A HOLISTIC APPROACH FOR RAPID DEVELOPMENT OF IIoT SYSTEMS

Konstantin Merker, Tilman Klaeger, Lukas Oehm
Fraunhofer Institute for Process Engineering and Packaging (IVV)
Division Processing Technology
Dresden, Germany
konstantin.merker@ivv-dd.fraunhofer.de

February 1, 2022

ABSTRACT

While lots of research has been conducted on the architecture of Industrial Internet of Things (IIoT) systems, concepts of structuring their development processes are missing. Therefore, we propose a holistic approach supporting organizations in rapid development of IIoT systems. It includes the structuring of the development process into multiple projects sharing project conventions. Utilizing a single configurable build script for all projects, our goal is to make the integration of code from various projects into broader IIoT systems easy and the systems decomposable with minimal effort.

1 Introduction

Usually, development of information and automation systems have very different characteristics in terms of abstraction, reusability, (automated) testing and deployment. One reason is the longer lifetime of hardware in industrial equipment compared to IT systems. Changes in automation structures are complicated because of regulatory aspects and safety checks. Looking at the field of industrial automation, IIoT systems are more and more spreading out and bringing new methods of software development closer to the hardware [1]. This opens up opportunities for faster and more agile development methods.

On the software development side there are many possible ways how to structure software systems in general. During the last years, trends to compose systems based on small units, mostly known as microservices, have gained much interest [2]. As hardware becomes more efficient these technologies and aspects of development are pushing into new domains like IIoT. This creates opportunities but also challenges and things to consider for the development of such systems.

Some advantages of the new software architecture are more decoupled and better scalable systems. Following this trend, lots of new technologies have been established for containerized applications and for development operations and build automation, mostly referred to as DevOps [3]. This all leads to a new approach of developing industrial applications and is eagerly needed for integration of so called “smart products” often integrating some sort of machine learning applications.

The reusability of microservices makes it possible to develop such systems split across multiple projects and even organizations. Individual parts of the system become decomposable and integrable in multiple applications.

However, despite these possibilities we found it hard to find a structured approach for the distributed development of systems in a practical way, where a service is developed once and might be used within different projects.

So, our focus is on the decomposability of individual services, in this paper also referred to as modules. During the lifetime of a software system, some functionality might be moved between projects. Also, some projects are discontinued, but a portion of the functionality is still maintained. Being able to transfer them to active projects and dismissing the not needed parts results in a low maintenance effort. To do so, it must be possible to develop individual
modules in different projects and being able to transfer them without causing large refactorings. The same applies to libraries, tests, configurations and documentations.

In this paper we show, how it is possible to develop IIoT systems split in multiple projects using our approach and combining them in multiple, distinct IIoT systems while maintaining the possibility to develop new code rapidly and still be very flexible.

2 Related work

The related work can be categorized into general software development trends and proposed architectures for IIoT. Within the first category, current software trends like structuring software systems into small deployable units, also called microservices happen. There are approaches for migrating monoliths to microservices [4]. Microservices are shown to be useful for IoT with some limitations as well [5]. They have been applied and show advantages in flexible reusability, on-demand scalability and to speed up the creation of remote sensing products [6]. In other domains microservices have shown potential as well, for example in health applications [7]. Other research looks into the interoperability of microservice-based systems by proposing a microservice-based reference architecture and a reference implementation [8].

Another trend related to microservices is the availability of Open-Source technologies usable for microservice architectures. There is research conducted combining them in a framework [9]. Deploying microservices using the container runtime Docker for IIoT-Applications with resource-constrained devices has shown to be possible [10] [11].

Within the second category, there are framework proposals for IIoT. One of them focuses on protecting personalized privacy data for mobile crowdsensing. It includes a rational data uploading strategy where users can choose their preferred privacy level [12]. Another framework focus on the integration of several subsystems via an intelligent API layer [13]. Other authors focus on a framework utilizing blockchain technology to provide protection against data confidentiality attacks [14]. The list may be continued, but all these frameworks focus on the software architectures, but not on the development aspects of these systems.

3 Approach

The presented challenges are solved by using project conventions and a build script, also referred to as build tools, managing all aspects needed for software development and deployment. This results in projects without a difference in their kind from each other, having a similar order and thus allow functionality to get moved easily across projects.

3.1 Project setup

Each project starts with a setup phase. Common project files can be generated based on templates. The generation process is conducted via a build script and can be customized to match the project needs and provide a good starting point for development, optionally including sample modules with method stubs.

To make transferring of functionality between projects possible, each project must be capable of handling a broad range of features. A not limited list includes providing a custom library, multiple modules, automated tests, configuration files for deployment, documentation beside some other project-related files. It is possible to have projects only containing a small range of the named features for example only a library, but they are in the same way structured as projects also including more features like modules.

Different use cases also require different features, but for the IIoT domain in the context of machine learning, this set of features covers most requirements. Furthermore, it is possible to add features for projects in a similar way in the future, if required.

Setting up all project related files into one single repository may look like one project for a self-learning assistance system for machine operators (SAM, [15]) as shown in Fig. 1.

The figure shows, where certain files are typically found within the project. Configurations, libraries, modules, tests, beside some other files are located following a convention. The convention is verifiable via the build tools, so violations are detected automatically.

Fig. 2 shows another project named SAM-Machine Learning (SAM-ML). The project includes modules and libraries from SAM and provides modules implementing various machine learning algorithms to be used by other projects.
A holistic approach for rapid development of IIoT systems

Figure 1: Project related files of SAM-Project in a single source code repository.

As can be seen the project layout is very similar to the files of the previous project SAM. Therefore, functionality can be moved easily between them. This enables agile and rapid development.

3.2 Bootstrapping

Having all projects share a common layout and build tools managing them results in the challenge, how the build tools are built. One possible option is to have a separated project layout for the build tools to match its needs. This solves the problem, but it also increases the implementation and maintenance effort significantly because then two different project layouts must be managed.

Instead, establishing project conventions allowing the build tools to build themselves. On a technical level, it can be done if the build tools are obtainable as a library. The build tools project must provide the build tools as a library and at the same time use its functionality for itself. This approach can be implemented using an interpretable programming language like Python to include the library during runtime without the need for compiling it beforehand. All other projects can then list the build tools as a library dependency to have access to it.

3.3 Module development

After setting up the project layout, requirements need to get implemented. In the context of IIoT domain logic this is not done by highly skilled programmers often, but instead by experts from different domains like data scientists or automation engineers. As these are in many cases not explicitly IT specialists but domain experts with a good sense for programming, complex topics must be encapsulated and abstracted. Those topics include but are not limited to microservice handling, data models and data serialization for messaging between the services. Functionality for managing the connection to various services like databases, message brokers etc. are supported by shared libraries within the framework. Handling configuration files for specific modules like hardware configuration as well as framework-wide environment variables can be managed by framework libraries as well.

Each module can indicate deployment requirements. These can be ports, devices, databases, etc. and restart policies. For easier fault-recovery containers are typically configured with an always-restart policy. The framework also includes exception handling functionality because in modern programming languages like Python, many tasks run asynchronously.
and the built-in exception handling might not redirect exceptions properly and instead drop them. This is often the case with Python’s async library. Thus, the framework handles exceptions in a way to cause the program to restart. This is particularly the case of any issues with connectivity to hardware, services or other unexpected behavior.

It is also possible to generate modules based on templates or construct modules inheriting from an abstract base class implementing common functionality. Doing so speeds up the development.

### 3.4 Testing

A major task during development is testing. Automated tests like unit tests or integration tests play an important role for good quality software and therefore must be covered. Specific test configurations allow to start containers with broker and other services like databases for local automated testing and automated testing on a Continuous Integration Service (CI-runner). However, when interfering with local hardware like cameras, test automation might come with a high cost. Focusing on project requirements, most often, hardware interfaces are not automatically tested. A possible – but expensive – solution can be customized hardware test benches. Mocking some of this functionality can still be useful for testing other program logic.

Also writing automated tests seems not to be necessary for small projects or prototypes with a foreseen lifetime. Instead, having fast results and iterating on them is more important. To do so, starting modules locally can be used for debugging while being able to connect to a local broker in a container or a central broker on a development server. This can be done by setting environment variables. They can be set either directly in the integrated development environment (IDE) or via importing a specific "env-file" in the IDE containing all environment variables.

### 3.5 Deployment configurations

The focus is on reusability of modules across projects. Therefore, each project must be able to use other projects modules. To do so, each project has one or many deployment configurations. The configuration is in a specified format including modules and custom environment setups for the modules, if needed. Project-internal modules can be built. External ones are just included with their corresponding image name in the deployment. As a project may contain many different deployments, it is possible to setup, for example, a server with some modules and single board computers with distinct modules. Differentiating between development/debugging setups and production can be achieved by using different image tags for debugging and production respectively. Optionally, configurations may include hosts it should be deployed on.

![Figure 2: Project related files of SAM-ML-Project.](image-url)
3.6 Deployment

Building for production is usually done by an automated build pipeline on the CI-runner. Having a similar project setup, a shared build pipeline can manage the build process of all projects reducing the maintenance effort even when many projects are building at the same time.

The shared build pipeline needs to handle unit testing, compilation of software artifacts like libraries and services, cross architecture compilation for different architectures like x86 and ARM and generation of source code documentation.

The build pipeline could also handle automated deployment. However, automated deployment is hard to implement because if not done carefully, bugs can be deployed instantly on all deployments. There are concepts to prevent the outcome from being too fatal and some of them providing zero downtime, like Blue Green Deployment, Canary Deployment or Rolling Deployment [16]. In theory, these concepts can be integrated into the build pipeline. However, this has not been done yet because we do not have zero downtime as a requirement for our projects and the costs for maintaining automated deployment outweigh its benefits.

Instead, manual deployment with infrastructure as code concepts [17] is implemented. Needed files for deployment are generated via build tools from project’s deployment configurations. Utilizing configuration management systems like “Ansible”, deployment can be done in a single command for the actual roll-out if hosts are specified in the configurations. Otherwise the generated files for deployment have to be copied to the machine manually. In each case, the deployment configuration is stored as code in the repository.

3.7 Documentation

Project and source code documentation can be helpful for onboarding new project members and for internal and external communication. It is also a store of information. Tooling for documentation supports automatically generating API-documentation from source files. This allows the maintenance of source files as a single source of truth. From there most needed documentation is generated.

The documentation can be considered as a part of the project conventions. By applying this, each project has its own documentation. If modules are moved in a different project, the automated generation process of documentation files will handle it properly. It synergizes with the established concepts and enables developers and researchers moving between projects easily.

4 Findings

We employ the explained concepts for various projects in the domain of self-learning assistance systems for machine operators in the food industry [18], for a protein database, adaptive cleaning solutions and beyond and conclude, that it works very well for setting up new projects and integrating results from various projects together. Similar project layouts allow researchers and developers to move from one project to the other quickly and it is possible to transfer functionality or modules between projects easily. Typical tasks in IIoT like hardware interfacing with machine controls, still often being quite a hassle, can thus be reused and also help for a lot of brown-field applications as proposed in [19].

For an experienced developer the whole setup process from zero to a project can be done in 10 minutes starting from a fresh virtual machine. This includes the creation of the project, creating a central repository for the source code, registering the project in the build system and deploying two microservices communicating with each other locally.

Moving functionality between project works well, especially if projects are discontinued and only a portion of the functionality must be extracted.

Also, we found the maintenance of the automated build pipeline not a huge issue, as all projects share the same build pipeline so there is only one to maintain.

However, we also faced some challenges concerning versioning. Applying specified versioning schemes like semantic versioning is a must to integrate several libraries from different projects [20]. Using different versions of imported modules is not a problem as all modules are encapsulated and bringing their own libraries in their containers. But as soon as a project also provide a library, the sub-dependencies of the library have to be kind of loose or the installation of the requirements will fail if two libraries explicitly request distinct versions. Applying semantic versioning 2.0 can these issues, if strictly applied to internal and external libraries.

Another challenge has been in the development of the build tools and the framework. There is a need for customization to be compatible with required technologies and functionality. We found, that during the development of the build tools, a “paved road”/expected way is created for developers and researchers to use. Leaving the paved road may result in
significant more work to do, so it requires good decision making, which features to include in the build tools and which not. We also have not explored a built-in support for multiple programming languages to choose from which may result in further complications.

Creating the sample modules do help to quickly start a project but are critical to maintain. While it is good to look for backward compatibility when adding new functions or refactoring some parts of the code this does not always work. Whereas older modules can simply stick to an older core library, projects just generated from scratch will use the latest library. Refactoring parts of the code is easy with modern IDEs, but those will usually omit templates. If the templates are not adjusted to changes in the core library this will cause bugs in fresh modules and especially novices in the code will have a hard time to find the bugs. A possible next step would be to also test the templates using automated testing techniques.

Another pain point we found to be is the serialization mechanism. Multiple developers have different ideas how to implement it properly or whether to include certain aspects. This resulted in discussions and refactorings.

Also, we found the correct design of data types to be very important in this field. Especially data types used by many modules like sensor data must be designed very carefully. Changes in data types require at least recompilation - if not refactoring of all modules using this data type for communication. An alternative are custom data types for specific projects, but this is unfavorable because it denies the ability to transfer modules between projects easily. Multiple data types for the same entity also cause higher maintenance effort.

A minor difficulty lies within the dependency between modules in a way that sending data often causes modules to wait until another module can use the data. Otherwise it is lost. As an example, a module reads out the process image of a machine controller (PLC) and sends changed values further on. If no suitable listener is available when sending the initial process image, the data will be lost forever. This is an issue because some values don’t change often. The question is, what defines a listener to be suitable? Is a module just receiving the data suitable? Or a module storing the data? What, if in a special case, there is no need for storing the data in a deployment? This is still an open issue as there are many corner cases.

4.1 Conclusions

We presented a holistic approach to manage the development of software systems across multiple projects in the IIoT domain. We showed how it supports the development, compilation and deployment of micro services across various, distinct systems and how functionality can be moved between projects.

The proposed approach reduces effort for developers and researchers to move between projects. Additionally, the usage of templates reduces the time needed for the setup of projects and modules. The sharing of features like the build pipeline reduces the amount of work for single projects and decreases the maintenance effort. Shared libraries can evolve over the lifetime of multiple projects and thus reduce the cost for new projects further.

Even though much complexity of such systems is covered and moved to abstraction layers, learning to work with the build tools and the framework still takes some time for people new to the field. Documentation is helpful for the first steps but is not yet able to cover all complexity arising when using multiple setups with various machines, services, technologies. Understanding the infrastructure handling deployments is still not trivial and needs skills in various domains from programming to IT infrastructures.
References

[1] Y. Liao, E. de Freitas Rocha Loures, and F. Deschamps, “Industrial Internet of Things: A Systematic Literature Review and Insights,” IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4515–4525, Dec. 2018. doi: 10.1109/JIOT.2018.2834151

[2] P. Di Francesco, P. Lago, and I. Malavolta, “Migrating Towards Microservice Architectures: An Industrial Survey,” in 2018 IEEE International Conference on Software Architecture (ICSA), Apr. 2018. doi: 10.1109/ICSA.2018.00012 pp. 29–2909.

[3] L. Ferreira Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, “A Survey of DevOps Concepts and Challenges,” ACM Computing Surveys, vol. 52, pp. 1–35, Nov. 2019. doi: 10.1145/3359981

[4] L. D. Lauretis, “From Monolithic Architecture to Microservices Architecture,” in 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Oct. 2019. doi: 10.1109/ISSREW.2019.00050 pp. 93–96.

[5] D. Lu, D. Huang, A. Walenstein, and D. Medhi, “A Secure Microservice Framework for IoT,” in 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE). San Francisco, CA, USA: IEEE, Apr. 2017. doi: 10.1109/SOSE.2017.27. ISBN 978-1-5090-6320-8 pp. 9–18.

[6] B. Xiang, Z. Li, Y. Liu, and H. Zhang, “Using Microservices for Rapid Creation of Remote Sensing Products,” in 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), Apr. 2018. doi: 10.1109/ICSA-C.2018.00037 pp. 111–114.

[7] M. A. P. da Silva, V. C. Times, A. M. C. de Araújo, and P. C. da Silva, “A Microservice-Based Approach for Increasing Software Reusability in Health Applications,” in 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), Nov. 2019. doi: 10.1109/AICCSA47632.2019.9035229. ISSN 2161-5330 pp. 1–8.

[8] E. Yuan, “Architecture Interoperability and Repeatability with Microservices: An Industry Perspective,” in 2019 IEEE/ACM 2nd International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE), May 2019. doi: 10.1109/ECASE.2019.00013 pp. 26–33.

[9] C. Akasiadis, V. Pitsilis, and C. D. Spyropoulos, “A Multi-Protocol IoT Platform Based on Open-Source Frameworks,” Sensors, vol. 19, no. 19, p. 4217, Jan. 2019. doi: 10.3390/s19194217

[10] D. Jaramillo, D. V. Nguyen, and R. Smart, “Leveraging microservices architecture by using Docker technology,” in SoutheastCon 2016, Mar. 2016. doi: 10.1109/SECON.2016.7506647. ISSN 1558-058X pp. 1–5.

[11] J. Rufino, M. Alam, J. Ferreira, A. Rehman, and K. F. Tsang, “Orchestration of containerized microservices for IIoT using Docker,” in 2017 IEEE International Conference on Industrial Technology (ICIT), Mar. 2017. doi: 10.1109/ICIT.2017.7915594 pp. 1532–1536.

[12] J. Xiong, R. Ma, L. Chen, Y. Tian, Q. Li, X. Liu, and Z. Yao, “A Personalized Privacy Protection Framework for Mobile Crowdsensing in IIoT,” IEEE Transactions on Industrial Informatics, vol. 16, no. 6, pp. 4231–4241, Jun. 2020. doi: 10.1109/TII.2019.2948068

[13] O. Uviase and G. Kotonya, “IoT Architectural Framework: Connection and Integration Framework for IoT Systems,” arXiv:1803.04780 [cs], Feb. 2018. doi: 10.4204/EPTCS.264.1. [Online]. Available: http://arxiv.org/abs/1803.04780

[14] D. V. Medhane, A. K. Sangaiah, M. S. Hossain, G. Muhammad, and J. Wang, “Blockchain-Enabled Distributed Security Framework for Next-Generation IoT: An Edge Cloud and Software-Defined Network-Integrated Approach,” IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6143–6149, Jul. 2020. doi: 10.1109/JIOT.2020.2977196

[15] T. Klaeger, A. Schult, and J.-P. Majschak, “Lernfähige Bedienerassistenz für Verarbeitungsmaschinen,” Industrie 4.0 Management, vol. 33, pp. 25–28, Aug. 2017. [Online]. Available: https://www.wiso-net.de/document/BLIS_636BC679FD5A2577C6E90967B5F9B592

[16] C. K. Rudrabhatla, “Comparison of zero downtime based deployment techniques in public cloud infrastructure,” in 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Oct. 2020. doi: 10.1109/I-SMAC49090.2020.9243605 pp. 1082–1086.

[17] K. Morris, Infrastructure as Code, 2nd ed. Sebastopol, CA: O’Reilly, Jun. 2016. ISBN 978-1-09-810303-3

[18] J. Rahm, M. Graube, R. Müller, T. Klaeger, L. Schegner, A. Schult, R. Bönsel, S. Carsch, L. Oehm, and L. Urbas, “KoMMDia: Dialogue-Driven Assistance System for Fault Diagnosis and Correction in Cyber-Physical Production Systems,” in 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), vol. 1, Sep. 2018. doi: 10.1109/ETFA.2018.8502615 pp. 999–1006.
A holistic approach for rapid development of IIoT systems

[19] T. Klaeger, S. Gottschall, and L. Oehm, “Data Science on Industrial Data - Today’s Challenges in Brown Field Applications,” Challenges, vol. 12, no. 1, p. 2, Jun. 2021. doi: 10.3390/challe12010002

[20] P. Lam, J. Dietrich, and D. J. Pearce, “Putting the semantics into semantic versioning,” in Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, ser. Onward! 2020. New York, NY, USA: Association for Computing Machinery, Nov. 2020. doi: 10.1145/3426428.3426922. ISBN 978-1-4503-8178-9 pp. 157–179.

Dipl.-Wirt.-Inf. (FH) Konstantin Merker studied business informatics at the HTW Dresden. He is working in the team data science at Fraunhofer IVV since 2019. One focus of his research is on software architectures and development of (IIoT) applications.

Dipl.-Ing. Tilman Klaeger studied mechatronics at the Technische Universität Dresden and started working at Fraunhofer IVV in 2016 and is now in the lead of the team data science. His major topic in research is machine learning on industrial data collected from packaging machines and processes.

Dr.-Ing. Lukas Oehm graduated in mechanical engineering in 2010 and received his PhD in 2017 from Technische Universität Dresden entitled »Joining of Polymeric Packaging Materials with High Intensity Focused Ultrasound«. He has been working as research assistant at Fraunhofer Institute for Process Engineering and Packaging IVV since May 2017. Since October 2018, he is group leader for Digitization and Assistance Systems. His research interests are in the field of product safety issues and process efficiency in food production as well as assistance systems for machine operators.