [14]. Correctly implemented, they would offer high network effects and increase understanding of the specialist domain. In this paper we will provide a testable platform, based on the CASO framework [11], which can be used for discussion and further research. CASO stands for *Capabilities-based and Self-Organizing Manufacturing Management*. The framework proposes the use of a domain-specific language to ensure the interaction of people with the factory and to enable them to create their own domain-specific models that meet their use case and increase their individual needs for the factory as well as their willingness to experiment and learn. Therefore we introduce the concept of domain specific assets (*DSA*) and show how OPC-UA can be used to provide a uniform basis for a standardized way of information processing. The goal was to create a platform that is as simple as possible, that does without constructs of high-level languages and that allows an easy entry and thus an easy familiarization while working with factories, their assets and the data flowing in them. The problem of insufficient consideration of individual needs is a sufficient and well researched problem in the

sponsored by German Federal Ministry of Education and Research within the funding program Forschung an Hochschulen. funding code: 13FH133PX8

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 April 2020 doi:10.20944/preprints202004.0167.v1

field of software engineering as well as industrial field of research. As basis for our platform we conducted research on google scholar and defined research questions to answer our motivated problem.

II. Introduction into DSA

A. CASO Framework

To demonstrate the use of domain specific assets (*DSA*) and the platforms usage we extend a software framework called CASO we created [11]. CASO serves as a framework for describing facilities and their capabilities in such a way that they can provide information about their manufacturing capabilities via standardized communication channels. This, according to the objective, is intended to provide the systems with a means of planning production steps independently and entering into negotiation processes with other machines as well providing functionality to describe assets of a factory. We define plant capability as a dynamic combination of process knowledge and plant characteristics, in which a plant can provide information on the extent to which it is capable of doing so, i.e. has the ability to manufacture a product. Various libraries are available for this purpose, which in principle implement these applications. (figure 1)

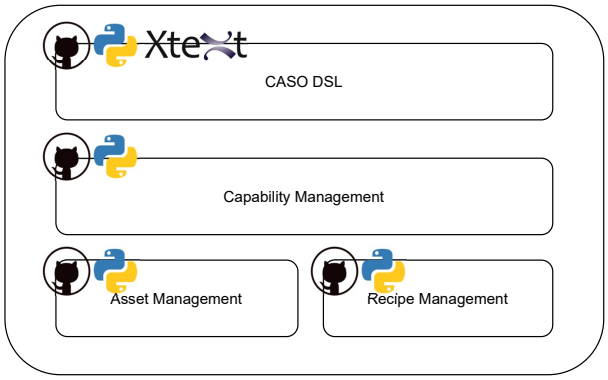

Fig. 1. CASO Components

The framework is based to a large extent on OPC-UA for communication and semantic description. Due to the wide range of use cases, no software stack was defined here, since different research results offer solutions in different languages. At the current state, a large part of the use cases was implemented with Python. CASO addresses the problem of human-machine interaction with a DSL(domain specific language) developed specifically for this purpose to simplify work with the framework. It proposes the use of different formats which are part of the DSL (figure 2)

III. CASO Platform Architecture

The creation of the CASO Platform initially concentrates on integrating all libraries of the CASO domain into services and making them available to the outside world via defined interfaces. These services are then combined in logical groups and

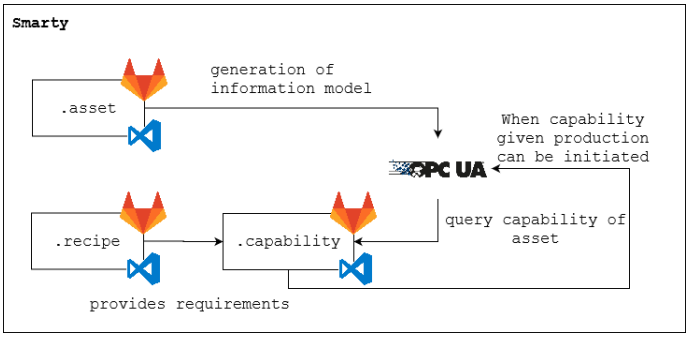

Fig. 2. CASO Components

form the individual domain services. CASO already provides the domains of asset management, capability management and recipe management but lacks of a user friendly way to communicate the users requirements to it. As of today CASO DSL focussed only on declarative asset creation and their connection to shop floor devices, but lacks the earlier mentioned problems of the user not being able to easily adapt the assets structure to their own needs. Therefore we define a light version of asset descriptions which fits a certain purpose rather than mirroring a field device or product as a whole.

A. Domain Specific Assets

As **domain specific asset** (*DSA*) we define entities that aggregates data from the factory to a degree that best suits the end-user's use case and gradually evolves to a full fledged administration shell along with monitoring capabilities of its state in an automatic manner. A *DSA* is charactarized by its short-lived nature as it changes over time as the requirements of the creator will. The structure is defined in form of an asset declaration and from this point on will be enriched with functionality till it reflects the users intention best. The job of the platform will be to provide an easy way for the end user to create and manage *DSAs* in a sophisticated and automated way. Most of the technology behind the framework should be abstracted away from the user. According to [20] and [21], there are a number of existing solutions that are used to develop and operate platforms. Kubernetes is currently the most important of these in terms of practical application. [19] Kubernetes is a tool for orchestration of container applications and therefore we restructured the services, CASO offers, into fitting artefacts that meet the requirements of kubernetes. For ease of use we provide a command line interface (cli). This cli can be used to create a *DSA* project-folder.

B. CASO Platform Architecture

Since the platform will be based on kubernetes, a series of services must be created, which are then containerised. For the platform these services are a *service api*, which is the interface for the already mentioned cli. A *deploy agent* acts as a provisioner for *DSAs*, while a *DSA* lives as its own deployment within kubernetes. A deployment is a declarative description of a desired state of an application and will take

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 April 2020 doi:10.20944/preprints202004.0167.v1

care of the *DSAs* life cycle. The deployment agent also takes care of deploying monitoring resources for the given *DSA*. As monitoring solution we use grafana, prometheus and several database solutions like influxdb (figure 3).

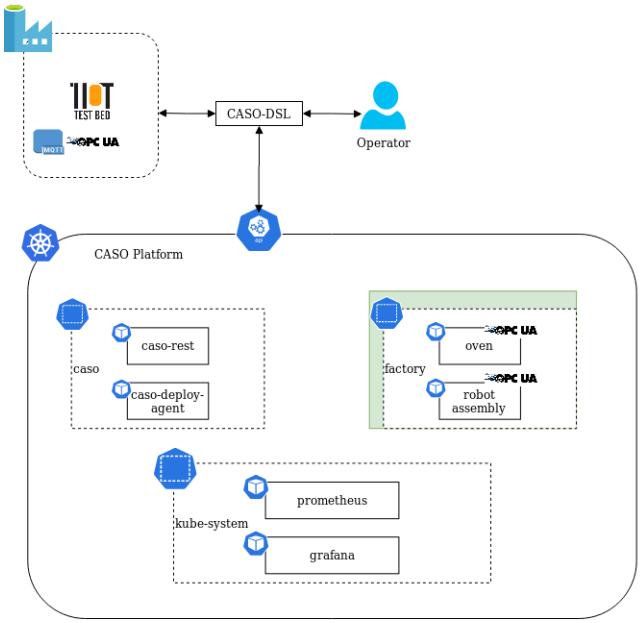

Fig. 3. CASO Infrastructure

C. Deployment Process of DSAs

A typical project structure for a *DSA* generated by the cli looks similar to the structure from listing 1.

```
dsa/
    + assets/
      - robotassembly.asset
```

Listing 1. CASO DSA Project

Within the platform, the deployment process of the *DSA* is now designed in a way that the cli is connected to an interface to the CASO platform. This interface processes the request and receives the project *DSA* from listing 1 as input. The Service API of the CASO Platform is capable of making requests to services, which will deploy a *DSA* and a monitoring structure. If available, OPC-UA information models are created from asset files within the asset generator and a server is created which the user can access. At the same time the Service API sends a deployment request to the deployment agent. The deployment agent will then queue a request to create a deployment for a *DSA*. The moment the *DSA* is created it will start listening for events and receive the previous request to start a server based on the asset description. For the project from listing 1 an asset is created and an OPC-UA server is started. Additionally, a mapping of the endpoints is required. For this purpose a mapper file is created, which takes over the generation of the mapper for different protocols. A mapper file is declared inside the dsl and caso will take care of generating the appropiate source code. The whole process is shown in figure 4.

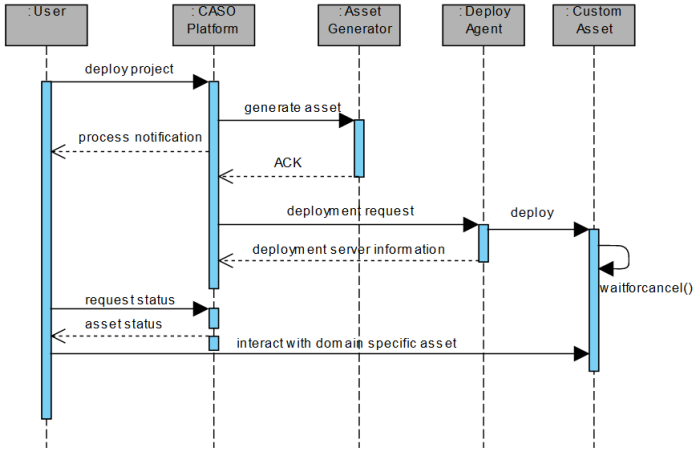

Fig. 4. DSA Deployment Process

IV. Demonstration of the Platform

The platform can be rolled out on any Kubernetes cluster. The repository [18] contains all necessary components. The only requirement is a configured kubectl cli and grafana tanka [19].

A. Use Case - Production Tracking

The first device chosen was a single standing station of the factory with two rfid sensors (figure 5). At this station, products are transported on a conveyor belt to a station where manual work is carried out. The station is equipped with a Siemens controller, which provides an OPC-UA server to the outside.

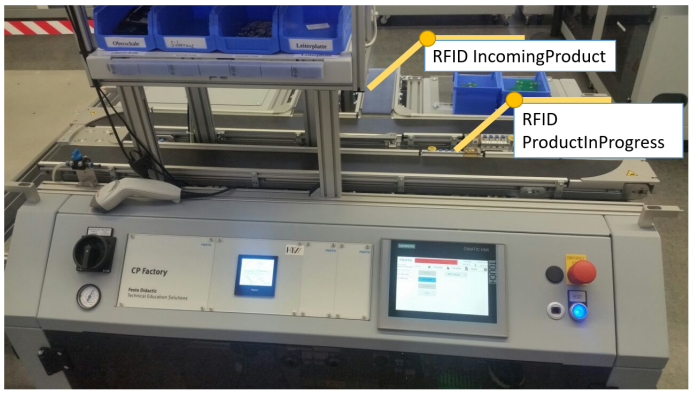

Fig. 5. Station One - Handworkplace

The OPC-UA model does not follow any specification and the endpoints are not named in a standardized way. By looking at the documentation, however, we get information that at two points during the process, information regarding the product is stored. The Use Case to be implemented is the monitoring of all products currently being processed at the manual station and which is the next product to be processed. For this we do not need all the endpoints provided by the controller and we

want to change the representation of the factory. As described both values will be received from the endpoints shown in figure 5. As soon as one product enters the production line, the values will be captured and sent to the internal opcua server running on the machine. To build our *DSA* we define the structure of it by creating a .asset file (Listing 2) within the projects asset folder as well as a mapping to connect the field devices to the *DSA*.

```
asset HandWorkplace
asset ProductIncoming {
    endpoints[ordernumber,orderposition]
}
asset ProductInProgress {
    endpoints[ordernumber,orderposition]
}
```

Listing 2. Workplace Asset Description

ProductIncoming represents the incoming product in the future and we are only interested in the order number and the position. ProductInProgress is our product, which is currently being processed. The function append builds the hierachy and puts the subassets under the specified asset. If we now deploy the *DSA* with the command **caso deploy** to the platform which will provide us with an OPC-UA Server, which structurally corresponds to our use case. The structure from listing. 2 is reflected in the address space of the OPC-UA Server.

V. Conclusion

The platform proposed in this paper currently provides a service API that enables end users to communicate with assets in a factory in a way that the user can reshape dataflows to a self defined representation without the need for special technology knowledge. We therefore introduces the term of domain specific assets. This communication is right now limited by the generated OPC-UA model and the expressiveness of the CASO DSL. However, the end user only needs to know how assets can be constructed and connected to existing infrastructure. From there it is up to the platform and the users project to gradually enrich it till the representation of the asset fits the use case of the user. Further research now focused on conducting empirical tests to validate the proposed platform. If this knowledge is available, assets can be created relatively easily for one's own use case. For this purpose, no source code has to be implemented apart from the DSL. This fulfills the need of fast experimenting and learning and lowers collaboration barriers for both developers and operators at factory level. Providing the OPC-UA server has the advantage that one of the freely available clients can be used to visualize and navigate the *DSA*. The declarative description of the use case can serve as a basis to introduce new end users to the structure and data flows of the factory. Through the consistent use of DevOps principles, the maintenance of the *DSAs* can be largely automated and facilitate faster validation of new components in the system. Various points are considered to be disadvantages that require further investigation. For example, it was found that the mapping of endpoints that do not return standard types cannot be serialized without a corresponding change to the template from which the OPC UA server is generated. If the server does not provide the functionality to create custom structs or load them from the source server, this has to be done manually. The Asset DSL offers rooms for extensions to enrich the language and make it more expressive. Also still showed weaknesses within the mapping component. Depending on the protocol, this currently requires the concrete specification of the endpoint, which in turn requires special domain knowledge, such as the concept of NodeIds in OPCUA endpoints or topics in MQTT endpoints that we don't want to have at that place because it collides with most of the devops principles and builds more barries than it removes.

References

[1] R. Morrar, H. Arman, S. Mousa, "The Fourth Industrial Revolution (Industry 4.0): A Social Innovation Perspective" Technol. Innov. Manag. Rev., vol. 7, no. 11, pp. 5–11, 2017.

[2] K. Zhou, T. Liu, and L. Zhou, "Industry 4.0: Towards future industrial opportunities and challenges," 2015 12th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2015, pp. 2147–2152, 2016.

[3] Kinzel, H. (2017). Where Does This Leave the Human Factor ? Journal of Urban Culture Research, 15, 70–83.

[4] Benešová, A., & Tupa, J. (2017). Requirements for Education and Qualification of People in Industry 4.0. Procedia Manufacturing, 11(June), 2195–2202. https://doi.org/10.1016/j.promfg.2017.07.366

[5] Jernæs S, A sland JE, Heber H, Morvik R, Leistad G, Enoksen AM,Ellingsen A (2005) Human factors in drill and well operations:challenges, projects, and activities

[6] Stanton NA, Salmon PM, Walker GH, Baber C, Jenkins DP (2013)Human factors methods. A practical guide for engineering anddesign, 2nd edn. Ashgate, Hampshire

[7] https://opcfoundation.org/about/opc-technologies/opc-ua/

[8] F. M. A. Erich, C. Amrit, and M. Daneva, "A qualitative study of DevOps usage in practice," J. Softw. Evol. Process, vol. 29, no. 6, pp. 1–20, 2017.

[9] R. Jabbari, N. Bin Ali, K. Petersen, and B. Tanveer, "What is DevOps? A systematic mapping study on definitions and practices," ACM Int. Conf. Proceeding Ser., vol. 24-May-2016, 2016.

[10] Dejanović I., Vaderna R., Milosavljević G., Vuković Ž. (2017). TextX: A Python tool for Domain-Specific Languages implementation. Knowledge-Based Systems, 115, 1-4.

[11] Brandt E., Brandt F. (2020). DSL-gestützte dynamische Generierung von Informationsmodellen. 17. Fachkonferenz für Automatisierung und Mensch-Technik-Interaktion, 2020-March, p. 203-212

[12] Evans. 2003. Domain-Driven Design: Tacking Complexity In the Heart of Software. Addison-Wesley Longman Publishing Co., Inc., USA.

[13] Hasselbring, W., Henning, S., Latte, B., Mobius, A., Richter, T., Schalk, S., & Wojcieszak, M. (2019). Industrial DevOps. Proceedings - 2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019, 123–126. https://doi.org/10.1109/ICSA-C.2019.00029

[14] Evans, D., Schmalensee, R. and Hagiu, A. (2014). Invisible Engines: How Software Platforms Drive Innovation

[15] Jha, P., & Khan, R. (2018). A Review Paper on DevOps: Beginning and More To Know. International Journal of Computer Applications, 180(48), 16–20. https://doi.org/10.5120/ijca2018917253

[16] Christof E., Gorka Gallardo (2016) DevOps. Published by IEEE Computer Society

[17] Hasselbring, W., Henning, S., Latte, B., Mobius, A., Richter, T., Schalk, S., & Wojcieszak, M. (2019). Industrial DevOps. Proceedings - 2019 IEEE International Conference on Software Architecture - Companion, ICSA-C 2019, 123–126. https://doi.org/10.1109/ICSA-C.2019.00029

[18] Brandt E., Brandt F., Heik D. https://gitlab.com/smartproductionsystems/projects/caso/caso-infra

[19] Grafana Tanka (2020), retrieved from: https://github.com/grafana/tanka