International Journal of Information Technology, Control and Automation (IJITCA)
Vol.11, No.2/3, July 2021

CaRE –an Architectural Approach for a Multimedia Assistance System for Singletons

Thomas Schirgi and Eugen Brenner

[1]Institute of Technical Informatics, Technical University, Graz, Austria

ABSTRACT

In contrast to the increasing degree of automation in the production industry, commissioning and maintenance activities will essentially be limited to manual activities. Production involves repetitive actions that are manageable and clearly defined as a process. Unlike this, commissioning and maintenance have to deal with uncontrollable, undefined, and non - standardized processes. The paper provides a framework for a multimedia assistance system for singletons. It was found that the paradigm has to consist of five key components to provide tailored assistance to customers. These key components are Expertise, Infrastructure, Application & Platforms, Security & Privacy and Business Process & Business Model. The resulting stack and the overlaying business model are called "CaRE – Custom Assistance for Remote Employees". With possible Architectural Smells and Anti-Pattern in mind, a Microservice Architecture shall be presented which forms the backend-system of CaRE.

KEYWORDS

Multimedia, Assistance, CaRE, Software, Microservices, Architecture, Anti-Pattern, Architectural Smells.

1. Introduction

During the last decades, many processes and workflows, especially in production environments, changed significantly. Thereby some of the processes are getting more and more complex and sophisticated. With the help of new information systems, it is tried to meet these challenges. The fourth industrial revolution and the fusion of humans and machines to form cyber-physical systems (CPS) are revolutionizing processes and procedures, especially in production-related companies. One of the consequences is that maintenance and service activities are becoming more and more complex and time-consuming. "Smart Maintenance" results in entirely new interconnection, qualification, and support requirements for people involved. Where maintenance-relevant data previously had to be entered manually, these can also be recorded automatically in the future.

In contrast to the increasing degree of automation in production, maintenance and servicing will also in the future be essentially limited to manual activities. Production involves repetitive activities that are manageable and clearly defined as a process [1]. Contrary, maintenance and servicing have to deal with "uncontrollable," undefined, and, above all, non-standardized processes. Furthermore, this is of utmost importance whenever commissioning or maintenance has to be done at singletons in electrical environments. To help users during their tasks, a multimedia assistance system can be used.

For users, the most important part is, of course, the front end of an assistance system. Nevertheless, the major aspect of the system is running behind, with interfaces to other systems

DOI:10.5121/ijitca.2021.11301

as well. It has been decided to implement this architecture in terms of a Microservice Architecture because the system is expected to be flexible and capable of dealing with different other systems.

Structure

In chapter 2, the basic principle of Microservices shall be described. In chapter 3 the CaRE pyramid and the items are described. In chapter 4, the Microservice Architecture of CaRE is presented. Finally, the conclusion and future work is represented in chapter 5.

Related Work

With products and singletons becoming more and more complex, it is logistically and for cost reasons, not always possible to send a specialist to the site. As a result, maintenance personnel often have no previous experience with a product to be commissioned or serviced. Nevertheless, implicit knowledge is necessary to carry out most of the work which has to be done on-site (see [2]). In literature, many possible solutions interms of “Smart Factory” ([3], [4], [5]), “Cognitive Assistance” ([6], [7]) or “Industry 4.0” ([8], [9])are mentioned. However, there are no examples of how employees can be supported if they are

1. not in a factory,
2. do not have a basic knowledge of the product to be maintained and
3. dealing with singletons

Especially in the energy sector, long-living singletons are common. Parts of an energy network may last for an entire generation in a company. Since such substations are often located outside inhabited areas, the cellular Network might be insufficient. Furthermore, with the help of mobile phones and email, important information might get lost. Multimedia Assistance Systems partially support non-verbal communication, which means that information can be made available more efficiently. Therefore, a flexible backend system, which provides data from various sources, needs to be developed.

2. MICROSERVICES

Since Microservices (MSA) were first mentioned by Fowler and Lewis in 2014 (see [10]) this type of architecture seems to be a paradigm change in software development. The Hype Cycle for Application and Integration Infrastructure published by Gartner in 2019 shows the increasing popularity (see [11]). Many of IT Companies – like the FANG companies – deliver their services based on Microservices. Contrary to that, a widely used architecture is a monolith, where all functionality of an application has to be deployed together (see [12]).

2.1. Differences between Monolithic, Microservices and Service Oriented Architecture

Basically, the main features of Service Oriented Architecture (SOA) and Microservices are comparable. Both depend on (hybrid) cloud environments to provide and execute applications. Both split a huge and complex application into smaller parts and can combine one or more required services to create and use applications. Microservices are defined as “independent deployable” modules and can be seen as an extension to service-oriented architecture (see [13]).

In the table below, the main differences are stated (see [14] and [15]):

Table 1: Differences between Microservices, SOA and Monolithics

	Microservices	**SOA**	**Monolithic**
Architecture	Host Services, operating individually	Provide Resources, commonly used by services	Single Architecture with Single Development Stack
Common use of Components	Components not shared but Shared Libraries (connectors, etc.)	Components shared	Components in terms of shared libraries
Granularity	Differentiated Services	Bigger, strong modular Services	Single Unit
Data Storage	Own Data Storage	Commonly used Data Storage	Mostly one data Storage
Governance	Collaboration between teams necessary	Common Governance Protocols, Team overarching	Governance not necessary
Size and Scope	Better for small, web-based Applications	Better for big integrations	Size varies, Scope dependent
Communication	API Layer	Enterprise Service Bus	Typically: Presentation Layer – Business Logic Layer – Data Access Layer
Coupling and Cohesion	Limited Context for the Coupling	Resources are shared	No Resources are shared
Remote-Services	REST or Java Message Service	SOAP or AMQP	Independent eventually proprietary
Deployment	Fast and Easy Deployment	Lack of Flexibility at Deployment	Lack of Flexibility and Deployment

2.2. Advantages and Disadvantages of Microservices

Microservices incorporate a lot of advantages and disadvantages. In the following section, some of the most important pros and cons shall be discussed:

Scalability

Each of the Microservices can be scaled individually. In comparison to monolithic architectures – where multiple instances are hard to implement – it is possible to implement Microservices in a fine-grained structure with fewer services. Furthermore, it is feasible to deploy multiple instances of services for load balancing reasons. With this scalability, exchanging of services is also viable easily (see [12]).

Technology Independent

For different purposes, various technologies can be used. The most suitable tool for each task can be selected. Each Microservice can easily be migrated to other – even new – technologies. This technology independence covers different database systems, different interfaces, and different programming languages (see [16]).

Resilient

Assuming that the network is reliable, Microservices are more resilient than monolithic approaches. If one service fails, the other services may remain up and running. In other words

failures are not cascading and cause a total app failure. Even if other Microservice needs to compensate for the failed service, values can be cached and used in case of failures (see [12]).

Deployment

Microservices also offer advantages for development teams and deployment of services itself. With smaller services, new developers apprehend the structure more easily. Each Microservice can be deployed independently, as required, enabling continuous improvement and faster updates. Specific Microservices can be assigned to specific development teams, which allows them to focus solely on one service or feature, also technology-wise. This means teams can work autonomously without worrying about the performance within the rest of the app (see [16]).

Complexity

The communication between services can be complex. With a huge amount of services, debugging can be challenging. This challenges occur if each service has its own set of logs. Additionally, testing might be hard since Microservices might be distributed and not running on one single computer (see [17]).

Controllability

With a huge amount of Microservices depending on each other, simple changes of APIs are dangerous. These changes need to be backward compatible, otherwise, each other Microservices depends on the changed one (see [17]).

Costs

If Microservices run on different hosts, the hosting infrastructure, security and maintenance are more expensive. Furthermore, interacting services require a huge amount of remote calls. This will also increase the network latency and processing costs (see [16]).

Both SOA, Monolithic and Microservices incorporate advantages and disadvantages. Thus, Microservices are no patent remedy for each type of application. It has to be determined if a Microservices architecture has more advantages rather than disadvantages for the type of application that needs to be implemented. If using Microservices distributed transactions and the CAP Theorem needs to be minded. Furthermore, an appropriate and good architecture has to meet some quality criteria.

2.3. Microservice Architecture Best Practices

To create a sustainable architecture for Microservices, some basic principles must be met (see [18]):

- A Microservice Architecture must split its item into modules.
- Modules should interact via non-proprietary interfaces.
- Specific modules should not depend on implementation details of other modules.
- Integration options must be limited and standardized.
- Communication should be limited to RESTful Web-Services or Messaging Protocols.
- Authentication must be standardized on all services.
- Each Module should have a continuous delivery pipeline.
- The operation should be standardized: Same configuration, deployment, log analysis, etc.

- Modules should be resilient. Modules should not fail if other modules are not reachable.

If all of those principles are met, it is not totally sure that the architecture is well designed and without any mistakes. The complexity of an MSA leads to several attack surfaces in terms of bad decisions when defining and implementing the architecture. Therefore, it can be distinguished between Architectural-Smells and Anti-Pattern. Sometimes Anti-Pattern and Architectural-Smells are seen as the same issues. Also, the transition in literature is unclear and sometimes blurred. To distinguish, Architectural Smells and Anti-Pattern can be defined as followed: (see [19])

- Architectural Smells: Smells should be investigated. Smells may and may not be bad.
- Anti-Pattern: This will lead to a worse design of a system. Anti-patterns are always bad and should be solved immediately.

In the following sections, some Architectural Smells and Anti-pattern concerning Microservices are discussed. It was tried to distinguish between Bad Smells and Anti-pattern, but – as mentioned – this difference is blurred.

2.3.1. Architectural Smells

In this part, some examples of Architectural Smells within MSA shall be discussed (see [20], [21], [12]):

- Hard-Coded Endpoints: IP Addresses and ports are hard-coded of the service which need to be used.
- Shared Persistence: Different Microservices are accessing the same database.
- Independent Deployability: In MSA, each service should be running independently from another service, so it should be possible to deploy and un-deploy a Microservice independently from others.
- Horizontal Scalability: A consequence of independent deployability is the possibility of adding and removing replicas of a Microservice.
- Isolation of failures: Failures should not be cascaded to underlying services, which means that a failure in Microservice A results in triggering a failure in Microservice B.
- Decentralization: In all aspects of Microservice-based Applications, decentralization should occur, implying that the business logic should be fully decentralized and distributed.

2.3.2. Anti-Pattern

Anti-Pattern are leading to a worse design of an Architecture. According to [21], Anti-pattern can be divided into four groups (also see [19]):

1. Design Anti-pattern:
    - Wrong Cut: Occurs if Microservices are split based on technical layers (presentation, business and data layers) instead of business capabilities, which leads to higher complexity
    - Cyclic Dependencies: Microservices involved in cyclic dependencies can be hard to maintain or reused in isolation.
    - Nano Service: A Microservice should be designed to fulfil single business capabilities, not more but also not less.

- Mega Service: As contrary to Nano Services, also Mega Services should be avoided.

2. Implementation Anti-pattern:

    - Shared Libraries: Microservices should not share libraries and source code directly.
    - Too many standards: Although Microservices are allowing different technologies, too many different protocols, frameworks, development languages, etc., should be avoided.
    - Too new technology: The technology used needs to be defined well. Too new technology is not always the best choice since it might not be fully developed.

3. Deployment Anti-pattern:

    - Manual Anti-pattern: Everything, which is possible, should be automated. Therefore, a configuration server should be used, which automates the configuration process.
    - No Continuous Integration (CI) / Continuous Delivery (CD): The independent deployability of Microservices also offers the possibility to apply iterative, continuous development and deployment (DevOps) processes.
    - No API Gateway: Microservices are communicating directly with each other. If no API Gateway is in place, the service consumers are also communicating directly with the Microservices. This increases the complexity of the system but also makes a system harder to maintain.
    - Timeouts: In distributed systems, consumer applications/tasks use timeouts to handle unavailability or unresponsiveness.
    - No API Versioning: In some cases, multiple Versions of APIs must be exposed by a service

4. Monitoring Anti-pattern

    - No Health Check: A health check API endpoint should be implemented, which periodically verifies the health status and the ability to answer requests.
    - Local Logging: During run-time, each Microservice produces a lot of information with logging. Therefore, a central logging service should be implemented and used. Distributed logging mechanisms are easy to implement and make debugging more accessible (like Elastic Stack).
    - Insufficient Monitoring: If Microservices are part of Service Level Agreements (SLA), the behavior and performance are crucial. To solve this, a global monitoring tool should be implemented.

3. CARE PYRAMID

For a multimedia assistance systems for singletons (both remote assistance and multimedia assistance), a holistic approach must be considered. Five key components are necessary to come over those needs. Those components can be visualized in the form of a pyramid. Those five components can be seen as parts of the pyramid which rely on each other (see [22]):

- Expertise
- Infrastructure

- Applications and Platforms
- Security and Privacy
- Business Processes and Model

The pyramid itself cannot be inverted. Those five components can be applied to both multimedia assistance and remote assistance, where an expert is assisting through telephone or "I see what you see" application. In both cases, those five components have to be fulfilled to provide successful assistance:

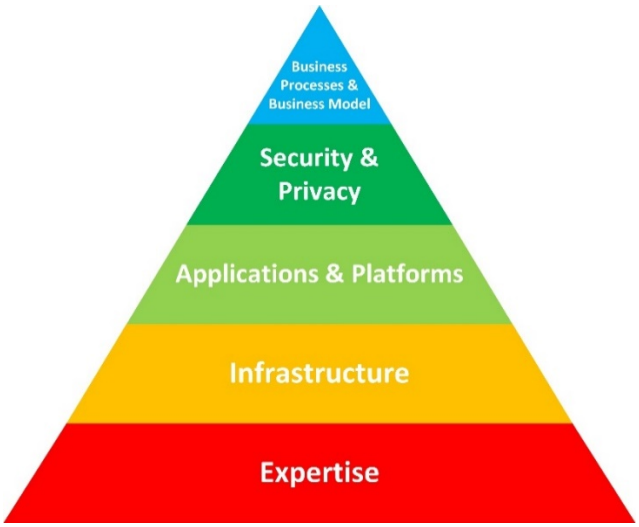

Figure 1: CaRE Pyramid

3.1. Expertise

Expertise describes the basis of the stack, as seen in Figure 1. Without domain knowledge, assistance is meaningless and not applicable. With remote assistance, it is likelier that the desired expert is available immediately. For example, field engineers have to wait more than one week to get working permissions in some cases. Furthermore, the daily fees for field engineers are expensive, which means that customers have to pay thousands of euros for unproductive dwell times. In this case, it would be possibly cheaper to send engineers from the customer to the site and help them via remote assistance. Furthermore, a single expert can work on more than one site in parallel, even if these sites are located in different places. Another benefit is that there is the possibility to accomplish a group conference call if more than one expert is needed.

When assisting with multimedia assistance applications, the users at the site need to have at least basic domain knowledge. If, for example, a specific jargon is used during maintenance or commissioning, the workers at the site need to at least have a basic understanding.

Moreover, the work on-site must be done in any case. The work which needs to be done is manual and cannot be done by robots or artificial intelligence: Traditional work will not go extinct.

3.2. Infrastructure

The infrastructure part can be divided into a network and hardware part.

3.2.1. Network

A comprehensive availability of broadband internet is necessary for both parties. This is both important for video- and audio conferring, as well as for sending data over the Network. According to recent studies, in more than 80 countries, more than half of the area is covered by 4G [23]. In the US, for example, the coverage is more than 90%. However, poor network quality and low bandwidth remote assistance will lead to dissatisfaction.

Since substations may be in regions that are not entirely inhabited, the internet connection might not be sufficient. Therefore, a small box with an embedded router was implemented. The so-called “Remote Access Box” consists of a switch-mode power supply that allows AC/DC and different voltage levels. The power supply cable is connected with so-called bayonet connectors, which are also safe against water intrusion and minor impacts. Whenever the power supply is connected, the box starts automatically. In closed operation, without any connected periphery, the box reaches IP protection class 64.

Inside the box, an LTE router is mounted. Two antennas are located inside the case; for the other two antennas, ducts in the surface are available. These two antennas do have a five-meter cable and can be placed at windows or doors. To connect to the wireless Network, the QR Code on the surface can be scanned.

The SIM card slot is also available with a duct at the surface. In addition, two RJ 45 Ethernet connectors are available for connections with LAN cables. With those cables, it is also possible to supply a wired internet connection.

After booting, the router automatically connects with the VPN Server inside the company network. Thus, a secure communication is guaranteed.

3.2.2. Hardware

In terms of Hardware, it must be distinguished between remote assistance and multimedia assistance.

Remote Assistance

In maintenance, the service technicians often need both hands free and have to be mobile. Due to that fact, the first evaluation mainly focused on wearables. Four types of wearables are available. [24]

- binocular and non-see-through
- binocular and see-through
- monocular and non-see-through
- monocular and see-though

A literature review has shown that some key criteria have to be met to provide a useful wearable in a maintenance context. ([24], [25], [26]) Those criteria are:

- Wearing Comfort
- Battery Lifetime
- Field of View
- Navigation

Wearing Comfort

In the case of service activities, the device may be worn over a more extended period. The musculature in the neck area is stressed, even if the wearable itself is not heavy. Some devices use nose bridges, brackets for ears, or some sort of neck holder. The device should not be too heavy, but it should be rugged if it falls to the ground. The device should offer the possibility of individual wearing settings. It has to sit well because the wearing feeling is mainly influenced by the mounting method. Users have to trust that the device is seated well and won't come off suddenly.

Battery Lifetime

The time assistance is needed during maintenance may be between some minutes or hours. Even if the device is not required during the entire working process, the battery shall be durable and have a battery life of approximately one working day. It is worse if the device battery is getting empty when assistance is needed. Therefore, the battery of the device should be fast loading or replaceable.

Field of View (FOV)

The standard FOV of humans is 200° horizontally and 135° vertically. [27]The user sight should not be limited significantly and the users should not lose their situational perception. With a limited field of view, the eyes are stressed, which leads to eye strain and dry eyes. [24] A wide field of view is desirable and also linked to improved user acceptance. It should furthermore be possible to wear eyeglasses beyond the wearable.

Navigation

It must be easy to navigate inside the operating system of the wearable. Therefore, these three input methods are available:

- Buttons directly at the device
- Speech
- Gestures

All of these three types are not useful during maintenance operations. Therefore, a companion app should be available. With these apps, it is possible to interact with the devices. For example, they may offer a keyboard that makes entering text more accessible.

Tests were conducted with the Microsoft Hololens (binocular and see-through), the Daqri Smart Glass (binocular and see-through) and the Realwear HMT-1 (monocular and non-see-through). Binoculars which are non-see-through are not suitable for maintenance activities. Also, Augmented Reality is not necessarily needed. Additionally, it has to be clarified if a full-face display is required. If not, wearable with a microdisplay or a tablet may also fulfill the needs during remote assistance.

It could be seen that situational perception suffers when full-face displays are used. For example, test persons could not see harmful items, even though they occur in their standard field of view. The Realwear HMT-1 has a microdisplay mounted on a small arm. Due to that, it is possible to move the display out of the field of view whenever it is not needed.

Furthermore, it could be seen that the Hololens and the Daqri Smart Glass are becoming heavy, although their weight is only around 370g. Wearing the Hololens or the Daqri Smart Glass underneath a helmet is physically exhausting over a more extended period. Therefore, a device is required which is comfortable to wear for the user. It is often necessary to wear personal safety equipment such as safety glasses or safety helmets in production environments. If engineers wear the Hololens, it is essential to wear safety glasses underneath. In contrast to that, Daqri Smart Glass would count as personal safety equipment (PSE).

Additionally, the Hololens is heating up at the processing unit located right above the ear. The Daqri Smart Glass computing unit is an external device, which can be mounted on a belt. In the case of menu navigation during usage, both Hololens and Daqri Smart Glass are using gestures. It takes some time to learn those gestures, but users got more and more experienced with these gestures after a short time. In contrast to that, the Realwear HMT-1 uses speech recognition for navigation. However, in environments with high sound intensity, speech recognition is not suitable.

Based on the evaluation results, it was decided to use the Realware HMT-1 for remote assistance activities. Despite having the smallest display has the best wearing comfort of all tested devices. Furthermore, it offers the possibility to replace the battery if needed and uses an Android operating system which is standard on many devices.

Multimedia Assistance

During commissioning and maintenance, supervisors must be mobile. This means that only a part of devices is suitable as the basis for multimedia assistance applications. Thus the following types were used during the tests:

- Tablet Computer (Vendor Samsung)
- Cell Phone (Vendor Apple)
- Laptop Computer (Vendor HP)
- 2-in-1 Computer Convertible (Vendor Microsoft)

It could be seen that the display of a cellphone is too small for usage. Even if the application is responsive and fits itself to the available screen size, the user interface is no more user friendly. Even if users are familiar with scrolling inside apps, the lists are becoming too long, and searching for specific items is hard. Usage of a tablet computer solves the issue with the screen size. Also, the interaction with fingers in different parts of the application (like annotating) is working well. As a drawback, it must be mentioned that the input of long character sequences is not easy since the ten-finger typing is hardly possible: The heel of the hand cannot be placed in the same way as on usual keyboards, which leads to an unfamiliar posture.

Furthermore, the feedback of the screen keyboard is also unfamiliar in terms of the ten-finger system. As a result of those two drawbacks, only a Laptop computer or 2-in-1 convertible seems suitable for those actions. Since it is sometimes necessary to climb on top of the workpiece, Laptop computers might be impractical. Furthermore, Laptop computers sometimes only have one camera, which is located in front of the screen. With this location, it is nearly impossible to take pictures. If no desk or position is available where the Laptop computer can be placed, it is also impractical to hold the Laptop computer with one hand, whereas the other hand is used to navigate. Even if the Laptop computer weighs only 1.74kg, the device becomes heavy immediately.

Based on the results of the evaluation, it was decided to use the convertible. Despite having the worst battery, it has the best features and characteristics of all tested devices.

To summarize, the following Hardware was selected as appropriate for an assistance system:

- The Realwear HMT-1 for remote assistance calls. Even if some advertisements show the benefits of apps running on such wearables, it could be seen that this is impractical in real life. As stated above, the perception is suffering, and it is exhausting. Therefore, those devices should only be used if needed for remote assistance calls.
- A 2-in-1 convertible for the usage of the Multimedia Assistance System. Since these kinds of devices are both laptop and tablet, they are flexible in many cases. For multimedia assistance, they can be used to take pictures and make quick notes and documentation in the site office.

3.3. Application and Platform

A supporting application or platform is necessary for sharing knowledge and providing help during commissioning and maintenance activities. In terms of Applications and Platforms, it can be distinguished between remote assistance and multimedia assistance.

3.3.1. Remote Assistance

There are many possible solutions for remote assistance which are already in place. Some of them are listed below:

- Librestream (https://librestream.com/)
- Fieldbit (https://www.fieldbit.net/)
- Stream (https://www.streem.com/)
- Skype for Business (https://www.skype.com/de/business/)
- Teamviewer Frontline (https://www.teamviewer.com/de/loesungen/frontline/)
- and many more.

All of the mentioned systems have their unique selling proposition. After deciding to go along with Realwear HMT-1 for remote assistance, the application Skype for Business, Librestream and Teamviewer was evaluated because licenses were available for further researches. It was decided to use an existing one since all the required features below are already in place in all of the mentioned ones. It would make no sense to develop an own one in this case.

After discussions and interviews with the target group, criteria for a software platform that is usable in maintenance activities were identified:

- Usability & User Acceptance
- Annotations or Pointer Functionality
- Mutliconferencing
- Chat Function
- Reporting

Usability & User Acceptance

The key factor of the software is that it has to be accepted by the users both in the field as well as in the office (experts). Therefore, the user interface has to provide the most important

functionalities at a glance so that service technicians can navigate quickly. For example, it should be easy to select the desired expert. The web interface of all systems is user-friendly. Even though Teamviewer and Librestream have more possibilities and features than Skype, the user interface appears to be tidy and well-structured. Depending on the bandwidth and the selected device, the video and audio quality is equal on all systems.

Annotation or Pointing Functionality

The expert should have the possibility to point or annotate certain things in the video. In the field tests, it could be seen that some field engineers – if they are non-native speakers of the language spoken by the expert – have problems understanding what the expert intends. This helps the field engineers to identify objects and makes it easier to understand what the expert is talking about. According [28], visual annotations are a key factor for successful remote assistance. But the usage of annotations on static images is dangerous, especially when using head-mounted devices. They may cause a loss of orientation for a short period, which may become hazardous in industrial environments.

Furthermore, annotations onto a live video stream may limit one's sight. This means that annotations are helpful, but they must be used carefully. Additionally, it should be possible to send documents or drawings, which also might be annotated.

Multi-conferencing

In some cases, one expert alone is not capable of resolving the issues at the site. Therefore the possibility of conference calls should be available. With this feature, a group of people can help one or more engineers at the site.

Chat Function

In some cases – especially in noisy environments – some of the commands to the field engineers are not understandable. In this case, a chat function where an expert can write down requests is necessary. Furthermore, it is imaginable that engineers at the site have to change some parameters of the software. Therefore, it is easier to send text messages with the desired parameter than to describe the values.

Reporting

After finalizing a remote assistance session, it should be easily possible to create a report. So it should, for example, be possible to create a PDF document with all participants written down, the period of the conference, or the text messages sent. This report can be used for internal documentation or as a prove for the customers.

As mentioned above, all of the three systems do have some features and characteristics which are helpful. Thus it cannot be said which of the systems above is the best and which suits most. Skype for Business has the advantage of seamlessly integrating into the Windows environment, which makes it easier to select an appropriate expert. Teamviewer has the advantage of sharing the screen with other participants who can also interact with the remote computer. Librestream has the advantage of providing end-to-end communication with no server in between, which makes it the best in terms of security and Privacy. Unfortunately, there is no mixture of all three systems, selecting the best of all three solutions.

3.3.2. Multimedia Assistance

As already stated, some multimedia assistance applications are already in place. Examples of systems that are already in place are:

- Reportheld (https://www.reportheld.com/)
- Workheld (https://tabletsolutions.at/)
- Evoassist (https://evoassist.evolaris.net/)
- ProWorkFlow (https://www.proworkflow.com/)

Unfortunately, non of them fitted to the needs of the users due to:

- Different contexts of use
- No offline availability
- No spare part management
- No interface for external systems or sources
- Drawings cannot be integrated

So it was decided to create a flexible multimedia assistance application as a prove of concept. Therefore, the CaRE lifecycle shown in Figure 2 was used:

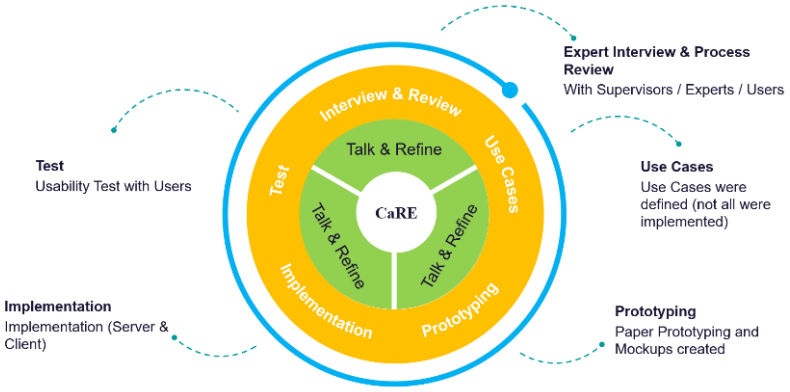

Figure 2: CaRE Lifecycle

As it can be seen, the CaRE lifecycle has its basis in the Deming PDCA (plan-do-check-act) cycle [29] and has the following items:

- Interview and Review
- Criteria Catalogue
- Prototyping
- Implementation
- Test

Above everything, "Talk and Refine" needs to be mentioned, which means that it was communicated with the target group during the entire lifecycle. During the mentioned interview and review part, it was found out that the following topics should be met:

- The application should lower the cognitive workload of the users

- The user interface should only show the topics which are of relevance (parts which are not applicable should be invisible)
- It should be user-friendly and supervisors should get a friendly frontend. If possible, the look and feel of the paper should be indicated.
- Data should be shown in real-time.
- The application should have the same look and feel as it is already known to the supervisors.
- It should work offline without an internet connection.
- New and revised drawings should be available automatically. Furthermore, it should be possible to mark changes on the drawings.
- Serial numbers should be detected automatically.
- It should be able to create markers not to forget work-items
- Data should be already inserted if it is available in another system.

Based on this criteria catalog, a paper prototype was created together with the target group. This prototype was implemented in the form of a prove of concept. Due to Covid-19, no face-to-face usability test was possible. Instead, a remote usability test was done. Therefore an Microsoft Teams session was started. The interviewer launched the application and shared the screen with the participant. The participant turned on the camera and got control over the interviewers' computer. The participant was asked to fulfill the tasks and think loudly. With the camera turned on, the facial expressions and gesturing could be seen.

3.4. Security and Privacy

Applications like CaRE or remote assistance provide new challenges regarding time-sensitive networking (TSN) [30] as well as for real-time networks [31]. Security and Privacy have to be met at all levels all the time. Companies must provide security measures to protect their data, Business, and reputation. Security breaches often lead to loss of reputation and monetary loss. In addition to that, data protection regulations, like the GDPR, force business to protect their data. That means that security is mostly driven from top to down, which also means that the usability of these security measures suffers. In former times, the security measures were often bypassed, which should not happen now.

According to Gartner, 124 Billion Dollars were spent on information security from enterprises in 2019 [32]. Not only enterprise security has a high significance. According to Eurostat, approximately 1 in 3 EU citizens reported security-related incidents in 2019. Especially in the electrical industry, security is essential to ensure an undisturbed power supply. In North America, NERC is responsible for highly reliable and secure power systems.

Furthermore, the NIS Regulation (Measures for the high common security of network and information systems) also enforces high-level security and usability. Therefore, security tools should be as flexible as possible to allow the best user experience possible. Unfortunately, high security often leads to non-user-friendly systems. One way to achieve usable security is to use the "Security by default" approach. The GDPR encourages that security by default is the fundamental factor in achieving usability and secure products. For example, it should be easily possible to remove personal data from a webpage.

3.4.1. Security and Privacy by default

Security by default is a fundamental factor whenever a high level of usability and security want to be achieved. It means that the default configuration is the most secure configuration which is

possible. This should also be the case for Privacy. The principles of "Security by default" enhanced with Privacy are listed below [33].

- Security and Privacy mechanisms should be built into products starting from the beginning.
- The root cause of a threat should be handled, not the symptoms.
- Security must be continued through the entire lifetime of a product.
- Security should not require extensive configuration to work – it should work reliably where implemented.
- Security and Privacy should never compromise usability
- It should defeat the latest threats
- Security through obscurity should be avoided
- Special technical knowledge should not be required

Therefore, security tools should be as flexible as possible to allow the best user experience possible. To achieve this goal, it is essential to incorporate users during the design process to improve the user experience.

3.4.2. Security Controls

Users always want to work in a manner they are comfortable with. For example, a system – even a security system – should not hinder users during their day-to-day tasks. [34]. Often there are multiple ways to achieve a task, and security itself should be accommodating of this. When designing user interfaces, considerations in terms of varying levels of security or risk-based features should be taken into account. For example, user interfaces should use new technologies like biometrics (face recognition or fingerprints) or smart cards. If this is not possible, two-factor authentication and authorization (like SMS PINs) would also be possible. To create secure user interfaces on the web, captcha is visible on more and more systems. On the one hand, captchas are perfect if automated access from robots needs to be prohibited. On the other hand, captchas are not ideal in terms of usability and accessibility. [35]

The world-wide-web consortium defined alternatives for captcha like temporary tokens or multi-factor authentication. For users themselves, it is hindering if authorization and authentication must be done for every transaction. This leads to frustration and a cutoff of security mechanisms. To gain the utmost level of security that users accept, the following topics should be addressed:

- Security must not be hindering: If users feel that the security measured are slowing them during their daily work, the security measures won't be accepted by them.
- Disabled Security measures during commissioning & re-established in production: Security measures should not be disabled during commissioning. This could be a backdoor for cybersecurity attacks and also be forgotten.
- Use Best Practices from other companies with more experience: Companies with less experience in cybersecurity should get help from companies with more experience. Furthermore, best practices should be established.
- Use of Multifactor Authentication or Biometrics: Standard Login mechanisms often lead to cybersecurity holes. Enabling multi-factor authentication or biometrics may address this.
- Amount of login processes: the number of login processes should be kept as low as possible. Users will not accept systems where it is necessary to log in for each transaction.
- Disabling of default users: Default users should be disabled or deleted.

3.5. Business Process and Business Model

A business model for revenue strategy needs to be implemented. Even though remote assistance is faster and cheaper than sending people directly to the site, it must be accounted for. So it might be possible to offer remote assistance over a defined period for free. If more time is needed, the billing is at cost, which might be cheaper than a flight, daily allowance and other fees. The time of assistance can be recorded and accounted minute-wise. With video recording, it is easy to document the work done and these videos may then be used during training. Another advantage is that customer satisfaction may increase due to the quickness of the service. This may also lead to a long-term partnership between the vendor and the customer.

4. MICROSERVICE ARCHITECTURE OF CARE

In this section, the Microservice Architecture of CaRE shall be described. Since it has been decided to use an application which is already in place for remote assistance, only the architecture for multimedia assistance system shall be described. The architecture has been implemented in terms of an proof of concept.

With the best practices in mind, a Microservice Architecture has been created. An architectural overview is shown in Figure 3.

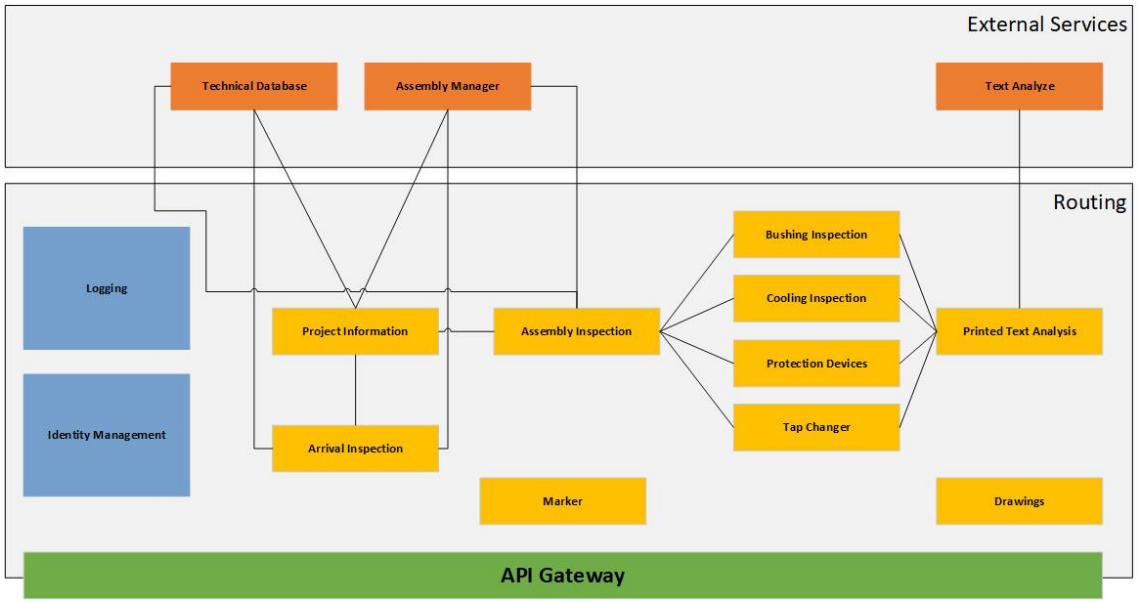

Figure 3: Architectural Overview of CaRE

4.1. External Services

The Microservice Architecture connects to three external services:

- Text Analysis: For text analysis, the OCR Service from Azure is used. An external service has been used rather than a local one because of the concept's scope. This external service can also be made internal with an appropriate OCR service like
    - Tesseract (https://github.com/tesseract-ocr/tesseract)
    - OCRopus (https://github.com/ocropus/ocropy)
    - Kraken (http://kraken.re/)

- Technical Database: The Technical Database is an internal service that holds all the technical information about the singleton
- Assembly Manager: The Assembly Manager is an internal service that hold all the assembly information of the singleton

With the MSA, further external services can also be added to the architecture. Even if a service is replaced by a different one, the interface to the frontend might be the same due to the API Gateway.

4.2. Internal Services

For the proof of concept, each business capability (or commissioning process) has been developed as its own service. Each of those Microservices has been implemented as a standalone service with its own database. All services have been developed in .NET Core and can thus run in their own Docker Container. Thus, the following services have been implemented:

- Project Information: Holds all the information about a single project. Basic Information about Projects is gathered by the external Services “Technical Database” and “Assembly Manager”.
- Arrival Inspection: With Arrival Inspection, the process of Arrival Inspection is reproduced. All information regarding Arrival Inspection is collected and provided. Therefore, the Microservice connects to the external services “Technical Database” and “Assembly Manager”.
- Assembly Inspection: With Assembly Inspection, the process during is reproduced. Therefore, the Assembly Inspection gets design and structure information from two external services (Technical Database and Assembly Manager)and sends the respective data to the respective sub-services.
- Printed Text Analyze: In some cases, long serial numbers need to be inserted to track them. From a usability perspective, this is not ideal. Hence, a service has been developed which does OCR on the images which have been sent. This service depends on an external OCR Service in Azure.
- Drawings: Since many drawings are necessary for commissioning, its own service for maintaining and storing those files have been developed. For now, this service is not connected to any other external services, but it would be possible to connect it to a company-internal drawing repository.
- Marker: In some cases, annotations are done in pictures or changes need to be marked in drawings. This information also needs to be stored persistently.

4.3. Auxiliary Systems and Services

The following auxiliary systems and services were used for different purposes:

4.3.1. Logging

For logging purposes, a central logging system with the Elastic Stack was created. For the proof of concept Serilog (https://serilog.net/) was used in each of the services for Logging. In contrast to Logging Frameworks like log4j, NLOG, log4net and others, Serilog can store log files in a structured way. This means that also objects – in the form of a JSON String – can be logged. With sinks extensions, it is possible to store data on the file system or send it to central repositories.

4.3.2. Identity Management

In terms of Microservice Security as a basis, the IdentityServer4 was used, which is an open-source OpenID Connect and OAuth 2.0 framework. On the database side, again, a PostgreSQL database server was used. The passwords were stored with the library bcrypt. Whenever a request to the Identity Service is sent, the identity service returns a token in JsonWebToken (JWT) Format. In this token, the following information – within other – is sent to the client:

- Expiration Time
- Authentication Time
- Issue Time
- Scopes
- Refresh Token

In each of the requests to the Microservices, the token has to be added to the header in terms of Authorization: Bearer < token>. If the token is no more valid or not available, each endpoint returns the HTTP Status code 401 (unauthorized request).

4.3.3. API Gateway

As API Gateway Consul.io was used and installed on an virtual machine with Ubuntu as OS. In this case, Consul acts as Service Registry and API Gateway. All above-mentioned Microservices were added. For the proof of concept only one instance of each service was started. With Consul it would be possible to start multiple instance in order to scale horizontally.

Additionally, routes were created: If a request path contains the service name, it will be forwarded to the respective service. For instance, a request at the API Gateway to http://<ip>/arrivalapi/<something> will be directly forwarded to the Arrival API.

For each Microservice two checks were implemented:

- SSH TCP Check on Port: This check simply verifies if a respective port is open and reachable. This also implies that the machine or the container, is at least, reachable.
- HTTP check on “http://<ip>/health”: Each Microservice has an endpoint that responds to its current status with some metrics regarding the service and the environment. These metrics can then be used for further monitoring. The fields are:
    - Memory consumption
    - Hostsystem
    - Machinename
    - Uptime in minutes
    - Each Disk with capacity, free space and free space in percent.

4.3.4. Documentation

The documentation of each Microservice is done with Swagger. This documentation is only available internally and not visible via the API Gateway. Nevertheless, this provides a straightforward way to test the respective endpoints during development. Furthermore Swagger provides an straight-forward way for API Versioning.

5. CONCLUSIONS

As a more and more accepted and adopted architectural style, MSA overcomes the limitations of traditional monolithic architecture. During the implementation of this proof of concept, a lot of re-factoring was necessary to meet all the desired quality attributes. First of all, the detection of the correct granularity for the Microservices proves challenging. The first approach was more or less a monolith, which was caused due to an implementation start without a clear big-picture, no strategy and no structure. Without those failures, the implementation will surely lead to Anti-Pattern and architectural smells.

After the final definition of the entire working process of the supervisors on-site, those process steps were meant to be a single Microservice. This process was identified with the user-centered design approach and through interviews with the users, later working with the system. In order to ensure proper code quality, test-driven development with NUnit was used for both the model and RESTful interface.

With all those Microservices in place, Logging into files was found not to be the best solution to debug. Therefore, the interface to the Elastic Stack was implemented. After that, it could be seen that Identity Management should be taken into account right from the beginning of the programming. The integration of Identity Management into already existing Microservices took a lot of work and re-factoring. If security considerations are integrated from scratch, the required work will be decreased significantly.

Since the CaRE System is a proof of concept, Scalability, Availability and Performance were not taken into account in terms of the Microservices. Sometimes it makes sense to implement and include already existing services. As an example, in the CaRE architecture, the TextAnalysis needs to be mentioned. If – for example – a system requires language processing, it also makes sense to use already existing services.

Additionally, the integration of the external services – Technical Database and Assembly Management – needs to be mentioned. Due to data security and restrictions within the company, access to those services, even if they are crucial for the entire application constituted a challenge. The company feared data breaches and/or data loss since some design-specific data transmitted outside the local network. As a result, only read access to well-defined endpoints and scopes was granted, which was also done with JWT Tokens.

To summarize, it needs to be mentioned that a clear understanding of the big picture is required before starting the implementation. Otherwise, a lot of re-factoring needs to be done. Furthermore, the best solution must be found for each individual project. Solutions of big players like Netflix, Facebook and so on sometimes do not suit for small applications and vice versa. Even if Microservice architecture is emerging, it is not always the best choice and needs to be investigated carefully.

5.1. Future Work

There is also room for improvement in some areas, which would also provide further assistance to users. For example, context-sensitive help with recommendations would be appreciated, where the application automatically detects desires. Such features have to be handled with care since false positives would lead to disturbances. Another topic would be auditory checklists: With this, the checklist can be spoken to a device and the system is behind automatically detects the part in the checklist. A further possibility would be the detection of emotions with an analysis of the

spoken texts. This must also be handled with care since this might be hard in a noisy environment. In terms of the user interface, some research must be done within adaptive user interfaces. Only one application needs to be developed, which adapts itself according to the current device and the available features.

REFERENCES

[1] D. Spath, O. Fanschar, S. Gerlach, M. Hämmerle, T. Krause and S. Schlund, Produktionsarbeit in der Zukunft - Industrie 4.0, Dieter Spath; Fraunhofer IAO, Stuttgart, 2013.

[2] R. Eckhoff, G. Günter and M. Mark, Bedürfnisse, Anforderungen und Trends in der Instandhaltung 4.0, 2020.

[3] P. Okeme, A. Skakun and A. Muzalevskii, Transformation of Factory to Smart Factory, 2021.

[4] N. Ilanković, A. Zelić, G. Miklós and L. Szabó, Smart factories - The product of Industry 4.0, 2020.

[5] A. Sharma, V. Dwivedi and D. Sharma, Industry 4.0 A Smart Factory: An Overview, 2020.

[6] C. Gerdenitsch, L. Deinhard, B. Kern, P. Hold and S. Egger-Lampl, Cognitive Assistance to Support Maintenance and Assembly Tasks: Results on Technology Acceptance of a Head-Mounted Device, 2021.

[7] M. Haslgrübler, B. Gollan and A. Ferscha, A Cognitive Assistance Framework for Supporting Human Workers in Industrial Tasks, 2018.

[8] G. Kanagachidambaresan, Industry 4.0 for Smart Factories, 2021.

[9] S. Grabowska, Smart Factories in the Age of Industry 4.0, 2020.

[10] J. Lewis and M. Fowler, *Microservices - A definition of this new architectural term,* 2020.

[11] T. Permikangas, *Digital Ecosystem Development: The future of integration environment,* 2020.

[12] S. Newman, Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith, O'Reilly, 2019.

[13] O. Zimmermann, M. Stocker, D. Lübke, C. Pautasso and U. Zdun, "Introduction to Microservice API Patterns (MAP)," in *Joint Post-proceedings of the First and Second International Conference on Microservices (Microservices 2017/2019)*, Dagstuhl, 2020.

[14] Talend.com, *Microservices vs. SOA: Was ist der Unterschied?,* 2020.

[15] P. Katkoori, *Application Architecture: Monolithic vs SOA vs Microservices,* 2019.

[16] J. Nemer, *Advantages and Disadvantages of Microservices Architecture,* 2019.

[17] P. Wittmer, *Microservice Disadvantages and Advantages,* 2019.

[18] W. Eberhard, Das Microservices-Praxisbuch: Grundlagen, Konzepte und Rezepte, dpunkt.verlag, 2018.

[19] T. Schirgi and E. Brenner, "Quality Assurance for Microservice Architectures," in *Proceedings of 2021 12th IEEE International Conference on Software Engineering and Service Sciences*, 2019.

[20] I. Pigazzini, F. A. Fontana, V. Lenarduzzi and D. Taibi, "Towards Microservice Smells Detection," in *Proceedings of the 3rd International Conference on Technical Debt*, New York, NY, USA, 2020.

[21] R. Tighilt, M. Abdellatif, N. Moha, H. Mili, G. El-Boussaidi, J. Privat and Y.-G. Guéhéneuc, "On the Study of Microservices Antipatterns: a Catalog Proposal," 2020.

[22] T. Schirgi and E. Brenner, "CARE – A Framework for a Multimedia Assistance System for Singletons," in *ITCA*, Toronto, 2021.

[23] *LTE Coverage June 2021,* 2021.

[24] B. Kirchhoff, S. Wischniewski and L. Adolph, Head-Mounted Displays – Arbeitshilfen der Zukunft Bedingungen für den sicheren und ergonomischen Einsatz monokularer Systeme, Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA), 2016.

[25] F. D. Crescenzio, M. Fantini, F. Persiani, L. D. Stefano, P. Azzari and S. Salti, "Augmented Reality for Aircraft Maintenance Training and Operations Support," *IEEE Computer Graphics and Applications,* vol. 31, pp. 96-101, 1 2011.

[26] J. Imtiaz, N. Koch, H. Flatt, J. Jasperneite, M. Voit and F. van de Camp, "A flexible context-aware

assistance system for industrial applications using camera based localization," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014.

[27] J. Lanier, V. Mateevitsi, K. Rathinavel, L. Shapira, J. Menke, P. Therien, J. Hudman, G. Speiginer, A. S. Won, A. Banburski, X. Benavides, J. Amores, J. P. Lurashi and W. Chang, "The RealityMashers: Augmented Reality Wide Field-of-View Optical See-Through Head Mounted Displays," in *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)(ISMARW)*, 2017.

[28] M. Rice, S. C. Chia, H. H. Tay, M. Wan, L. Li, J. Ng and J. H. Lim, "Exploring the Use of Visual Annotations in a Remote Assistance Platform," in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA, 2016.

[29] W. E. Deming, The New Economics for Industry, Government, Education, vol. 1, The MIT Press, 2000.

[30] M. Wollschlaeger, T. Sauter and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," *IEEE Industrial Electronics Magazine,* vol. 11, pp. 17-27, 2017.

[31] W. Wolf, "Cyber-physical Systems," *Computer,* vol. 42, pp. 88-89, 3 2009.

[32] Gartner, *Gartner Forecasts Worldwide Information Security Spending to Exceed 124 Billion in 2019,* 2018.

[33] NCSC, *Secure by Default - Technology which is Secure by Default has the best security it can without you even knowing it's there, or having to turn it on.,* 2018.

[34] M. Magalhaes, "Security vs Usability: Does there have to be a compromise?," *Web,* 2018.

[35] S. Penninger, S. Meier and F. Hannes, "Usability von CAPTCHA-Systemen," 3 2012.

[36] D. Taibi and V. Lenarduzzi, "On the Definition of Microservice Bad Smells," *IEEE Software,* vol. vol 35, 5 2018.

AUTHORS

Thomas Schirgi completed a MSc in information management in 2013 and worked over years in companies with singleton manufacturing. Today he is responsible for digital services and is currently working on this PhD.

Eugen Brenner is professor at the Institute of Technical Informatics at the Graz University of Technology. Numerous projects with academic and industrial partners included topics such as software development for automotive systems, communication protocols and assemblies, energy management, agile smart systems and Industry 4.0.