5th Conference on Production Systems and Logistics

Proposing A Solution For A Self-Managed Data-Ecosystem In Production: Use-Case-Driven IT-OT-Integration With An Event-Driven IT-Architecture

Sebastian Kremer[1], Christian Mennerich[2], Max-Ferdinand Stroh[1], Günther Schuh[1]
[1]FIR e.V. at RWTH Aachen University, Aachen, Germany
[2]synyx GmbH & Co. KG, Karlsruhe, Germany

Abstract

With the development of publicly accessible broker systems within the last decade, the complexity of data-driven ecosystems is expected to become manageable for self-managed digitalisation. Having identified event-driven IT-architectures as a suitable solution for the architectural requirements of Industry 4.0, the producing industry is now offered a relevant alternative to prominent third-party ecosystems. Although the technical components are readily available, the realisation of an event-driven IT-architecture in production is often hindered by a lack of reference projects, and hence uncertainty about its success and risks. The research institute FIR and IT-expert synyx are thus developing an event-driven IT-architecture in the Center Smart Logistics' producing factory, which is designed to be a multi-agent testbed for members of the cluster. With the experience gained in industrial projects, a target IT-architecture was conceptualised that proposes a solution for a self-managed data-ecosystem based on open-source technologies. With the iterative integration of factory-relevant Industry 4.0 use cases, the target is continuously realised and validated. The paper presents the developed solution for a self-managed event-driven IT-architecture and presents the implications of the decisions made. Furthermore, the progress of two use cases, namely an IT-OT-integration and a smart product demonstrator for the research project BlueSAM, are presented to highlight the iterative technical implementability and merits, enabled by the architecture.

Keywords

Event-Driven IT-Architecture; IT-OT-Integration; Data-Ecosystem; Self-Managed; Industry 4.0

1. Introduction

Since its conception in 2011, the term Industry 4.0 defines the digital revolution of industry and proposes a target vision for digitalisation activities for production environments [1]. While developments of the last decade have achieved the realisation and validation of Industry 4.0 use cases in both research demonstrators and integrations into commercial system solutions [2], the overall state of digitalisation in the industry has, in experience, remained poor. While the realisation of a true data-driven ecosystem, as proposed in Industry 4.0, poses a challenge, a previous investigation proposes the perceived problems to derive from an IT-architectural point of view and has identified event-driven IT-architectures to be a suitable fit for the application in and realisation of Industry 4.0 [3].

The adaption of event-driven architectures (EDA) as an overall IT-architecture for production addresses the need for a multi-agent, real-time, data-driven ecosystem while retaining a manageable complexity due to modularity, decentralisation, and decoupling [4,5,3]. While businesses from a broad range of branches have already identified event-driven IT-architectures as an efficient tool to self-manage business operations via

DOI: https://doi.org/10.15488/15268
ISSN: 2701-6277

digital services in a distributed way [6,7], the adaption in industry is limited to a few prominent examples, mainly from the automotive industry [8–10]. In experience, the industry's interest in such an architecture in production is quite high, yet only few decided to undertake such a transformation. The is mainly due to the uncertainty of success in the face of high initial effort; the proposed long-term benefits are undermined by the short-term expenditures. Furthermore, the philosophy of self-managing IT-architecture development and digitalisation still poses a barrier despite the existence of publicly available, powerful software solutions.

To solve this stalemate, the decision was made to undertake such an endeavour within the research environment of the Cluster Smart Logistics in Aachen. The research institute FIR partnered with the IT-expert synyx to transform the Demonstration Factory Aachen (DFA) into an event-driven smart factory. The factory is subject to various research entities and partners acting and developing in the production environment, which induces heterogenous IT-solutions with secluded data spaces and uncertain organisational responsibilities, resulting in an IT-landscape comparable to commercial productions. The goal of the transformation lies in the creation of a unified data space for enhanced data availability and for the DFA to regain control and independence in the management of its complex IT-landscape. To address the DFA's IT-strategy in creating a self-managed data-ecosystem, a target vision was conceptualised, and its iterative realisation addressed along use cases, generating a tangible benefit.

2. State of the art and design principles

The conceptualisation and realisation of a self-managed, event-driven IT-architecture intends to establish sustainable digital sovereignty. The aspiration for the executing parties is to reach this goal both efficiently and effectively, which is pursued by value-creating development increments in an agile manner and the utilisation of a CI/CD process for streamlining deployments. The following chapters present the principles of event-driven architectures, digital sovereignty, agile software development and the CI/CD process.

2.1 Event Driven Architectures

Event-driven architecture (EDA) describes an architecture in which components communicate asynchronously through the exchange of events via a middleware (= broker). Such events can be described as a data package communicating, for example, a state change of a component or the attainment of information by a service. Events consist of a payload carrying event-relevant data that is contextualised by specifying an action that this data has occurred in (e.g., an end of operation or a measurement). The transmission of events is conducted by a component publishing/producing an event in a specific topic (analogous to a directory) of the broker and recipients actively subscribing/consuming events by topic from the broker. The overall design principle of distributed systems entails eventual consistency (possibility of data not being known to all systems at a single point in time) and possible loss of events that is to be dealt with accordingly in application design. [11–14]

Because this architecture enables the interconnection of distributed functionality and various systemic domains, event-driven architectures have historically evolved from a central design principle for operating system to a backbone for complex software solutions [11]. With the availability of industrially proven open-source broker software, like Apache Kafka, this design principle can now be extrapolated to the scale of corporate data-driven IT-architectures, to interconnect business domains in a manageable decoupled manner.

2.2 Digital Sovereignty

Digital sovereignty describes the extensive control of a company over its own data. This logically extrapolates towards control over the software and hardware that process such data, and ultimately implies complete control of their value chain. This provides valuable insights into mission critical business processes or can reveal indicators on how to execute them more reliably or efficiently. Digital sovereignty thus can

support making the right business decisions towards business evolution, growth, or expansion into new markets. [15]

To achieve digital sovereignty, a transformation, or digital evolution, of a company is necessary. Digital processes are not a simple replacement for existing ones, but become the essence of the value chain, and thus move in the centre of the organisation [16,17]. Thus, Conway's Law [18] applies: The organisational structure of a company is necessarily reflected in any digital design. This relation is isomorphic, and thus the companies' organisational structure and digital value chain are interdependently related [19]. Adoption of modern software development methodologies are the means to facilitate this evolution. This includes agile methodologies borrowed from e.g. DevOps to streamline teams around a common vision. EDA as an architecture style can help uncover communication flows in the company, between departments or parts of a facility. These can then be analysed with modern sociological approaches e.g. Luhmann's system theory [20], and the complex interplay of requirements engineering and recent trends in modern software development be combined and tackled.

2.3 Agility and agile software development

Agile software development has become the state-of-the-art methodology of modern software development. Since the publication of the agile manifesto in 2001 [21], methodologies like SCRUM, Kanban or Extreme Programming have become the de-facto standard in professional software development. Agile focusses on customer satisfaction with working software as the highest goal and accepts that the world is permanently unpredictably changing requirements. This is tackled by continuously learning, and results in development employing frequent increments, inspect and adapt patterns, fail fast philosophy, and short feedback cycles. Techniques from domain-driven design (DDD) [22] experience increased awareness as integration into agile software development processes (elaborated in appendix). These techniques are event-centred and focus on communication flows, thus comply with EDA naturally.

Ultimately, agile is a characteristic of an organisation's vision and mindset, not it's IT department. Agile organisations are more viable in free markets by the ability to adopt to everchanging environments and regulations [16]. In experience, agility is not a buyable off-the-shelf tool for incorporating or outsourcing software development but needs cross-functional software teams to be embedded in agile organisations (or projects). Often, this means to introduce a new mindset in an organisation and comes along with the complexity of change management and associated challenges. If not addressed properly, the boundaries between the agile (software) development teams and rigid organisation structures become bottlenecks for communication flows and prevent efficient decision-making processes, as stated, again, by Conway's law.

2.4 CI/CD – Continuous integration and development/deployment

In Experience, modern software systems are subject to frequent changes that need to be releasable into production on a regular basis on short time scales, in case of mission-critical bugs within hours or even minutes. To have control and knowledge about the components that run in production, proper versioning of every software service is inevitable. Versioning systems like git are widely used in combination with integrated platforms like GitLab or GitHub that provide DevOps methodologies. These systems offer mechanisms for automating the release process of software into test, staging and production environments and, at the same time, ensure compliance with defined quality standards.

Preferably, this is an automated process that deploys the new (or any other stable wanted) version into production environments: continuous integration and development/deployment (CI/CD). At best, every successfully implemented feature (i.e., feature that passes the quality gate) triggers an automated deployment. That way, software development teams can release features into production several times per day. High deployment cycles (at least to test and staging environments) and short lead time for changes are valuable, as they enforce high quality software and keep the teams trained, thus increasing confidence and

reducing cognitive load [17]. Automated CI/CD pipelines in general serve several purposes which are further discussed in the appendix.

3. Proposed solution

The implementation of an event-driven IT-architecture in a production environment is not to be understood as a general solution to Industry 4.0 use cases. The selection of an IT-architecture is a strategic decision that must match the requirements and prospects of the concerned company [3]. The applicability of EDA for the Demonstration Factory Aachen (DFA) was identified as a solution to address the challenges specific to the situation of multiple stakeholders acting in one factory (refer details in appendix). With the decision of EDA to be the central design principle for the DFA, the IT-architecture was then specified along the by experience relevant architecture dimensions *technology*, *organisation*, *data,* and *applications*.

3.1 Technology Dimension

The foundation of an event-driven architecture is a central entity accepting and making contextualised data packages available [11]. In the scope of an IT-architecture, such a central entity is often implemented using a message broker to integrate all components within the network. From experience gathered in industry projects, the realms of IT (e.g., IT-Systems, Applications) and OT (e.g., PLCs, machines, sensors) have different demands towards the central connectivity: IT-systems conventionally expect data persistence as in query-able master records, OT puts more demands on real-time availability. Consequentially, the allocation of two dedicated interconnected brokers is proposed, providing technologies that complement and serve IT and OT separately. The developed technical IT-architectural is visualised in Figure 1.

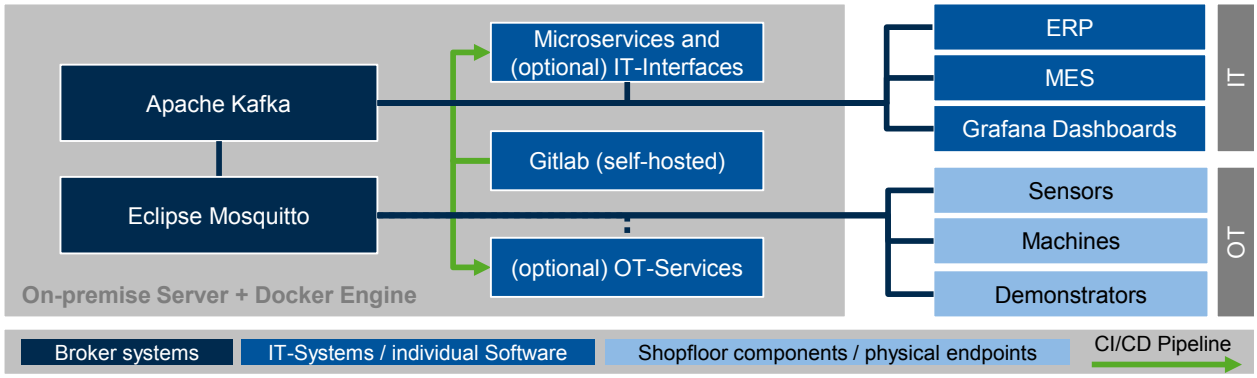


Figure 1: IT-Architectural overview of the EDA concept for the Demonstration Factory Aachen

Regarding limited resources in the transformation process and retaining an overall independence in the DFA's research environment, the choice of broker systems focuses on publicly available open-source solutions. The chosen technologies are Apache Kafka and Eclipse Mosquitto. Kafka offers valuable scalability and persistence capabilities for IT on top of the TCP based Kafka messaging protocol. Mosquitto is based on the IoT standard protocol MQTT and provides real-time and lightweight interconnectivity for OT. Even though the industry standard OPC-UA does offer plug-ins to support publish/subscribe patterns, the decision was made to focus on MQTT, due to its reduced complexity, broader support for IoT-sensors and the enhanced flexibility to individually structure event on different abstraction levels (for e.g. custom services) [23]. Ultimately, OPC-UA natively employs 1:1 data-querying, which in the context of DFA both risks maintainability and does not inherently fulfil the contextualisation of data in events. Prospectively, a more production-robust OT-solution will be evaluated with commercial products that offer convenience functionality for integration (including OPC-UA endpoints). IT- and OT-endpoints that do not natively connect to the brokers, will be integrated via a dedicated service and eventually a technical solution for physical endpoints. The broker software is planned to be deployed on-premise on dedicated servers within

the network of the DFA, accessible for all relevant resources. The deployment of the broker software and subsequently developed services that interface network-components or implement specific logic, is handled via CI/CD, using a locally maintained Gitlab instance and a docker container runtime.

3.2 Organisation Dimension

From an organisational point of view, the main challenge lies in combining various demands regarding functionality and available datapoints into a single consistent model and governing adherence, which is required for the functioning of distributed systems [12]. To align demands in the central data structures and corresponding responsibilities, the architectural orchestration is proposed as a change management process. Any request for change regarding data access and the centralised data model is funnelled through designated architects, who align them with the architectural vision and derive specific measures. These measures are communicated to the corresponding owners that are then responsible for implementation. From a technical perspective, each endpoint is required to manage its own connector to the broker, if not already defined within the standard stack of interfaces provided by central organisation. Adherence to correct usage of devised models will be governed with access control and schema validation. The management of services, interfacing endpoints and implementing specific logic components is handled via CI/CD that serve as a quality gate for the software and ultimately install deployable artifacts into production environments.

3.3 Data Dimension

Managing distributed systems in a centralised data space requires the definition of a standardised data model, to be adhered to by contributing components [12]. The overlying philosophy is defined to be “finding data instead of searching for data”. This philosophy is expected to harmonise with the nature of EDA to contextualise data by specifying the context in a topic that the event is published within. The data dimension of the architecture focuses on designing a coherent structure for the topics that allows a user to find data by matching its context against the literal topic. Furthermore, the data dimension specifies top level segments designated for production, development and third parties, to structurally decouple these domains and allow for an effective management of authorisation.

Hence, the top level of the topic tree defines the domains *dfa* (DFA, as the main productive domain), *dev* (development) and *ext* (external) – the domain *fir* was later added for production unrelated research projects by FIR. The *dev* and *ext* branches define subtopics named after use cases and external company name, respectively. The *dfa* topic partitions subtopics into both specific and abstract business objects. For in depth reference, Table 1 presents an overview of the topic structure envisioned for the EDA.

The topics define a tightly scoped context that allows for specification of strict responsibilities. The base topic of each business object contains general events regarding the resource; further segmentation (e.g., identifiers for individual orders) and specific data points (e.g., sensor data of a machine) are placed in individual subtopics. For potential direct interaction with a domain, a “request” subtopic is defined for topic necessitating such a functionality. Data schemas transported via the topics are still in development, as they will be defined according to the requirements of implemented use cases and hence are omitted in this paper.

Table 1: Topic structure within the broker based on business objects

Business Object	Topic structure	Potential Owner [1]	Description
OT-Device	dfa.ot.<id>	Respective device	Status updates of shopfloor machines, sensors, or robots
IT-System	dfa.it.<id>	Respective system	Proprietary (not adhering defined model), but relevant data produced by IT-systems [2]
Microservice (Business Functionality)	dfa.svc.<id>	Respective service	Results and heartbeats of microservices managing logic, data transformation, and analysis
Order	dfa.order	e.g. ERP	Updates on production orders
Operation	dfa.operation	e.g. MES	Updates on shopfloor operations
Transport Order	dfa.transport	e.g. EWM, FTS, MES	Updates on intralogistics orders
Warehouse	dfa.warehouse	e.g. EWM	Warehouse inventory
Product	dfa.product	e.g. PLM, ERP	Assembly status of parts / product
Master Record	dfa.rec.<record>	e.g. ERP, MES, EWM, CRM	Persistence for e.g. material specifications, packaging lists, worker info or customers
Development Domain	dev.<use case>	Respective use case / service	Data space for use case specific developments by research entities
External Domain	ext.< name>	Respective partner	Data space for partners for integration and testing of individual IT and OT
FIR Services	fir.svc.<id>	*Analogous to dfa.svc.<id>*	
FIR Devices	fir.dev.<id>	*Analogous to dfa.ot.<id>*	

3.4 Application Dimension

The application dimension in the IT-architectural design focuses on the definition of standard applications and their integration into the IT-landscape. While the DFA has specific IT-systems defined for required functionality (e.g., an ERP), EDA does not require for a specific system to be set as a standard application. In fact, the architectural design allows for the expected heterogenous IT-landscape to incorporate multiple IT-Systems reading and writing data from and to the centralised broker.

Nevertheless, the specific role of existing IT-Systems and general applications within the DFA's IT-landscape need to be defined, to manage responsibilities regarding provided data. By applying a contextualised data space as proposed in 3.3, the distribution of responsibilities regarding these data spaces can be determined and retained. Data-producing applications are bound to their respective service or business object in accordance with Table 1 in appendix. Prospectively, standard applications will be defined for developing dashboards, interfaces, and low-code / no-code logic with the aim to accelerate the utilisation of the architecture by new stakeholders.

4. Implementation and validation of proposed solution

Having proposed a concept for the IT-architecture's dimensions, the goal was set to pursue its implementation. Like experienced in the industry, the risk of implementing an unverified concept with a lack of ample resources hinders the extensive implementation at once. Hence, the advancement was decided to derive from actual realisable increments with a tangible benefit that iteratively pay into the overall vision of

[1] Likely candidates. Any designated service can own a topic to abstract interfacing one or even multiple IT-systems.
[2] Business-object-relevant data is either directly published to the domain-topic or handled via a topic-designated service.

the IT-architecture. This iterative approach suits the design of an EDA [3] and allows to consecutively add new requirements and adapt the choices made during the design phase in an agile manner.

Derived from user stories in agile development, the basis for these realisable, beneficial increments are use cases within the environment of the Cluster Smart Logistics. Relevant use cases for and with a self-managed data-ecosystem were identified in discussion with stakeholders like the DFA and research entities, and in regard of demonstrators for active research projects. Subsequently, a roadmap was designed by breaking down use cases with an adequate benefit to effort ratio into sprints with a priority on creating a minimum viable product first. The two use cases implemented by the time of this paper are a case of machine data acquisition for transparency and a demonstrator for smart products further discussed in 4.2 and 4.3 respectively. Section 4.1 first presents an overview on the current state of implementation.

4.1 State of Implementation

The DFA's current EDA solution is deployed on dedicated on-site Linux servers. A single Mosquitto Broker (MQTT) handles both IT and OT to rationalise complexity in the early stage of development. A self-hosted GitLab instance is used for software version control and serves as the environment for CI/CD pipelines. Microservices, mainly written in Rust, interface the broker and enable the use cases functionality by gathering data from endpoints like machines and sensors, feeding data to endpoints like databases and dashboard, abstracting endpoints into digital interfaces and executing logic.

These services are developed and maintained in a GitLab project that is CI/CD enabled and defines pipelines that trigger project build, test, and image creation. For now, a manual containerisation and deployment step is necessary to start the service into the production environment with Docker. At a later stage of the transformation, a container orchestration tool like Docker swarm or Kubernetes will be used to maintain the overall operations infrastructure, and pipelines will be extended to directly deploy into production environments. Such container orchestration platforms also offer means to comply with requirements to the availability of services, via health checks and monitoring, including automatic restarts and dynamic scaling.

4.2 Validation: Machine data acquisition – Tracking inert gas usage for laser cutting process

The first developed use case developed into the EDA addresses an uncertainty that the DFA faces regarding the consumption of inert gas for a laser cutting process. While the material processing is mainly unconcerned from the eventual exchange of gas containers during the process, personnel had no way to estimate the remaining useful levels for short term planning or estimate required volumes for specific orders. This impaired the scheduling of resupplies, predicting the order-specific cost and fine planning operations. The target was hence to achieve transparency in the inert gas usage and predictability regarding orders.

The realisation of the use cases first focused on attaining real-time process data (unbuffered, 1 Hz – 10Hz) from the laser cutting machine. This was achieved by a stack of flow sensors and relay-states feeding into an IoT-Box[3], a component attached to the machine and establishing connectivity. The data was then pulled into the centralised broker via a dedicated microservice, continuously fetching the data from the IoT-Box and posting it into the machine specific topic *dfa.ot.trumpflaser* and respective sensor subtopics. Moreover, the service enriched the forwarded data by determining the operation state of the machine into a predefined, factory-wide data format. To achieve the first benefit the data was then written into a time-series database via a dedicated second microservice, to allow the implementation of a simple Grafana dashboard for the end users. Thus, workers were able to match the usage of gas to the periods of corresponding operations and ultimately the consumption of inert gas per type of operation. To automate the matching of process to order data, a microservice similar to the connection of the OT device will then interface the ERP's API and post current order data into the topic *dfa.order* with subtopics specifying the order id. This data will both serve

[3] Based on an HPE Edgeline EL300 with OT Link

to generate a workers' dashboard of active order (that is required in another use case) and feed into a microservice mapping the integrated consumptions to produce estimates per known type of order.

The realisation of this use case allows to validate certain assumptions made with the pursuit of an EDA in a production environment. The integration of the laser cutting machine via an IoT-Box showed that the architecture is robust against various types of endpoints in context of specific protocols and formats or it being a server or a client. By the utilisation of a dedicated microservice the integration did not need to rely on the machine proactively posting its data but could remain a passive endpoint. Furthermore, the architecture provided excellent availability and usability of the machine data, without the necessity of gateway systems like MES or third-party platforms. At last, the scope of the deployed microservices were precise, which allowed for a clear identification of responsibility and introduction of changes along the development process. Yet, as stated in 2.2, the implementation required direct confrontation with technical challenges and hence expertise in the technical solution design and development of software-components.

4.3 Validation: Demonstrator for research project BlueSAM – Realisation of a smart product

The second use case substantiated itself in the planned demonstrator for the publicly funded research project BlueSAM[4]. The research project aimed to construct **Blue**prints for **S**mart product **A**rchitecture **M**anagement that derive relevant IT-architectural components for the pursuit of individual use cases with the application of a smart product. In the project a method was developed and build into a publicly accessible webtool[5] which helps users identify relevant smart product use cases and learn among others about functional requirements a respective IT-architecture must fulfil. As part of the research project, the method was tested with a digitalised espresso machine (specified as DE1) as an exemplary smart product to (analogously) demonstrate digital use cases on handcraft with industrial machines. The espresso machine already offered Bluetooth capabilities and a local interface for operation but needed to be upgraded to realise the envisioned smart product use cases of assisting the user in the operation of the machine, offering data analytic capabilities to the user, and upgrading the provided digital services on data-based learnings.

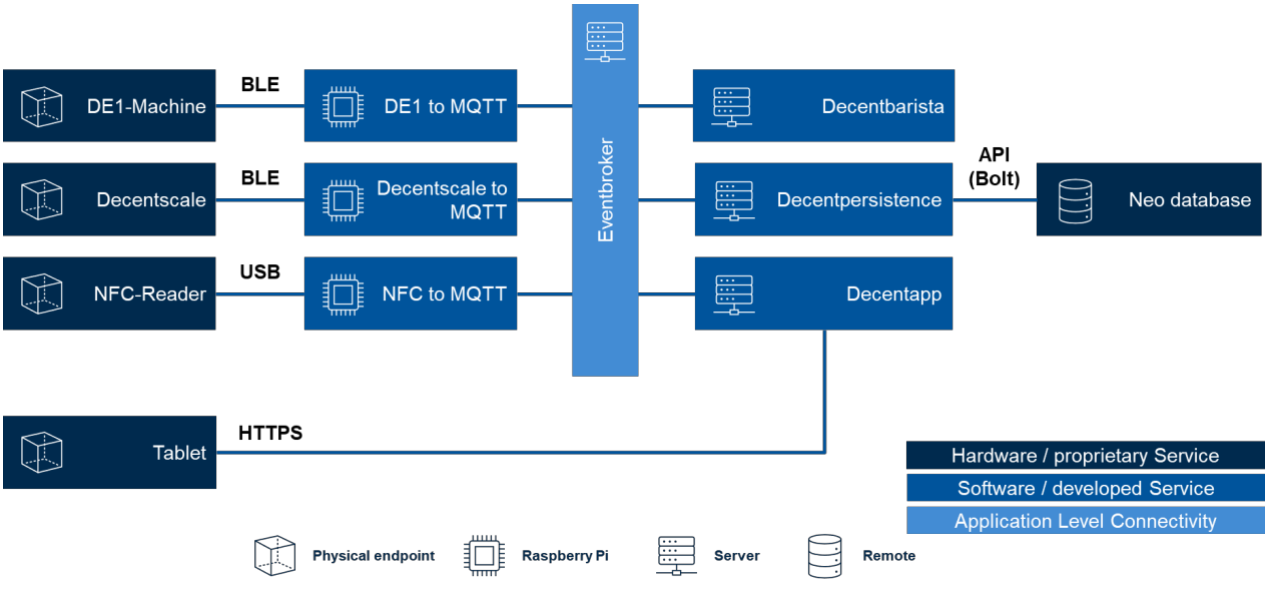


Figure 2: Smart product architecture of the BlueSAM demonstrator utilising the centralised event broker

The realisation (as illustrated in Figure 2) first focused on the digital interfacing of the machine. This was achieved by creating a service running on a Raspberry Pi situated at the espresso machine that both interfaces the machine via Bluetooth and directly communicates with the centralised broker. This way status updates

[4] IGF/CORNET no. 303 EN
[5] Webtool is available at bluesam-tool.fir.de

by the machine were published in the topic *fir.dev.de1* (specifying *dev* for device instead of *ot*) while listening for direct requests to the machine on the topic *fir.dev.de1.request*. The demonstrator required the integration of two more devices handled likewise by the Raspberry Pi, namely an USB NFC sensor for user identification (*fir.dev.de1-nfc*) and a Bluetooth weighing scale for the espresso making process interface (*fir.dev.decentscale*). Three further digital services required for full operation were then iteratively added and incorporated into the EDA and deployed on an on-premise server: A service to supervise the espresso extraction process and command the DE1-machine to stop on a specific weight (*fir.svc.decentbarista*); a service to interface a remote Neo4J database to store and provide process and configuration data (*fir.svc.decentpersistence*); and a user application to provide a web-application as an user interface to a tablet mounted on the machine (*fir.svc.decentapp*). All services were designed to strictly interact only via the broker, reacting to event data and if necessary, placing requests into each other's topics.

The resulting demonstrator was not only capable of realising all use cases envisioned within the research project BlueSAM, but with EDA also offered further non-functional benefits. The services could be adapted independently and deployed anywhere without the need for reconfiguration of the entire system. This not only benefitted the role of the demonstrator to represent a smart product that is situated in the field and is hence mobile. It also allowed for multiple instances of user application to be accessed and used remotely for testing and demonstrating vendor-supervision with each instance receiving live data from the machine. The incorporated responsive design induced by the EDA also deemed to be essential during error fixing. Anecdotally, the remote database crashed in the week of project conclusion and demonstration. Even though the functionality of saving and loading process and configuration data was thus lost, the application and machine control did not seize to respond correctly to the otherwise usual operation. Furthermore, the database functionality was then easily and fully restored by setting up a new local database from a backup and linking the responsible *decentpersistence* service to the new database endpoint. Whereas in conventional software/application design a reconfiguration of any service using the database would have been necessary, the decoupled design induced by EDA reduced the outage and maintenance to a precise and limited scope.

5. Conclusion and Outlook

The introduction of EDA in the production environment of the DFA and its surrounding research environment was conducted along the implementation of two use cases as an iterative increment towards the proposed architecture target. In the scope of the use cases the EDA was an effective tool to precisely scope the development of new components within the network and achieve high data availability. With adherence to the design principles of EDA, the developed solutions offered good decoupling and hence robustness in the case of errors. As the achieved state of realisation for now only focuses on two use cases in a small development team, the effectiveness of multiple top level domain design (*dfa*, *dev*, *ext*, *fir*), of the architectural governance via the change management process and the overall management of authorisation in the broad data space could not yet be validated.

For now, the overall applicability of EDA regarding the DFA is perceived as good. The diverse environment at the Cluster Smart Logistics offers availability of expertise for technical enquiries, software development, IT infrastructure and process knowledge, which proof to be of great value in the self-managed development of a data ecosystem. Even in its infancy the project has gained prominence and broad interest within the Cluster, which stems from the promise of high data availability and the reactive, real-time design.

The overall applicability in the context of other production depends both on the heterogenous nature of the environment and the willingness to develop and self-manage such an ecosystem. From experience even small factories offer enough complexity due to organisational divisions, which fulfil the heterogeneity of interest in the scope of a shared data set. The willingness itself is dependent on the individual IT-strategy of a

company. A pursuit of event-driven IT-architecture is not advisable if the IT-strategy does not include own digital expertise and organisational adaptability or a third-party market solution in its standard does suffice.

The continuation of the project will focus on the concurrent iterative addition of use cases into the EDA of the DFA. The current roadmap includes the integration of order data via a microservice interfacing the ERP, the creation of worker terminals for active orders and the implementation of external data spaces with use cases from external partners of the Cluster's Center Connected Industry CCI and Center Integrated Business Application CIBA.

Acknowledgements

Part of this report is the scientific result of a research project BlueSAM undertaken by the FIR at RWTH Aachen University (Germany) and Sirris (Belgium). The research project was carried out in the framework of the industrial collective research programme (IGF/CORNET no. 303 EN). It was supported by the Federal Ministry for Economic Affairs and Energy (BMWi) through the AiF (German Federation of Industrial Research Associations e.V.) based on a decision taken by the German Bundestag

Appendix

Addition on 2.2 Digital Sovereignty: **Technical dimension on digital sovereignty**

When it comes to technical solutions to achieve digital sovereignty, operation of the IT solutions must be thoroughly considered. Usually, different parts of a complex (naturally distributed) ecosystems have different requirements to availability, resilience, and fault tolerance. Thus, these components can, in principle, be operated, with different SLAs. By buying or outsourcing crucial components of the digital infrastructure, full data sovereignty must be secured in the operations by contracts. The danger of disadvantageous vendor lock-in and reliability on third parties resides. Open-source solutions and on-premise operations are a way out to achieve complete control but come with the cost to secure the knowledge and work power in the organisation to guarantee stable operations and demanded SLAs.

Addition on 2.3 Agility and agile software development: **Domain driven design (DDD)**

DDD centres the development around a model and language common to developers and domain experts, strengthening the cross-functional character of and incorporating communication into the development process [24]. With DDD collaborative modelling techniques arose like domain-story telling [25] or event storming [26]. These incorporate sociological/sociotechnical and psychological research [27–29] and bring together different stakeholder groups to make the most of shared knowledge.

Addition on 2.4 CI/CD – Continuous integration and development/deployment: **Software Development process**

Following the agile philosophy, software is developed in short iterations, each delivering small, valuable increments, developed on small, short-lived feature branches within the version control system. These branches are merged into one common main branch that always needs to be stable and releasable (in the current as well as in all former versions). During the software development process, developers ensure that the software meets certain functional requirements by writing and maintaining sets of tests that describe aspired quality standards. Successful tests serve as the quality gate and are integrated in the development process via test-driven or behaviour-driven development to ensure correctness of both, technical and business-behaviour of the service, at best always including stakeholders in the process. So-called pipelines run these tests automatically before merging a new feature into the stable main branch. Once the pipeline succeeds, the new version of the software can be released into production.

Addition on 2.4 CI/CD – Continuous integration and development/deployment: **Automated Pipelines**

Automated Pipeline enforce documented and repeatable processes, decrease error rates and shorten mean time to restore by automating rollbacks (i.e., deployments of known stable versions). Failures in a pipeline prevent faulty versions to be rolled into production, and thus enable fast feedback cycles by braking builds. This enables highly endorsed continuous learning processes [30]. The term 'DevOps' is often used for this set of methodologies. From the DevOps culture and DevOps topologies ultimately Team Topologies arose [17], ideas on how to streamline development teams along the value chain to enable for fast development, improvement of production software and maintenance of the software development and production infrastructure. Agility and adoption of a DevOps mindsets need to be embedded in the core of an organisations value chain to make it efficiently work. Again, Conway's law applies - digitalisation at full scale that aims to achieve digital sovereignty, demands a structure that is fully organised around their digital value chain.

***Addition on* 3 Proposed solution: Information on the Demonstration Factory Aachen (DFA)**

The DFA represents the practical test ground within the Cluster Smart Logistics in Aachen. While being an independent organisation executing individual production to order, it offers capabilities to develop, test and demonstrate applications and technologies related to Industry 4.0 within a genuine production environment. Stakeholders of such demonstrators are research entities at the cluster as well as partners using the environment for individual development. Over time, this environment has evolved into a heterogenous IT-landscape with numerous isolated solutions that are organisationally, and hence functionally, unrelated. To centrally organise the development of solutions within the DFA, and to allow for implemented solutions to build on each other regardless of their owner, the introduction of an IT-architecture that focuses on digital sovereignty for the DFA as the de facto central entity was deemed necessary. Having a broad spectrum of capabilities available at the Cluster, the self-reliant technical conventionalisation and implementation of digital solutions does not present an obstacle. EDA offers a suitable solution to manage multiple stakeholders acting within a data ecosystem with heterogenous solutions, by abstracting interfaces into a centralised, collaborative dataspace to reduce organisational and technical dependencies [3].

References

[1] Kagermann, H., Lukas, W.-D., Wahlster, W., 2011. Industrie 4.0: Mit dem Internet der Dinge: auf dem Weg zur 4. industriellen Revolution. VDI Nachrichten (13), 2.

[2] BMWK, 2022. Mittelstand Digital - Demonstrationsorte. https://www.mittelstand-digital.de/MD/Navigation/DE/Praxis/Demonstrationsorte/demonstrationsorte.html. Accessed 15 December 2022.

[3] Kremer, S., Konrad, L., Stroh, M.-F., Schuh, G., 2023. Event-driven IT-architectures as enabler for Industry 4.0. Hannover : publish-Ing.

[4] Duggan, D., 2012. Enterprise software architecture and design: Entities, services, and resources. Wiley, Hoboken, N.J., 482 pp.

[5] Jahn, U., 2021. Verteilte Systemarchitektur für mobile Roboter. Dissertation, Bielefeld.

[6] Confluent Inc., 2020. Event-Streaming als Herzstück der Industrie 4.0: Whitepaper, 12 pp. https://assets.confluent.io/m/119f24bff45a5865/original/20201231-WP-Event_Streaming_at_the_Core_of_Industry_4-0-DE.pdf. Accessed 28 January 2022.

[7] English, J., 2019. Event Mesh: Event-driven Architecture (EDA) for the Realtime Enterprise, online, 10 pp. https://solace.com/resources/stream/wp-download-intellyx-eda-for-real-time-enterprise-nov19?utm_source=Email&utm_medium=Direct&utm_campaign=EDA-Nurture&utm_content=Intellyx-EDA-Paper&mkt_tok=MDcyLUNCSS05MjUAAAGCfiI5xBZdldMPBNgIGuxRDD9sKOol08MCnX0bnaz6Zw8rEsVRThL1qpeA39ADgNNvUtLghxGoTj27yUvAjuK-LHm_ca0NPASoRn1Q0CnluZE. Accessed 9 February 2022.

[8] Confluent, Inc., 2022. Confluent-Kunden: Echtzeit-Event-Streaming für alle Branchen | DE. https://www.confluent.io/de-de/customers/. Accessed 31 January 2022.

[9] Kai Waehner, 2021. Real-World Deployments of Kafka in the Automotive Industry - Kai Waehner. https://www.kai-waehner.de/blog/2021/07/19/kafka-automotive-industry-use-cases-examples-bmw-porsche-tesla-audi-connected-cars-industrial-iot-manufacturing-customer-360-mobility-services/. Accessed 28 January 2022.

[10] Solace, 2022. PubSub+ Customer Success Stories - Solace: Powered by Solace - From startups to the world's largest companies. https://solace.com/company/customers/. Accessed 31 January 2022.

[11] Bruns, R., Dunkel, J., 2010. Event-driven architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse. Springer, Berlin, 241 pp.

[12] Hastbacka, D., Kannisto, P., Vilkko, M., 2018. Data-driven and Event-driven Integration Architecture for Plant-wide Industrial Process Monitoring and Control, 2979–2985.

[13] Jakubowski, S., 2020. Warum Event Driven Architecture für Ihr Unternehmen sinnvoll ist. https://newcubator.com/blog/warum-event-driven-architecture-fuer-ihr-unternehmen-sinnvoll-ist. Accessed 11 February 2022.

[14] Wipro Digital, 2020. A Guide to Enterprise Event-Driven Architecture. https://wiprodigital.com/2020/11/10/a-guide-to-enterprise-event-driven-architecture/. Accessed 14 January 2022.

[15] Hummel, P., Braun, M., Tretter, M., Dabrock, P., 2021. Data sovereignty: A review. Big Data & Society 8 (1), 205395172098201.

[16] Forsgren, N., Humble, J., Kim, G., 2018. Accelerate: The science behind DevOps : building and scaling high performing technology organizations. IT Revolution, Portland, Oregon, 257 pp.

[17] Skelton, M., Pais, M., 2019. Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution Press, California, 5 pp.

[18] M. E. Conway. How Do Committees Invent? 1967.

[19] Lise Hvatum, Allan Kelly, 2016. What do I think about Conway's Law now?: Conclusions of a EuroPLoP 2005 Focus Group.

[20] Mennerich, C., Meseck, F., 2020. Systemtheorie und Softwaredesign: Wie Soziologie und Softwareentwicklung zusammenspielen können 4, 48–53.

[21] Kent Beck, 2001. The Agile Manifesto. https://agilemanifesto.org/. Accessed 29 July 2023.

[22] Evans, E., 2004. Domain-driven design: Tackling complexity in the heart of software / Eric Evans. Addison-Wesley, Boston, London.

[23] Biedermann, E., 2023. OPC UA versus MQTT: Gelingt der direkte Vergleich? WEKA Industrie Medien GmbH, January 1.

[24] Vernon, V., 2013. Implementing domain-driven design. Addison-Wesley, Harlow.

[25] Hofer, S., Schwentner, H., 2022. Domain storytelling: A collaborative, visual, and agile way to build domain-driven software. Addison-Wesley, Boston, 1 online resource.

[26] Alberto Brandolini, 2021. Introducing EventStorming: An act of Deliberate Collective Learning. https://leanpub.com/introducing_eventstorming. Accessed 3 July 2023.

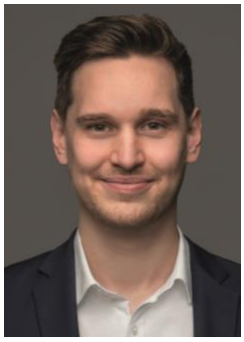
[27] Kahneman, D., 2012. Thinking, Fast and Slow. Penguin Books, London.

[28] Syed, M., 2016. Black Box Thinking: Marginal gains and the secrets of high performance, First published in paperback ed. John Murray (Publishers), London.

[29] Taleb, N.N., 2007. The Black Swan: The impact of the highly improbable. Random House, New York. N.Y.

[30] Humble, J., Farley, D., 2011. Continuous delivery. Addison-Wesley, Boston, Mass., London.

Biography



Sebastian Kremer, M.Sc. (*1994) is head of division Information-Technology-Management and PhD candidate at the FIR e.V. at RWTH Aachen University since 2020. As a senior project manager, Kremer executed research projects in the field of IT-OT-integration towards a digitally connected factory and industry projects for the development of modern IT-architectures for producers and logisticians in Germany.

Dr.-Ing., Dipl-Inf. Christian Mennerich (*1981) has studied computer sciences with a focus on theory and implementation of database systems. He started working with synyx GmbH & Co. KG in 2013. Here, he is occupying himself with NoSQL and message-oriented distributed systems. Mennerich talks at conferences and user-groups, writes articles and occasionally edits books. Since 2022, he is lives in England and establishes synyx UK.

Max-Ferdinand Stroh, M.Sc. (*1991) is head of division Information-Management and PhD candidate at the FIR e.V. at RWTH Aachen University since 2018. As a senior project manager, Stroh executed research projects in the field of cyber-physical systems, smart products and IT-OT-integration towards a digitally connected factory.

Dr.-Ing. Dipl.-Wirt. Ing., Univ.-Prof. Günther Schuh (*1958) is director at the FIR e.V. at RWTH Aachen University. As a directing professor Schuh oversees digitalisation projects at the Institutes IPT, WZL and FIR at the Campus Melaten in Aachen.