Continuously evaluated research projects in collaborative decoupled environments

Oliver Schmidts
schmidts@fh-aachen.de
FH Aachen - University of Applied Sciences
Medical Engineering and Technomathematics
Jülich, Germany

Bodo Kraft
kraft@fh-aachen.de
FH Aachen - University of Applied Sciences
Medical Engineering and Technomathematics
Jülich, Germany

Marc Schreiber
marc.schreiber@fh-aachen.de
FH Aachen - University of Applied Sciences
Medical Engineering and Technomathematics
Jülich, Germany

Albert Zündorf
zuendorf@uni-kassel.de
University of Kassel
Software Engineering Research Group
Kassel, Germany

ABSTRACT

Often, research results from collaboration projects are not transferred into productive environments even though approaches are proven to work in demonstration prototypes. These demonstration prototypes are usually too fragile and error-prone to be transferred easily into productive environments. A lot of additional work is required.

Inspired by the idea of an incremental delivery process, we introduce an architecture pattern, which combines the approach of Metrics Driven Research Collaboration with microservices for the ease of integration. It enables keeping track of project goals over the course of the collaboration while every party may focus on their expert skills: researchers may focus on complex algorithms, practitioners may focus on their business goals.

Through the simplified integration (intermediate) research results can be introduced into a productive environment which enables getting an early user feedback and allows for the early evaluation of different approaches. The practitioners' business model benefits throughout the full project duration.

CCS CONCEPTS

• **Software and its engineering → Programming teams**; *Software prototyping*; *Empirical software validation*;

KEYWORDS

Research Best Practices, Research Collaboration Management, Metrics, Lean Software Development, Software Architecture

SER&IP'18, May 29, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5744-9/18/05…$15.00
https://doi.org/10.1145/3195546.3195549

ACM Reference Format:
Oliver Schmidts, Bodo Kraft, Marc Schreiber, and Albert Zündorf. 2018. Continuously evaluated research projects in collaborative decoupled environments. In *SER&IP'18: SER&IP'18:IEEE/ACM 5th International Workshop on Software Engineering Research and Industrial Practice, May 29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3195546.3195549

1 INTRODUCTION

Collaboration projects between researchers and practitioners offer many opportunities. Practitioners face hard to solve problems while trying to reach their business goals while researchers are searching for theoretically rewarding challenges. However, these collaboration projects have to overcome a lot of obstacles. Common ones are different priorities, misunderstandings and various understandings of project goals. While researchers often focus on difficult problems from a theoretical point of view, practitioners rather focus on cheap, quick and easy solutions for their business challenges. **Me**trics **Dri**ven Rese**a**rch Collabora**tion** (MEDIATION) [19] is an approach to unify researchers' and practitioners' understanding of project goals through measurable metrics. This approach is based on an incremental-iterative project procedure and includes the automatic continuous monitoring of common project goals formulated as a high level business goal. Therefore, everyone is able to observe any progress made towards the defined business goals.

All metrics are provided as a set of tests which enables Acceptance Test Driven Development (ATDD) to be applied. Ideally, each iteration converges at least one step further on the desired results. Nevertheless, MEDIATION does not prevent the problem of transferring research results into production. The collaboration of researchers and practitioners is heavily focused on delivering proof of concept prototypes to show intermediate project results. Although MEDIATION ensures that all parties work towards the same goals, it is not necessarily guaranteed, that demonstration prototypes can be transferred into production as a Minimum Viable Product (MVP). In this case no additional business value is gained for a company through the collaboration project.

Preparing a demonstration setup for a collaborative project is sometimes bound to a tremendous time overhead. However, prototypes are fragile and there is no assurance that a setup will be

functional during the demonstration due to stability issues and complex configuration management.

As a result, prototypes may need a lot of debugging, refactoring and testing until they can be introduced to productive systems. Therefore, the cost to benefit relation of demonstration setups and prototypes is an economical disaster. This characteristic leads to frustration on all sides.

In this paper we extend MEDIATION. Based on the existing approach, that ensures every party has the same view of a project, we now introduce a microservice software architecture setup to tackle the prototyping challenge as well as promote the "right tool for the right job", while still monitoring the project progress continuously.

The remainder of this paper is structured as follows: Section 2 briefly introduces into our application domain and offers a detailed view of collaborative aspects our approach covers and related work. In Section 3 we present our micro-service based approach for collaborative research projects in decoupled environments which is evaluated within the SmartLAB-project (c. f. Section 4). Section 5 concludes the paper.

2 APPLICATION DOMAIN, PROJECT CHALLENGES AND RELATED WORK

Collaborative research projects are challenging in technical and social aspects. This section provides an overview of our application domain motivating the use of a decoupled environment.

2.1 Application domain

The research collaboration which led to our approach is settled in the fields of life-science and biology eCommerce. The overall project goal aims to automate our partners Extract, Transform, and Load (ETL) processes. ETL processes are used to homogenize data from multiple input sources into a target system (e. g. data warehouse, database, etc.). In early stages of ETL, existing or incoming data has to be analyzed and transformed into a common data model before further steps can be taken [1, 21].

In our case data source files are provided by customers and need to be transformed into a single common data model. Every customer uses their own, inconsistent and frequently changing schema. Consequently, matching-schemata requires immense (manual) effort and automation needs to be robust and flexible to cover frequent schema changes. Nevertheless, schema-matching or ontology-matching is familiar to the scientific community, which already developed a lot of approaches to this kind of problem [2, 4, 13, 14].

Those approaches have to be adapted to the specific needs of practitioners, which increases efforts and cannot be carried out by small to medium sized companies because their day-to-day operations would be shut down by a single project. Hence, practitioners most of the time use solutions which are uncomplicated to implement and effortless to deploy.

Therefore, the production-software solely covers a small aspect of input data to be transferred automatically into a common data model. A lot of manual work remains. Algorithmic improvements have to be applied with low priority in addition to day-to-day operations which only leads to small and stepwise gains in a grown environment.

2.2 Collaborative aspects

In general, practitioners often are open minded towards algorithmic or architectural improvements. However, due to their lack of time and algorithmic complexity, enhancements often are not implemented. Things are different, if a proof of concept already demonstrated the economic value of a concept or an algorithm, substantiated by Key Performance Indicators (KPIs). Practitioners may then afford taking the time to implement these enhancements.

Nevertheless, they do not want to focus on algorithmic and implementation details if it is avoidable, just as researchers do not want to become domain experts to solve a specific problem. For both parties, value can be gained in collaborative projects if researchers focus on algorithmic details, while practitioners provide domain experts and real business data.

Additionally, researchers and practitioners both prefer their own technical environment like a familiar programming language and tooling. Each person is an expert in their own preferred tool-chain. Furthermore, forcing any side into an unknown environment lowers productivity substantially [12].

Everyone who has participated in such a project knows the occurring prototyping issues. In order to show progress development, prototypes are built though they only function in a specific configuration and with specific data. Therefore, proof of concepts often cannot be integrated (easily) into real environments and follow-up projects have high investment cost. Although the key challenge is already solved, there has to be a lot of coding and integration work to get the results into production. By using a single integration point for such a collaboration, practitioners do not need to worry about maintenance and specialized employees. Algorithms developed by researchers should be usable as a black box by only providing domain data.

Through the application of MEDIATION [19] we ensure that every project member is pursuing the same goals. The MEDIATION life-cycle in Figure 1 illustrates the process of defining and adjusting metrics. First of all, a metric with an initial threshold needs to be defined. Afterwards, the software is run and evaluated against the defined metrics with an ATDD approach, monitored and visualized through a Continuous Integration (CI) pipeline. For further reading we refer to the detailed description of MEDIATION by Schreiber et al. Due to applying MEDIATION our approach includes collaborative aspects as instant feedback by measurable metrics and ensures a higher acceptance rate of cooperation results by users among an iterative-incremental project procedure.

2.3 Related work

Software applications resulting from collaboration projects are to be integrated into a monolithic legacy application or are a new monolithic application themselves due to Conway's Law [10]. Thereby, the monolithic application consists of a single deployment unit that has several responsibilities and dependencies [17]. This approach works well for small teams and projects [11]. However, collaboration projects involve wider teams. Especially when the developed software shall be taken into production.

Monolithic software applications suffer numerous issues as

- the difficulty to maintain and evolve [5],
- the "dependency hell" [15],

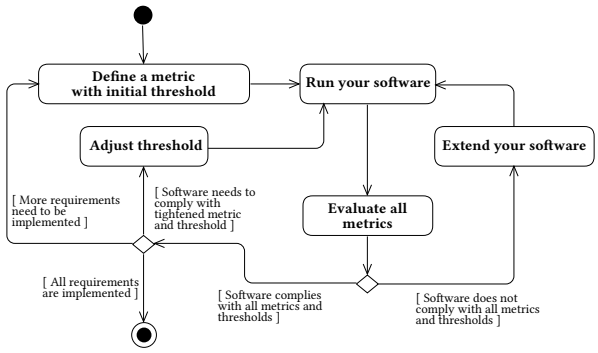

Figure 1: MEDIATION life-cycle of metric definition and continuous refinement [19]

- difficult deployment due to conflicting requirements [5],
- limited scalability [5] and
- "monoliths also represent a technology lock-in for developers, which are bound to use the same language and frameworks of the original application" [5].

Such issues are impediments for collaboration projects and taking the results into production. Additionally, this circumstance compromises approaches for shorter time-to-market strategies which Scrum or Kanban should embrace [20]. Moreover, these issues impede strategies for successful collaboration projects described by Fraser and Mancl [7].

Microservice architectures have been introduced to solve these issues. Therefore, an application should be developed as a service suite, where every service running uses its own resources and centralized management is reduced to a bare minimum [6].

Concepts and characteristics of a microservices are [5, 11]:

- focus on one dedication,
- loose coupling,
- programming language neutrality and
- bounded context.

Containerization is a common concept in the context of microservices. Cito and Gall [3] demonstrated how this concept can be applied to improve reproducibility in software engineering research projects. Nevertheless, the decision for a microservice architecture involves several challenges such as failure isolation and requirements for automation, testing and scalability. The MEDIATION approach requires automation and testing as well. Rahman and Gao have demonstrated that ATDD and Behaviour Driven Development (BDD) can be applied to microservice architectures[16]. For that reason, microservice architectures as a whole are suitable for MEDIATION. Additionally, every service itself can be developed with MEDIATION.

3 SOLUTION APPROACH BASED ON A MICROSERVICE ARCHITECTURE

This section is dedicated to introducing our Collaborative Microservice MEDIATION Pattern (CMMP). Through using the MEDIATION [19] approach project goals and progress are well defined and continuously measured. This approach encourages transparency as claimed by lean product development [18]. With the addition of microservices we provide a framework which allows us to use the advantages of MEDIATION and adds a simplified integration process into productive environments. Furthermore, CMMP provides user feedback in an early stage of the project.

In order to introduce CMMP, we first present how collaboration projects usually work and how MEDIATION can be applied (c. f. 3.1). Building on these patterns, we introduce a microservice based MEDIATION approach to minimize possibly occurring issues. Afterwards we provide a reference architecture for CMMP (c. f. 3.3) with some implementation details from our technology stack in Section 4.

3.1 Collaborative development pattern

In classic collaborative projects researchers and practitioners first need to reach an agreement on the technology stack to use for the project duration. As depicted in the upper part of Figure 2 one party knows how to use the technology stack, while the chosen stack might be far away for the other project partner. This approach causes an imbalance between collaboration partners: the partner whose technology stack is used, can to be productive right from the beginning, while the other one needs to learn a framework first.

Additionally, the development of new features within unknown legacy software is accompanied by a flat learning curve until either side can handle the other one's software. During the period of learning project progress is slow and delayed. The introduction of MEDIATION to monitor the project progress and follow the same goals also adds complexity through new acceptance tests and CI pipelines. This causes a long ramp up stage.

Furthermore, the work with legacy software in a research project might mislead a developer to introduce new features for prototypes under pressure of time thus neglecting software quality, which causes fragile, error-prone prototypes. In order to push research results into production after a successful demonstration a lot of additional work is required, since a research project should be developed in an exclusive branch. Before results may be merged into a productive code branch, refactoring, testing and quality assurance is needed. As a result of all these issues many developed features are dismissed.

One way to avoid these issues is relying on common standards and conventions. Therefore, we introduce a web service as a collaboration layer with standardized interfaces (c. f. lower part of Figure 2) for a research project, which shortens the familiarization time needed. Through commonly used frameworks like Spring Boot[1] or Django[2], ramp up time to create a web service is minimized. Additionally, the research project is starting from scratch without legacy code, which simplifies applying MEDIATION through using an exclusive repository and CI pipeline for the project. An advantage of this approach is, that the researcher can apply their technology stack while the practitioner is able to integrate (intermediate) results into production through REST or similar web service interfaces over the course of the collaboration. This possibility does not exist in the monolithic approach. Furthermore, the introduction

[1] http://spring.io/ (Java)
[2] https://www.djangoproject.com/ (Python)

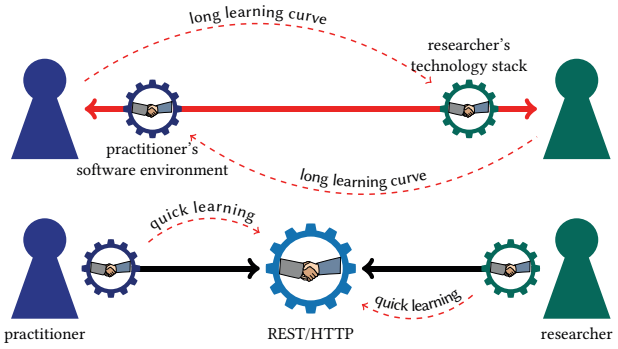

Figure 2: Improved collaboration pattern based on common web service standards for instance Representational State Transfer (REST) with Hypertext Transfer Protocol (HTTP)

of a commonly known interface reduces possibilities of misleading communication.

However, MEDIATION as introduced by Schreiber et al. [19] does not protect a software engineering project ending up as monolithic software. If developers are using a single branch for everything, copy-pasting, adding new approaches without refactoring etc chances are high that the project result is going to be a monolith. Additionally, not every approach implemented needs to be available for production after the projects end. Approaches evolve over time and some are dismissed throughout the project, but were important as intermediate steps leading to higher KPIs at some point of the project. A monolithic software will still contain these approaches if they are not removed, which causes additional work for practitioners, who need to run the service after the project duration. The integration of such a monolithic research results into productive environments is obstructive and slow.

In Figure 3 classic MEDIATION, which ended in a monolithic software, is illustrated. The given ATDD scenarios (*Input*) are executed against the monolithic evolved research service which generates output data. This *Output* is compared to the optimal output ($Output_{gold}$). With a monolithic approach the system's behavior is solely measurable as a whole. Development teams may only get feedback on system metrics. Hence, they may not feel directly responsible for metric compliance results. Additionally, tests might get more complex as the software evolves.

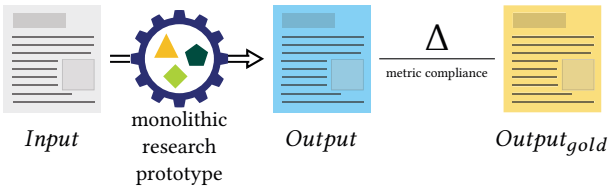

Figure 3: MEDIATION approach for monolithic applications

Nevertheless, researchers and practitioners should focus on collaboration and MEDIATION to target project goals. To tackle the identified issues of developing a monolithic web service over the course of a project, we derived the following requirements for a successful collaboration software architecture approach:

- Every project partner should be able to focus on their own implementations and deploy independently.
- Code sharing should be avoided whenever possible.
- Technical integration is needed in every iteration, so it should be easy.
- High level project metrics and collaboration should be optimized continuously on the basis of MEDIATION.
- Developed web services should be exchangeable during runtime without changing web service interfaces.
- For the project duration the concept of "you build it, you run it" [9] should be applied.

A software architecture approach which handles these issues is a microservice. Microservices provide a loose coupling with high cohesion. Additionally, they enable meeting the defined requirements. These properties make them our primary choice to tackle collaboration issues.

3.2 Microservice extended MEDIATION

A monolithic research system can be divided into multiple microservices, which offer several advantages over monoliths to address the issues mentioned earlier. Every research microservice should contain a single algorithm or approach. Applying MEDIATION to a single microservice is flawless. A microservice has its own repository with a unique CI pipeline, which enables MEDIATION in general. Additionally, MEDIATION implies the usage of an ATDD framework, as well as monitoring and visualization possibilities. These necessary parts of MEDIATION have to be added to the microservice, which is developed MEDIATION driven. The implementation is straight forward and works similarly to the monolithic approach with the exception that a single microservice has a smaller code base.

Through implementing every approach as a microservice with previously introduced interfaces, each approach serves the principles of loose coupling and high cohesion. If multiple approaches are implemented MEDIATION driven, they should be evaluated using the same metrics and tests. This setup allows comparing different academic approaches to the same problem and evaluating which approach is promising for the application domain. This procedure may raise managers' acceptance of time invested by practitioners due to the KPI visualization of MEDIATION which enables them to recognize progress.

Figure 4 illustrates how the same *Input* is given to different microservices. These microservices process this input producing an individual output. Each output is then compared to the defined $Output_{gold}$. Afterwards metric compliance results are compared and evaluated. In collaborative research problems, algorithms are adapted to a specific application domain. However, often there is not a single best performing algorithm for all domain specific boundary conditions. Nevertheless, in the setup of Figure 4 each microservice has its own MEDIATION pipeline within its CI pipeline although the given *Input* and $Output_{gold}$ are the identical. Those duplications cause overhead which should be avoided.

For this reason, we introduce a microservice MEDIATION setup, where acceptance tests are stored in an individual repository and

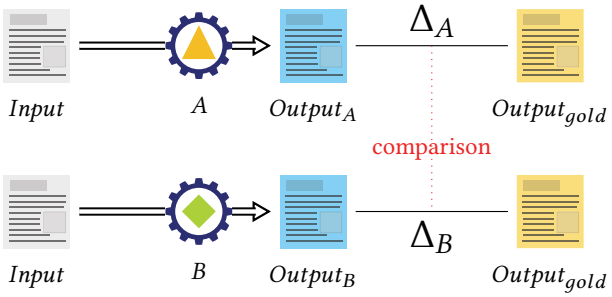

Figure 4: MEDIATION approach for microservice based applications. Pipelines are individual with identical *Input* and $Output_{gold}$

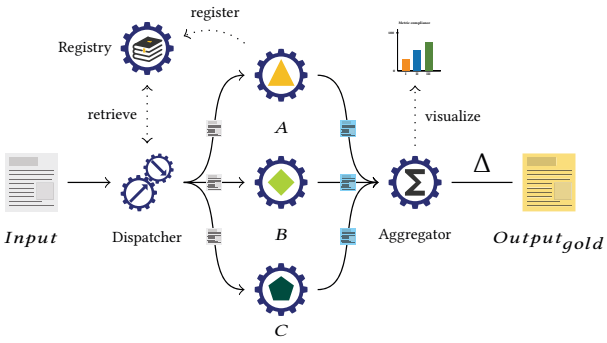

Figure 5: MEDIATION applied for comparing multiple algorithms with identical input documents

CI pipelines are separated from MEDIATION pipelines. In order to separate these two pipelines additional microservices are needed:

- a dispatcher service, that routes test data to all corresponding microservices,
- an aggregator service, that combines processed output data and evaluates the metric compliance for every service and
- a registry service, where every microservice can register to receive MEDIATION test data.

Those architecture changes are depicted in Figure 5. Every microservice, where MEDIATION should be applied, needs to register to the registry service. After the MEDIATION pipeline is started, input data is sent to the dispatcher service which retrieves all registered approaches from the registry. The dispatcher then forwards the documents. Subsequently the documents are processed by each microservice. Afterwards results are aggregated, evaluated and visualized by the aggregation service.

The main advantage of the introduced architecture is developing additional approaches or algorithms. Through a little overhead for implementing interfaces to register and communicate with the Dispatcher and Aggregator a new algorithm can be developed. Afterwards the new algorithm can be compared to existing ones with identical input data. Microservices, which implement those interfaces, are referred to as MEDIATION microservices (MMSs). Additionally, the application of MMSs leads automated evaluation methods for academic publications. As a result, multiple approaches can be compared and evaluated for the given project domain.

3.3 The Collaborative Microservice MEDIATION Pattern (CMMP)

The Collaborative Microservice MEDIATION Pattern (CMMP) aims to combine a MEDIATION microservice (MMS) with simple integration of (intermediate) research results and to transfer it into other environments during the project. Additionally, this approach simplifies handing over developed approaches to practitioners after the project.

After approaches achieve a defined metric compliance threshold, they are considered to be the Minimum Viable Product (MVP). Those MVPs can be used as a basis for demonstration prototypes during the project. However, an MVP is able to gain higher KPIs for practitioners. If an algorithm is embedded into a MMS, introduction into productive systems is made flawless.

As depicted in Figure 6 an external system needs to make a request through common interfaces (e. g. REST) to send input data to a dispatcher service, which acts as a layer between any registered algorithm and the requesting system. Thereby, algorithms are usable as black boxes for practitioners. Embedded algorithms are solely connected to the dispatcher service. In Figure 6 the dispatcher service forwards incoming data to algorithm *C* which processes the data and sends back the corresponding output. This output is returned back to the requesting system.

This approach simplifies the integration of new algorithms for researchers and practitioners. Practitioners only need to implement a request for a commonly known interface while researchers provide an interface which is available for the dispatcher. Through providing the same interface for the dispatcher, microservice embedded algorithms are exchangeable during runtime. On a service level, this approach is comparable to the strategy pattern introduced by Gamma et al. [8].

The dispatcher can be adapted to the needs of the collaboration project. In an early stage no dispatcher will be needed. Later in the process the dispatcher might get more complex. An example would be the coupling of the dispatching process to MEDIATION metric compliance: if multiple algorithms are available to the dispatcher, the MMS with the highest metric compliance could be chosen as forwarding destination. This approach makes demonstration setups painless. Since research results may be integrated into productive environments at every stage of the project, there is no need for an additional demonstration setup. Furthermore, users get in touch with research results earlier. Utilizing their experiences, comments and ideas may lead to improved project goals, since managers and users do not have the same focus.

After project end the best performing approach can be taken and handed to practitioners. Due to containerization concepts, software environments and research results are reproducible [3]. Therefore, researchers need to provide their infrastructure as code (e. g. a Dockerfile[3]). During the project we propose that MMSs with embedded algorithms are run by researchers based on the concept

[3]https://www.docker.com/

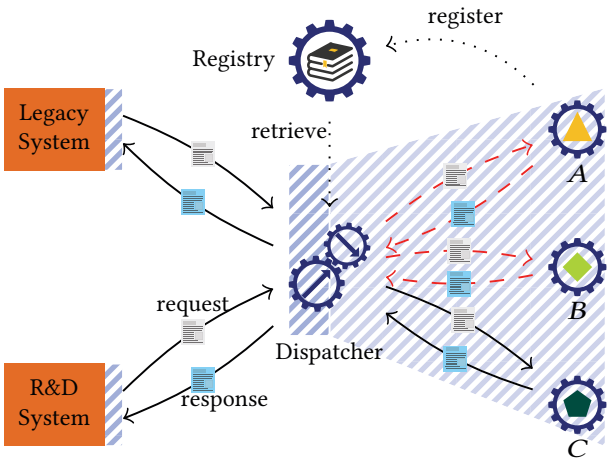

Figure 6: Architectural view of developed microservice

of "you build it, you run it". Additionally, researchers should run the dispatcher service as well, due to the reason that they need to provide multiple approaches connected to the dispatcher while practitioners are using the dispatcher as a black box for the most part.

4 APPLICATION AND EVALUATION WITHIN SMARTLAB

We developed our extension of MEDIATION in a previous research project of the "SmartLAB"[4] network. Our SmartLAB collaboration includes multiple partners: two medium sized companies and one research institute. During the previous project, we already applied MEDIATION to monitor our project progress. However, we invested a lot of time in demonstration prototypes, which were dismissed afterwards due to fragility reasons. This circumstance lead to significant overhead for integrating research results into a productive environment. This section is dedicated to the evaluation of our approach within this cooperation.

4.1 Collaboration scenario

The collaboration project arose from the need of an optimized and highly automated ETL process by an eCommerce company in the field of biology and life-science. The project serves the following purposes:

- SmartLAB automates the established manual work for homogenizing incoming data into a common data model. The company needs to match incoming data schemata into their own schema to further analyze the data and to provide condensed information about products.
- The software architecture consists of decomposable services, which enables a microservice architecture in general. Therefore, developed software shall be provided as a standalone service, which can be addressed through interfaces.
- The ETL process is manual to a high degree aside from unifying schemes into a common model. This process needs to be automated as much as possible.

In the first stage of SmartLAB we focus on homogenizing incoming data into a common data model. Several approaches are proven to work for this problem [2, 4, 13, 14]. However, it is not clear which strategy performs best in the given application domain.

Additionally, research results or intermediate steps in this topic need to be deployed to production to assist the company's employees in their day-to-day businesses of data integration. Therefore, we decided to work in an agile model, make use of MEDIATION and provide an extension for a fast productive usage if defined requirements are met.

4.2 Application

According to MEDIATION we defined concrete goals for our application. Since we focus on schema matching one of these goals is reaching $O_{gold} \geq 95\%$ detection rate with an initial threshold of a $O_t \geq 30\%$ for the MVP.

To find proper schema matching strategies applicable to our project domain, we implemented different strategies and approaches over the course of SmartLAB. The process through the different stages is shown in Figure 7. In the first stage we developed a simple database with known synonyms for different data models to fit the schema. This simple approach already reached the threshold of 30% correct detection rate. Then, we provided a microservice containing this dictionary which could afterwards be integrated by a REST call. By using this interface, schema matching is already simplified for the company's employees. Thus, a business value was gained right from the start of the project.

In the second iteration we implemented another approach using string comparison metrics for determining which string pair matches best. Additionally, a simple dispatcher was introduced to switch between these two strategies and compare them with MEDIATION. Both approaches reached the defined threshold. However, they only reached a metric compliance of $< 50\%$. The integration point had to be changed to allow the switch in production between our strategies.

Introducing the dispatcher in stage two made it possible to implement a third strategy in the following third stage without the need to change any code for the eCommerce company. Solely the dispatcher needed some adjustment. This allowed us to integrate research progress in a minimal amount of time and guaranteed quality to production. Another advantage of integrating research results early is, that users get used to a higher algorithm support. Hence, the acceptance rate of assisting software increases throughout the project.

4.3 Technical Solution

To implement different approaches to schema-matching as a microservice we used a standard Java 8 stack with Maven and Spring Boot. Since we need further integration for later stages we decided to include Apache Camel[5] to our stack as well to make integration painless.

[4] Research network in the field of digital laboratories http://www.labvolution.de/en/conferences-events/themenschwerpunkte/smartlab/

[5] http://camel.apache.org/

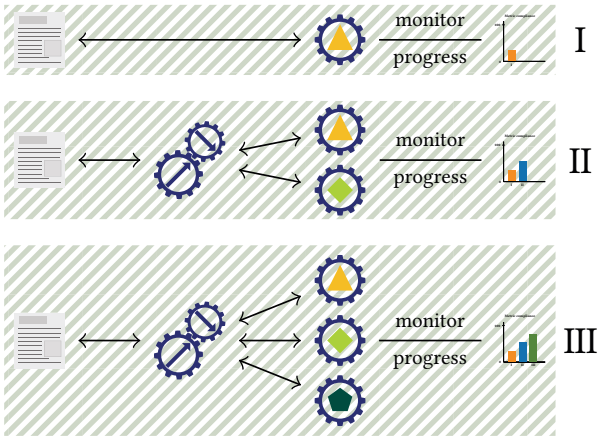

Figure 7: Consecutive stages of microservice based mediation over the course of a project

Listing 1: Simplified integration with Apache Camel

```
from("rabbitmq://localhost:5672/exchange?param=foo&param2=bar")
.routeId("synonymApproach")
.process(synonymProcessor)
.marshal(DataFormat)
.to("...");
```

Listing 1 demonstrates integration with Apache Camel routes. A route is consists of two endpoints (from and to), which Camel needs to know where a signal comes from and where the processed data should be sent to. The given route id serves debugging purposes and has no further meaning for data processing. All computation in this route is done in the synonymProcessor, which refers to the dictionary approach. This approach enables researchers to focus exclusively on algorithms implemented as processors since Apache Camel takes care of the integration process after endpoints are defined.

Additionally, Listing 1 shows that we make use of the message broker RabbitMQ (RMQ)[6]. Due to the utilization of RMQ a registry service may be left out since a service has to register an exchange or queue to RMQ to receive data. Furthermore, RMQ serves as the simple dispatcher from iteration two (c. f. Figure 7) because of the publish-subscribe functionality it offers.

After implementing the algorithms as a microservice they are containerized with Dockerfiles. This ensures that practitioners are able to run the service themselves after the project finishes [3].

4.4 Experiences with practitioners

In the beginning of the SmartLAB project we decided to evaluate the presented microservice based MEDIATION framework due to experiences in previous projects. Since we already had know-how in setting up CI pipelines, the overhead was acceptable. We evaluated our framework together with all affected groups. These were managers, developers and employees, who had to use our algorithms in production (users). The following results came up:

- Managers stated in the beginning of SmartLAB, that our approach leads to high integration efforts, if we implement algorithms as microservices with a web service interface instead of learning their framework. Through a lot persuasive efforts, they were convinced. Over the course of the project they realized introducing new approaches into production was simplified for their developers. MEDIATION aided every participant focusing on project goals and made them visible. With rising KPIs project progress was appreciated in every iteration.
- Developers stated, that using web service interfaces might cause performance issues. However, they were open to try our approach due to the simplified integration possibilities. They work with Java as well, so integration could be done in a similar way as demonstrated in Listing 1. Performance issues did not occur during SmartLAB. The most difficult part was defining interfaces which provided enough data for processing them since uploading a file of several hundred MB was not a reliable solution. Nevertheless, the reduction of information in given data sources might cause issues in other projects where algorithms depend on huge data sources all the time.
- Users were affected after the MVP threshold was reached. Dissatisfaction with the usage of the MVP in production occurred. The users had other expectations than their management for how a MVP should assist them in their day-to-day business. The initial threshold of 30% was too low to be acceptable for users because they still had to review everything via extensive manual work. Moreover, some user stated that our early approach was causing more work than they had had before. This fact shows that an MVP should be defined more carefully. Again, with further progress the acceptance increased with every iteration, which demonstrates that an early feature introduction is not bad in general.

Over the course of SmartLAB, it turned out that CMMP is almost effortless to apply for practitioners. Therefore, time-to-market for developing algorithms from the idea to their successful evaluation shrank significantly through MEDIATION combined with microservices.

5 RESUME AND LESSONS LEARNED

In this paper we provide an approach to extend MEDIATION with microservices, which provides a faster time-to-market of research results. The separation of algorithms and integration in collaboration projects between researchers and practitioners enables faster progress towards project goals because every party may use tools which are already well known to them.

Encapsulating algorithms into microservices allows integration into productive systems right from the start of the project. Nevertheless, the ramp up for a new project is high and many tools and conventions have to be evaluated. Through the proposed CMMP the additional time used for ramp up is reduced significantly. Therefore, researchers may focus on algorithms while companies are not losing the possibility of using these in a productive environment. Additionally, algorithms are developed while interacting with real data. On account of CI and the iterative approach big bang releases are avoided. Through the early productive evaluation of the MVP user feedback can be utilized for a future iteration.

[6] https://www.rabbitmq.com/

Due to the presented framework practitioners were able to transfer research results into production in a short amount of time, which enabled them to see their KPIs increasing over the project duration. This approach evaluated MEDIATION as well, since metrics were directly applicable to business value, ensuring a successful project. However, there are downsides to the presented approach. The extensive ramp up stage of MEDIATION [19] gets extended as well. It needs to be determined when it is a good time to start with the continuous collaboration. Some progress has to be made initially to show first results to the project partner. This often leads to a point where too much has been done already. Additionally, these intermediate step might be too complex to be integrated quickly. After some iterations there might be too many microservices to manage.

In conclusion, it can be said that the presented CMMP approach monitors project progress and to integrate intermediate research results prevails over the downsides for practitioners, since additional business value is gained throughout the collaboration project.

REFERENCES

[1] Alexander Albrecht and Felix Naumann. 2008. Managing ETL Processes. *NTII* 8 (2008), 12–15.

[2] Philip A. Bernstein et al. 2004. Industrial-strength Schema Matching. *SIGMOD Rec.* 33, 4 (Dec. 2004), 38–43. https://doi.org/10.1145/1041410.1041417

[3] Jürgen Cito and Harald C. Gall. 2016. Using Docker Containers to Improve Reproducibility in Software Engineering Research. In *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16)*. ACM, New York, NY, USA, 906–907. https://doi.org/10.1145/2889160.2891057

[4] Hong-Hai Do and Erhard Rahm. 2002. COMA: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment, 610–621.

[5] Nicola Dragoni et al. 2017. Microservices: Yesterday, Today, and Tomorrow. In *Present and Ulterior Software Engineering*, Manuel Mazzara and Bertrand Meyer (Eds.). Springer International Publishing, Cham, 195–216. https://doi.org/10.1007/978-3-319-67425-4_12 DOI: 10.1007/978-3-319-67425-4_12.

[6] Martin Fowler and James Lewis. 2014. Microservices a definition of this new achitectural term. (2014). Retrieved 2018-01-16 from https://martinfowler.com/articles/microservices.html

[7] Steven Fraser and Dennis Mancl. 2016. Strategies for Building Successful Company-university Research Collaborations. In *Proceedings of the 3rd International Workshop on Software Engineering Research and Industrial Practice (SER&IP '16)*. ACM, New York, NY, USA, 10–15. https://doi.org/10.1145/2897022.2897025

[8] Erich Gamma et al. 1995. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[9] Jim Gray. 2006. A conversation with Werner Vogels. *ACM Queue* 4, 4 (2006), 14–22.

[10] James D. Herbsleb and Rebecca E. Grinter. 1999. Splitting the organization and integrating the code: Conway's law revisited. In *Proceedings of the 21st international conference on Software engineering*. ACM, 85–95.

[11] David Jaramillo, Duy V. Nguyen, and Robert Smart. 2016. Leveraging microservices architecture by using Docker technology. In *SoutheastCon 2016*. 1–5. https://doi.org/10.1109/SECON.2016.7506647

[12] Andrew J. Ko et al. 2011. The State of the Art in End-user Software Engineering. *ACM Comput. Surv.* 43, 3 (April 2011), 21:1–21:44. https://doi.org/10.1145/1922649.1922658

[13] Jayant Madhavan et al. 2005. Corpus-based schema matching. In *21st International Conference on Data Engineering (ICDE'05)*. 57–68. https://doi.org/10.1109/ICDE.2005.39

[14] Sabine Massmann et al. 2011. Evolution of the COMA match system. In *Proceedings of the 6th International Conference on Ontology Matching-Volume 814*. CEUR-WS. org, 49–60.

[15] Dirk Merkel. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal* 2014, 239 (2014), 2.

[16] Mazedur Rahman and Jerry Gao. 2015. A reusable automated acceptance testing architecture for microservices in behavior-driven development. In *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on*. IEEE, 321–325.

[17] Eric S. Raymond. 2003. *The art of Unix programming*. Addison-Wesley Professional.

[18] Donald G. Reinertsen. 2009. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing.

[19] Marc Schreiber, Bodo Kraft, and Albert Zündorf. 2017. Metrics Driven Research Collaboration: Focusing on Common Project Goals Continuously. In *2017 IEEE/ACM 4th International Workshop on Software Engineering Research and Industrial Practice (SER IP)*. 41–47. https://doi.org/10.1109/SER-IP.2017..6

[20] Ken Schwaber and Jeff Sutherland. 2012. *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust*. Wiley.

[21] Dimitrios Skoutas and Alkis Simitsis. 2006. Designing ETL Processes Using Semantic Web Technologies. In *Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP (DOLAP '06)*. ACM, New York, NY, USA, 67–74. https://doi.org/10.1145/1183512.1183526