A Process Model for Test Driven Development in the Big Data Domain

Daniel Staegemann[a], Matthias Volk[b], Naoum Jamous and Klaus Turowski
Magdeburg Research and Competence Cluster VLBA, Otto-von-Guericke University Magdeburg, Magdeburg, Germany

Keywords: Big Data, Test Driven Development, TDD, Process Model, Design Science Research, DSR, Microservice.

Abstract: Big data has emerged to be one of the driving factors of today's society. However, the quality assurance of the corresponding applications is still far from being mature. Therefore, further work in this field is needed. This includes the improvement of existing approaches and strategies as well as the exploration of new ones. One rather recent proposition was the application of test driven development to the implementation of big data systems. Since their quality is of critical importance to achieve good results and the application of test driven development has been found to increase the developed product's quality, this suggestion appears promising. However, there is a need for a structured approach to outline how the corresponding endeavors should be realized. Therefore, the publication at hand applies the design science research methodology to bridge this gap by proposing a process model for test driven development in the big data domain.

1 INTRODUCTION

Today's society has developed to be heavily driven by knowledge, information and technology (Levin and Mamlok 2021). Consequently, big data (BD), respectively big data analytics (BDA) have gained huge popularity among organizations that want to profit from this rather new resource. Furthermore, those who do incorporate BDA into their processes experience (on average) a significant increase in productivity (Müller et al. 2018), further justifying the positive sentiment. Yet, this only does apply to proper use, which is, however, not always a given, since it is a highly challenging endeavor (Volk et al. 2019). The arguably most common issues in this regard are a low input data quality (Abdallah et al. 2022; Staegemann et al. 2021b), human error or bias in the use of the applications, and erroneous implementations of the respective systems (Staegemann et al. 2019).

For the publication at hand, the focus is on the latter. While there have been numerous works to facilitate the testing of BD applications, it is still a rather immature topic (Staegemann et al. 2021c). Therefore, further work in this field is needed. This includes the refinement of existing approaches and strategies as well as the exploration of new ones. One rather recent proposition was the application of test driven development (TDD) to the implementation of BD systems (Staegemann et al. 2020).

When done correctly, this could solve several issues at once. Not only would the quality and flexibility of the developed applications be increased, but possibly also the trust of the users, which is crucial to assure the frequent and genuine incorporation into the decision processes (Günther et al. 2017). However, so far, there has been no structured approach formulated how the corresponding endeavors should be realized. To bridge this gap, the following research question (RQ) shall be answered:

RQ: How can the process of applying test driven development in the big data domain be structured?

To answer the RQ, the publication at hand is structured as follows. After the introduction, the background is briefly delineated. This is followed by an overview of the applied methodology. Afterwards, in the main part, a process model for TDD in the BD domain is developed, which is also this work's main contribution. Subsequently, the model is further discussed and avenues for future research are outlined. Finally, a conclusion is given.

[a] https://orcid.org/0000-0001-9957-1003
[b] https://orcid.org/0000-0002-4835-919X

Staegemann, D., Volk, M., Jamous, N. and Turowski, K.
A Process Model for Test Driven Development in the Big Data Domain.
DOI: 10.5220/0011337200003335
In *Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2022) - Volume 3: KMIS*, pages 109-118
ISBN: 978-989-758-614-9; ISSN: 2184-3228

2 BACKGROUND

To establish a solid foundation and a common understanding for the further explanations, in the following, the most important terms and concepts are briefly introduced.

2.1 Big Data

The amount of data that is being produced, captured, and analyzed as a result of today's society's digitization has been and is still rapidly growing (Dobre and Xhafa 2014; Statista 2021; Yin and Kaynak 2015). Concurrently, its complexity and the demands for its processing also increased. Consequently, the systems that were previously used for this purpose are oftentimes no longer sufficient (Chang and Grady 2019). Therefore, new tools and techniques are needed to deal with the new requirements and simultaneously the term big data emerged to describe this phenomenon. Even though the origins of a term are not conclusively clarified (Diebold 2012) and there is also no unified definition for it (Al-Mekhlal and Khwaja 2019; Volk et al. 2020b), most of the relevant literature follows a similar understanding. The arguably most influential description (Chang and Grady 2019) is based on four characteristics, which are sometimes also termed the 4 Vs of big data. Those are volume (number and/or size of data entries), velocity (speed of data ingestion and/or required processing speed), variety (diversity of data and content), and variability (changes in the other characteristics over time). Due to the widespread need for high quality decision making, BDA is used in numerous domains, such as manufacturing (Nagorny et al. 2017), management support (Staegemann et al. 2022a), fashion (Silva et al. 2019), education (Häusler et al. 2020), sports (Goes et al. 2020), agriculture (Bronson and Knezevic 2016), or healthcare (Bahri et al. 2019).

2.2 Microservices

The general idea of the microservice concept is to decompose an envisioned application into several smaller services that then interact with each other to accomplish the given task (Nadareishvili et al. 2016). Usually, the services are based on business functionality. This, in turn, allows it to benefit from a high degree of specialization. The microservices all run in their own processes and for the communication among each other, only lightweight mechanisms are utilized. Due to their independent nature, the particular services implementation can be heterogeneous (Freymann et al. 2020). This, inter alia, refers to the utilized programming languages and technology stacks. Moreover, their properties allow an independent deployment and usage. For this purpose, usually continuous deployment tools and pipelines are used, allowing for the automation of the procedure.

Even though in software engineering componentization is generally considered a good practice, achieving a high degree of modularity is often seen as challenging task (Faitelson et al. 2018). However, when using microservices, this is achieved by design. This also reduces the effort for maintenance and the implementation of modifications, since it is often sufficient to only redeploy the affected service when incorporating changes. As a result, through the use of microservices, an evolutionary design, which is driven by frequent and controlled changes, is promoted (Krylovskiy et al. 2015).

2.3 Test Driven Development

TDD is generally seen as a development approach that (for the cost of a reduced speed) is feasible to improve an implementation's quality (Staegemann et al. 2021a). The corresponding advantages are twofold. On the one hand, the test coverage is increased. This helps to detect errors (early) and prevents that they affect the productive users. On the other hand, the system's design is also influenced, since a major part of TDD is its decomposition into the smallest reasonable pieces. This reduced complexity also helps to avoid errors and increases maintainability (Crispin 2006; Shull et al. 2010). Even though the primary application area of TDD, and also the one that is relevant for the remainder of this paper, is in software development, it is also used in other contexts, such as process modelling (Slaats et al. 2018) or ontology development (Davies et al. 2019; Keet and Ławrynowicz 2016).

In the traditional software development approach, new features are at first envisioned, then implemented and finally tested. However, in TDD, this order is changed. While the first step remains the same, the identified functionality is broken down into small parts (Fucci et al. 2017). In the following, tests for those parts are written. To assure that they indeed test new aspects, they are run and should, for a lack of the actual implementation, fail (Beck 2015). If they don't, they need to be reworked due to the premise. After the tests failed, the productive coding takes place, resulting in the desired functionality. The main focus here is just to make it work. In turn, other aspects, like the elegance of the code, are not important, as long as the previously written tests are passed (Crispin 2006). If this is the case, the code is then refactored to improve the readability, its adherence to standards, best practices, and conventions and to improve its overall quality (Beck 2015). While doing so, the previously written tests are utilized as a safety net to make sure that no errors are introduced during this procedure. As mentioned earlier, this focus on incremental modifications and small tasks (Williams et al. 2003) does not only affect the coverage, but also the design of the developed solution. Moreover, developers are provided with more immediate feedback, due to the shorter test cycles (Janzen and Saiedian 2005). While unit tests are usually the backbone of TDD, they can (and should) also be amended by other types of tests, such as system, tests, or integration tests (Sangwan and Laplante 2006). Hereby, especially the latter can be seen as essential (Kum and Law 2006). Furthermore, to make sure the necessary test frequency can be achieved without the developers having to cumbersomely deal with it manually, TDD is often combined with a continuous integration (CI) pipeline to enable test automation (Karlesky et al. 2007; Shahin et al. 2017). Consequently, whenever a change is committed, a CI server runs the existing tests, checking if the last change has introduced any new errors that need to be fixed.

2.4 Test Driven Development in Big Data

As it was already described earlier, applying TDD is a promising new approach for the engineering of high-quality BD applications. For this purpose, the use of microservices as a technical foundation has been proposed (Staegemann et al. 2020). Since a major component of TDD is to break down the desired application into small parts and microservices facilitate exactly this architectural concept, there is a huge synergy that can be exploited (Shakir et al. 2021). Their use allows to realize each business functionality as a separate service, which also gives the option for independent scaling, depending on the respective workloads. Further, this also impacts the implementation process, since the development of the respective services can be distributed across different teams. Additionally, those don't have to use a homogenous toolset, but can instead rely on the technology set they deem the most suitable for the given task, due to the independence of the services from each other. In another context, TDD also increases the flexibility. The created tests allow for easier and safer changes to the developed application because they can be immediately validated through the existing tests, leading to faster feedback, the avoidance of newly introduced errors and consequently more trust by the users. However, even though the general idea of applying TDD in the BD domain seems promising and there are already some works in the domain (Staegemann et al. 2022b), to facilitate its diffusion and make its application more accessible, it is still necessary to develop further corresponding patterns, frameworks, process models, best practices, and approaches to provide developers with a solid foundation they can lean on for their projects, instead of having to determine all steps (and their order) on their own.

3 METHODOLOGY

In order to assure scientific rigor while answering the RQ, the design science research (DSR) approach (Hevner et al. 2004) is applied. This constructive methodology is geared towards the development and evaluation of artifacts in the information systems research domain. The purpose of those is to solve organizational problems. They can be "constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems)" (Hevner et al. 2004). To further enhance the comprehensibility, the workflow of the design science research methodology (DSRM) presented in (Peffers et al. 2007) is followed. The DSRM decomposes the DSR into a sequence of six steps, which are depicted in Figure 1.

The DSRM begins with the *problem identification and motivation*, which are outlined in the beginning of the next section. In the second activity, the researcher shall *define the objectives for a solution*. This will also be part of the same subsection. The third step, *design and development*, will be discussed in the succeeding subsection, resulting in the construction of the DSR artifact as the

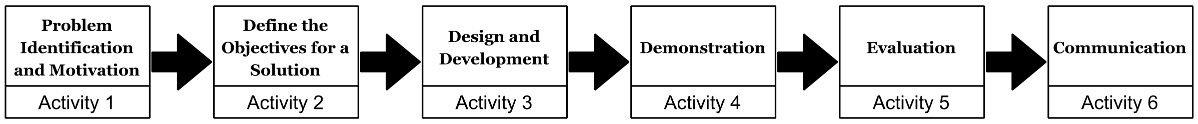

Figure 1: Process Sequence of the DSRM According to (Peffers et al. 2007).

main contribution of the publication at hand. Furthermore, the underlying explanations will serve as an implicit, preliminary *evaluation*, which corresponds to activity five. The final activity, *communication*, is performed through the publication at hand. However, due to the artifact being a process model, whose phases need to be filled with concrete activities (which is out of this work's scope) for its actual implementation, the *demonstration* will be deferred to the future.

4 THE PROCESS MODEL

In the following, using the DSRM by Peffers et al. (2007), a process model is proposed, facilitating the application of TDD in the BD domain through the provisioning of a structured approach that supports developers in implementing their respective BD endeavors in a test driven manner.

4.1 Motivation

When applying the DSRM, the first activity is to identify the problem that shall be solved, and to motivate, why this should be done. In the case at hand, it was already outlined why big data is of great significance for today's society. Further, the importance of proper quality assurance was outlined, and it was discussed how the application of TDD might help in the implementation of the corresponding systems. However, to our knowledge, an actual procedure for this has not yet been formalized. While it is necessary to maintain a certain degree of freedom to reflect the individual nature of such projects, this also constitutes both, a barrier for entry, as well as a potential source for errors and inefficiencies. Since the proposed concept for the application of microservice-based TDD in the big data domain (MBTDD-BD) contains several levels and types of tests, there is a big number of activities required for its implementation. Developers that don't have extensive experience with TDD in the BD domain might be deterred by the huge number of different possible orders of those (with wrong decisions leading to extra work or worse results), as well as the threat of overlooking important activities, which would reduce the effectiveness of the approach. Since TDD is usually more time consuming than the traditional approach (Staegemann et al. 2021a), this additional effort can only be justified if the corresponding benefits can actually be reaped. Therefore, it is necessary to provide developers with a structured procedure to reduce this uncertainty, eliminate potential sources of error and, hereby, facilitate the use of TDD in the BD domain to increase the overall quality of the developed solutions. Furthermore, this process should be easy and unambiguous to follow, which on the one hand refers to the outlined sequence of steps, but on the other hand also on the utilized notation.

4.2 Development of the Artifact

Since this work builds upon the MBTDD-BD proposition (Staegemann et al. 2020), it will also follow the general structure, which results in the existence of several levels (system, component, subcomponent/ microservice, method). Furthermore, the wording is adopted, increasing the comprehensibility. Moreover, even though in the following only tests are explicitly mentioned, as suggested in the MBTDD-BD, benchmarks can also be added alongside them to introduce another dimension of quality assurance. However, the main focus is on the functional testing.

To start the process, it is at first necessary to know the requirements for the system that shall be developed (ISO 2018; Sommerville 2007). However, in the context of this work, outlining their gathering would be out of scope. Therefore, the list of requirements is considered as an available input. Based on those, concrete features of the system can be derived. While it is not yet determined how they will be implemented, this step turns the identified needs into high level tasks and is therefore a prerequisite for the actual realization. In the TDD methodology, after determining what is to be implemented, the corresponding tests shall be written. Accordingly, the next step is to define the tests for the system as a whole. Those might be automated, manual, or a hybrid approach and are supposed to show if it provides the desired functionality. Implementing the system tests at such an early stage on the one hand corresponds with the TDD philosophy, and on the other hand potentially also brings practical advantages. This step, as the previous one, immensely benefits from having domain knowledge and a comprehensive overview of the product's business side, respectively the purpose it is developed for. Therefore, the process should heavily involve experts or potential users from that domain. Meanwhile the further steps are of rather technical nature and do not need that much comprehensive knowledge of all usage related aspects of the product. By creating the system tests early, it is possible to focus the involvement of the needed knowledge carriers on the starting phase, which allows them to focus on their day to day tasks afterwards, while the technical experts take over from then. (Even though some involvement of distinct business experts/users might still be needed for some decisions that might arise later.) Once the system tests have been created, the implementation can be progressed. For this purpose, the previously identified features are translated into distinct microservices, which inherently also determines the system's architecture. Further, not only the services and their functionality are defined, but also their interfaces. The result of this step is an overview of the required microservices as well as their interconnections. However, the concrete implementation of the services is not yet designed. In the following, those microservices, which are also called subcomponents in the MBTDD-BD, are grouped to components. A component constitutes a contentual unit that is deemed belonging together by the developers, respectively architect. Those could for example be the loading of data that consists of several services that are each specialized to provide data from one specific (type of) source or the preprocessing that comprises multiple steps that are each realized as a separate microservice. However, there are no fixed rules, instead the definition of components is subject to the individual assessment of the decision makers. Moreover, depending on the context, components can also overlap (e.g. a microservice can belong to several components), or just comprise a single subcomponent, in case it is rather standalone. Yet, for the sake of coherence, each microservice has to belong to at least one component.

Subsequently, to later on assure that not only the components itself but also the communication between them works as intended, corresponding tests have to be created. While all those steps, that happen on the system level, are only conducted once, the succeeding activities are performed repeatedly until the implementation of all components is finished. At first, is has to be chosen, which component shall be worked on next. The criteria for this decision can be individually determined. Possible reasoning could, for example, be based on factors such as the availability of certain experts, the perceived importance or complexity, or contentual relations and interdependencies. It is also possible that a specific microservice shall be implemented at this stage (for example based on above mentioned criteria) and therefore the corresponding component is chosen at this stage. After the decision is made, the system level is left and the work on the component level begins.

If the component has not yet been worked on before, the next step is to create the tests for the component, otherwise this can be skipped, since it has already been done in the past. Then it has to be determined which microservice will be implemented next. Further, in succession, there is also a change from the component level to the subcomponent level. There, analogous to the previous levels, at first, tests for the unit (in this case the microservice) as a whole are written, allowing to later on confirm that the envisioned capabilities have actually been successfully realized. When the creation of those tests is assigned to a team that is different from the one that is responsible for the implementation, this can also act as an additional safety net by adding another perspective on potential issues and edge cases. This also constitutes a deviation from the proposition expressed in the original MBTDD-BD paper (Staegemann et al. 2020), since there, the assurance of the functionality of the microservice as a whole was described as only being implemented indirectly, through the tests within the developed service. Explicit tests were not intended. However, since the inclusion of such tests for the entire service allows to incorporate a view on the slightly bigger picture, which is not necessarily given on the method level, their integration reduces the risk of overlooking issues that are not as apparent when only operating on the method level.

The creation of the tests for the microservice as a whole is followed by the test driven implementation of that service, as it is described in the related background section. Therefore, at first, the tests for a function are written, then the functionality is implemented and finally the code is refactored to increase its quality and readability. This procedure is repeated until the entire service is completed. While the described process as a whole takes place on the subcomponent level, the implementation of the particular functions corresponds to the method level. Once the implementation is finished, the aforementioned tests for the entirety of the subcomponent are run. In case that they do not pass completely, the service goes back to the previous implementation stage, where it is worked on until the issue is deemed resolved. Once the subcomponent tests pass, the subcomponent level is left, the process again enters the component level and the microservice can be integrated into the current iteration of the component.

However, this is not the final step concerning the regarded service. It is possible that a microservice in itself is not erroneous and, therefore, the testing is positive, but there are issues with the interplay with other services. An example (even though it is not big data related) that made the news was the NASA climate orbiter crash from 1999, where one involved partner used English units and the other metric ones, leading to a failed mission, despite both parts in itself being functional (NASA 2019). To avoid a similar situation, the integration of the subcomponent needs to be followed by a run of the component tests as well as the relevant tests for the communication. Only if those also pass, the microservice can be deemed finished. Otherwise, the developers have to go back to the development stage. However, in case of success, the component level is left and the system level is entered again. Now, the further procedure depends on the current status of the system's implementation. If there are still components that are not entirely finished, it has to again be decided, which component should be worked on next. From there, the process continues as already outlined above.

In case every component, and therefore every part of the envisioned system, has been implemented and individually tested with success, a final test run that comprises all tests (including those for the system as a whole) allows to check for a last time, if everything is working as intended. Should there be any problems, those have to be thoroughly analyzed. Once the source of error is identified, the developers shall fix the underlying issues, using the comprehensive test collection to assure that no new errors are introduced. However, if this last instance of quality assurance is also passed without the occurrence of any problems, the development process is finished and the system can be used productively.

The complete process model is displayed in Figure 2. To give an easy to follow overview of the proposed process model, its graphical depiction is heavily leaning onto the BPMN notation. However, this also introduces some constraints. The levels of the process are depicted as separate BPMN pools.

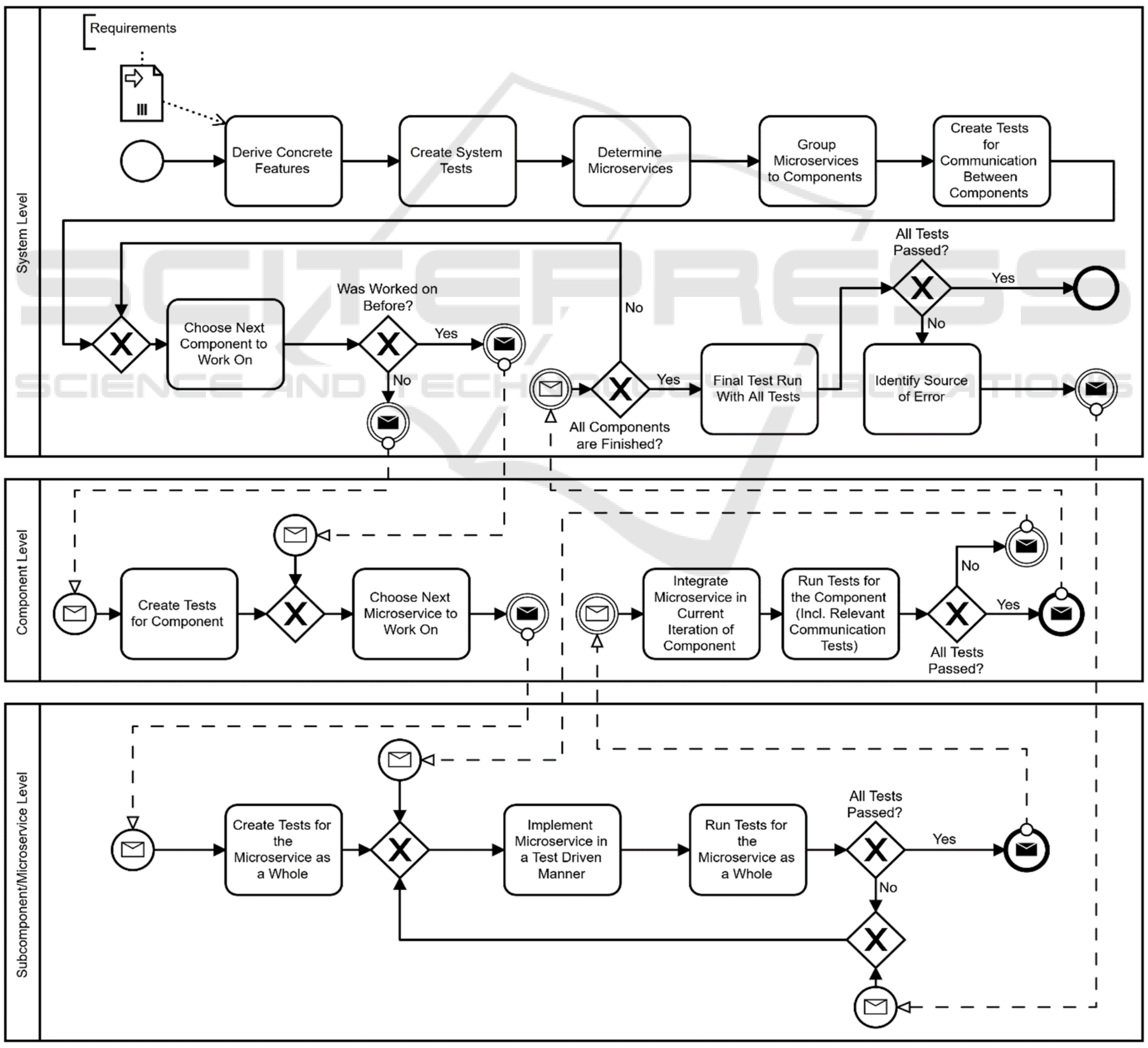
System Level
Requirements
Derive Concrete Features
Create System Tests
Determine Microservices
Group Microservices to Components
Create Tests for Communication Between Components
Choose Next Component to Work On
Was Worked on Before?
Yes
No
All Components are Finished?
Final Test Run With All Tests
All Tests Passed?
Identify Source of Error
Component Level
Create Tests for Component
Choose Next Microservice to Work On
Integrate Microservice in Current Iteration of Component
Run Tests for the Component (Incl. Relevant Communication Tests)
Subcomponent/Microservice Level
Create Tests for the Microservice as a Whole
Implement Microservice in a Test Driven Manner
Run Tests for the Microservice as a Whole

Figure 2: Process Model for Test Driven Development in the Big Data Domain.

While this slightly deviates from the idea behind the concept of pools in BPMN, it increases visual clarity and was therefore implemented. Since the test driven implementation of the microservice is depicted as one step and not further broken down, there are only three levels shown, with the method level being omitted.

Furthermore, especially in larger projects, it is likely that several teams work in parallel, whereas the depicted process presents a linear sequence. This is also for the sake of visual clarity. However, in reality, there might be several microservices (also from different components) be worked on at the same time. Yet, this does not crucially affect the actual flow, wherefore it is only mentioned but not graphically represented. Additionally, the outlined process refers to projects that are created from scratch. If an application that was built according to the proposed procedure shall be modified, the already existing tests can be utilized. Changes on any other pre-existing systems are out of scope of the proposed process model and individual approaches have to be found.

5 DISCUSSION AND FUTURE WORK

With the steady increase of the number of BD applications that are being used and their quality assurance being one of the major challenges (Staegemann et al. 2019), finding ways to tackle that issue is highly important. While the MBTDD-BD approach seems generally promising to increase the quality as well as the modifiability of the developed systems, up to now, there was no structured procedure for its application. The proposed process model is directed towards bridging this gap. By following the comprehensive sequence of steps, the necessary activities can be covered, while also assuring that the order is actually sensible and corresponds to the spirit of the TDD methodology.

However, several factors have to be taken into account. The first aspect is that the requirements for the system are taken for granted. While this makes sense for the aspired scope, they are extremely important for the success of an implementation project. Therefore, it is mandatory to find a suitable approach for their collection. This also means that the proposed process model cannot be seen as a panacea but has to be used in conjunction with other suitable methods. To a lesser degree this also applies to the test driven implementation of the distinct microservices not being described in detail. However, on this level, the development does not crucially differ from other development contexts, so that a specific description is not necessary.

Another aspect that is highly important but not directly covered by the process model is the selection of tools and technologies. While the modular nature of the MBTDD-BD allows for a high degree of flexibility and gives the developers the choice, which programming languages, frameworks or existing solutions they want to use, respectively incorporate, there is no support provided for those decisions. Since there is a plethora of available options, this task can, however, also be highly challenging. While there are already existing works that focus on a general decision support for the technology selection in BD projects (Volk et al. 2020a), additional material that is geared towards this specific situation might be helpful for prospective developers and, hence, also help to facilitate the dissemination of TDD in the BD domain in general.

Additionally, as previously mentioned, the proposed model slightly simplifies the development process by presenting it as a sequential flow. While is reality, several teams might work in parallel on several services, the increased comprehensibility was deemed worth it to accept that slight simplification as a trade-off. When applying the model in a parallel scenario, it is therefore necessary to account for this decision and adjust the actual workflow accordingly.

Further, the model only outlines which actions should be taken in which order, but not by whom. Even though the specifics of this decision obviously heavily depend on the structures of the organizations and teams that are involved, the identification of best practices and recommendations could still prove to be valuable support. Therefore, this might be a worthwhile task for future researchers that has strong practical implications.

Since the quality of big data applications heavily depends on the correct architectural choices (Ataei and Litchfield 2020) and there are numerous patterns proposed for the implementation of microservices, it also appears reasonable to regard those two aspects in context of each other to determine, which microservice patterns are best suited to deal with certain challenges of big data development and the underlying big data characteristics.

6 CONCLUSION

Big data and the corresponding tools, technologies, and applications have emerged to be one of the driving factors of today's society. Countless

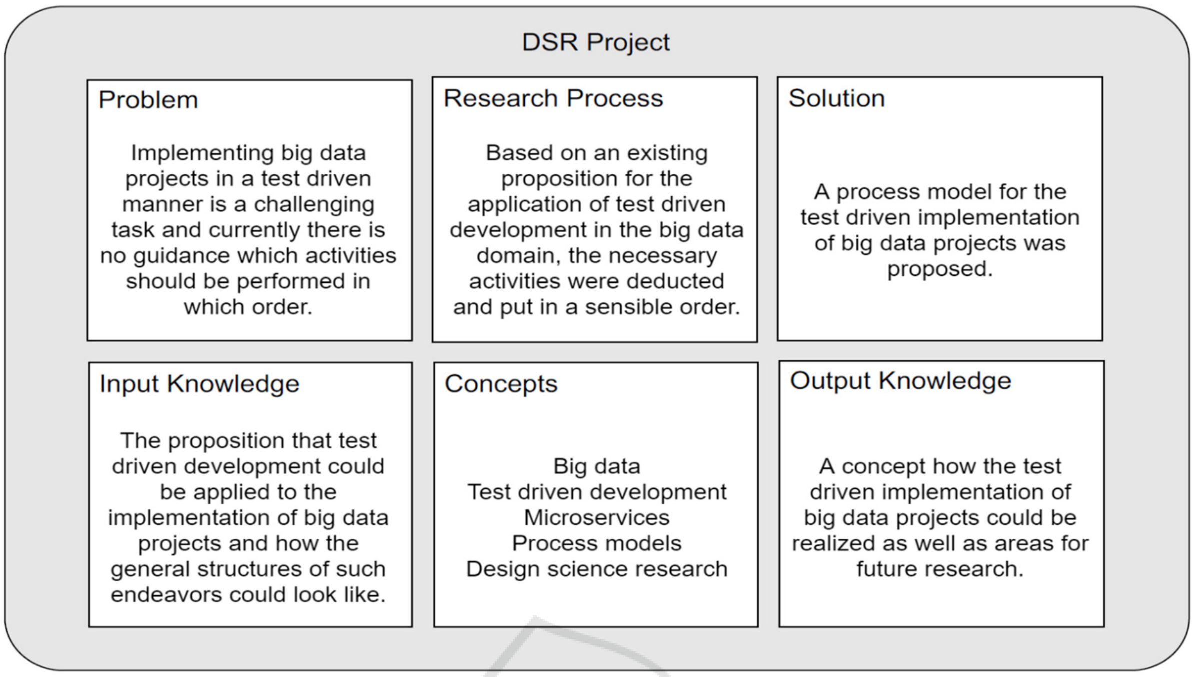
DSR Project
Problem
Implementing big data projects in a test driven manner is a challenging task and currently there is no guidance which activities should be performed in which order.
Research Process
Based on an existing proposition for the application of test driven development in the big data domain, the necessary activities were deducted and put in a sensible order.
Solution
A process model for the test driven implementation of big data projects was proposed.
Input Knowledge
The proposition that test driven development could be applied to the implementation of big data projects and how the general structures of such endeavors could look like.
Concepts
Big data
Test driven development
Microservices
Process models
Design science research
Output Knowledge
A concept how the test driven implementation of big data projects could be realized as well as areas for future research.

Figure 3: The DSR Grid for the Presented Work.

organizations from numerous domains rely on the ability to utilize information to an unprecedented extent to improve their inherent processes and decision making, and, thereby, inter alia, reduce their costs, increase their productivity, strengthen their marketing, support their maintenance, improve their logistics, or identify new opportunities. However, the implementation of those systems is a highly challenging and error-prone task, while at the same time their quality is crucial for the successful use. Therefore, their quality assurance is very important. Yet, this domain is still far from being mature. Therefore, further work in this field is needed. This includes the improvement of existing approaches and strategies as well as the exploration of new ones. One rather recent proposition was the application of test driven development to the implementation of big data systems. However, it was not outlined how the corresponding process should be designed.

The publication at hand bridges this gap and provides developers that are interested in the application of TDD in the BD domain with a process model that outlines, which activities should be performed in which order and, therefore, helps in structuring the implementation process. Thereby, it helps in disseminating the general approach, facilitates its effective utilization, promotes a stronger focus on the topic of quality assurance, and can be used as a foundation to advance the scientific discourse in the domain. An overview of the research endeavor in its entirety is given in Figure 3, in the form of the DSR Grid (Vom Brocke and Maedche 2019).

REFERENCES

Abdallah, M., Hammad, A., and Al-Zyadat, W. (2022). “Towards a Data Collection Quality Model for Big Data Applications,” in *Business Information Systems Workshops*, W. Abramowicz, S. Auer and M. Stróżyna (eds.), Cham: Springer International Publishing, pp. 103-108 (doi: 10.1007/978-3-031-04216-4_11).

Al-Mekhlal, M., and Khwaja, A. A. (2019). “A Synthesis of Big Data Definition and Characteristics,” in *Proceedings of the 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC),* New York, NY, USA. 01.08.2019 - 03.08.2019, IEEE, pp. 314-322 (doi: 10.1109/CSE/EUC.2019.00067).

Ataei, P., and Litchfield, A. (2020). “Big Data Reference Architectures, a systematic literature review,” in *Australasian Conference on Information Systems (ACIS) 2020,* Wellington, New Zealand, AIS.

Bahri, S., Zoghlami, N., Abed, M., and Tavares, J. M. R. S. (2019). “BIG DATA for Healthcare: A Survey,” *IEEE Access* (7), pp. 7397-7408 (doi: 10.1109/ACCESS.2018.2889180).

Beck, K. (2015). *Test-Driven Development*: *By Example*, Boston: Addison-Wesley.

Bronson, K., and Knezevic, I. (2016). “Big Data in food and agriculture,” *Big Data & Society* (3:1) (doi: 10.1177/2053951716648174).

Chang, W. L., and Grady, N. (2019). “NIST Big Data Interoperability Framework: Volume 1, Definitions,” *Special Publication (NIST SP),* Gaithersburg, MD: National Institute of Standards and Technology.

Crispin, L. (2006). “Driving Software Quality: How Test-Driven Development Impacts Software Quality,” *IEEE Software* (23:6), pp. 70-71 (doi: 10.1109/MS.2006.157).

Davies, K., Keet, C. M., and Lawrynowicz, A. (2019). “More Effective Ontology Authoring with Test-Driven Development and the TDDonto2 Tool,” *International Journal on Artificial Intelligence Tools* (28:7) (doi: 10.1142/S0218213019500234).

Diebold, F. X. (2012). “On the Origin(s) and Development of the Term 'Big Data',” *SSRN Electronic Journal* (doi: 10.2139/ssrn.2152421).

Dobre, C., and Xhafa, F. (2014). “Intelligent services for Big Data science,” *Future Generation Computer Systems* (37), pp. 267-281 (doi: 10.1016/j.future.2013.07.014).

Faitelson, D., Heinrich, R., and Tyszberowicz, S. (2018). “Functional Decomposition for Software Architecture Evolution,” in *Model-Driven Engineering and Software Development*, L. F. Pires, S. Hammoudi and B. Selic (eds.), Cham: Springer International Publishing, pp. 377-400 (doi: 10.1007/978-3-319-94764-8_16).

Freymann, A., Maier, F., Schaefer, K., and Böhnel, T. (2020). “Tackling the Six Fundamental Challenges of Big Data in Research Projects by Utilizing a Scalable and Modular Architecture,” in *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security,* Prague, Czech Republic. 07.05.2020 - 09.05.2020, SCITEPRESS - Science and Technology Publications, pp. 249-256 (doi: 10.5220/0009388602490256).

Fucci, D., Erdogmus, H., Turhan, B., Oivo, M., and Juristo, N. (2017). “A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last?” *IEEE Transactions on Software Engineering* (43:7), pp. 597-614 (doi: 10.1109/tse.2016.2616877).

Goes, F. R., Meerhoff, L. A., Bueno, M. J. O., Rodrigues, D. M., Moura, F. A., Brink, M. S., Elferink-Gemser, M. T., Knobbe, A. J., Cunha, S. A., Torres, R. S., and Lemmink, K. A. P. M. (2020). “Unlocking the potential of big data to support tactical performance analysis in professional soccer: A systematic review,” *European journal of sport science*, pp. 1-16 (doi: 10.1080/17461391.2020.1747552).

Günther, W. A., Rezazade Mehrizi, M. H., Huysman, M., and Feldberg, F. (2017). “Debating big data: A literature review on realizing value from big data,” *The Journal of Strategic Information Systems* (26:3), pp. 191-209 (doi: 10.1016/j.jsis.2017.07.003).

Häusler, R., Staegemann, D., Volk, M., Bosse, S., Bekel, C., and Turowski, K. (2020). “Generating Content-Compliant Training Data in Big Data Education,” in *Proceedings of the 12th CSEdu,* Prague, Czech Republic. 02.05.2020 - 04.05.2020, SCITEPRESS - Science and Technology Publications, pp. 104-110 (doi: 10.5220/0009513801040110).

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). “Design science in information systems research,” *MIS quarterly*, pp. 75-105.

ISO. (2018). “International Standard ISO / IEC / IEEE 29148 Systems and Software Engineering — Life Cycle process - Requirements Engineering,” *ISO/IEC/IEEE 29148:2018.*

Janzen, D., and Saiedian, H. (2005). “Test-driven development concepts, taxonomy, and future direction,” *Computer* (38:9), pp. 43-50 (doi: 10.1109/MC.2005.314).

Karlesky, M., Williams, G., Bereza, W., and Fletcher, M. (2007). “Mocking the Embedded World: Test-Driven Development, Continuous Integration, and Design Patterns,” in *Embedded Systems Conference,* San Jose, California, USA. 01.04.2007 - 05.04.2007, UBM Electronics.

Keet, C. M., and Ławrynowicz, A. (2016). “Test-Driven Development of Ontologies,” in *The Semantic Web. Latest Advances and New Domains*, H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S. P. Ponzetto and C. Lange (eds.), Cham: Springer International Publishing, pp. 642-657 (doi: 10.1007/978-3-319-34129-3_39).

Krylovskiy, A., Jahn, M., and Patti, E. (2015). “Designing a Smart City Internet of Things Platform with Microservice Architecture,” in *Proceedings of the 2015 3rd International Conference on Future Internet of Things and Cloud (FiCloud 2015)*, I. Awan (ed.), Rome, Italy. 24.08.2015 - 26.08.2015, Piscataway, NJ: IEEE, pp. 25-30 (doi: 10.1109/FiCloud.2015.55).

Kum, W., and Law, A. (2006). “Learning Effective Test Driven Development - Software Development Projects in an Energy Company,” in *Proceedings of the First International Conference on Software and Data Technologies,* Setúbal, Portugal. 11.09.2006 - 14.09.2006, SciTePress - Science and and Technology Publications, pp. 159-164 (doi: 10.5220/0001316101590164).

Levin, I., and Mamlok, D. (2021). “Culture and Society in the Digital Age,” *Information* (12:2), p. 68 (doi: 10.3390/info12020068).

Müller, O., Fay, M., and Vom Brocke, J. (2018). “The Effect of Big Data and Analytics on Firm Performance: An Econometric Analysis Considering Industry Characteristics,” *Journal of Management Information Systems* (35:2), pp. 488-509 (doi: 10.1080/07421222.2018.1451955).

Nadareishvili, I., Mitra, R., McLarty, M., and Amundsen, M. (2016). *Microservice architecture*: *Aligning principles, practices, and culture*, Beijing, Boston, Farnham, Sebastopol, Tokyo: O´Reilly.

Nagorny, K., Lima-Monteiro, P., Barata, J., and Colombo, A. W. (2017). “Big Data Analysis in Smart Manufacturing: A Review,” *International Journal of Communications, Network and System Sciences* (10:03), pp. 31-58 (doi: 10.4236/ijcns.2017.103003).

NASA. (2019). “Mars Climate Orbiter,” available at https://solarsystem.nasa.gov/missions/mars-climate-orbiter/in-depth/, accessed on Feb 27 2022.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems* (24:3), pp. 45-77 (doi: 10.2753/MIS0742-1222240302).

Sangwan, R. S., and Laplante, P. A. (2006). "Test-Driven Development in Large Projects," *IT Professional* (8:5), pp. 25-29 (doi: 10.1109/MITP.2006.122).

Shahin, M., Ali Babar, M., and Zhu, L. (2017). "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," *IEEE Access* (5), pp. 3909-3943 (doi: 10.1109/ACCESS.2017.2685629).

Shakir, A., Staegemann, D., Volk, M., Jamous, N., and Turowski, K. (2021). "Towards a Concept for Building a Big Data Architecture with Microservices," in *Proceedings of the 24th International Conference on Business Information Systems,* Hannover, Germany/virtual. 14.06.2021 - 17.06.2021, pp. 83-94 (doi: 10.52825/bis.v1i.67).

Shull, F., Melnik, G., Turhan, B., Layman, L., Diep, M., and Erdogmus, H. (2010). "What Do We Know about Test-Driven Development?" *IEEE Software* (27:6), pp. 16-19 (doi: 10.1109/MS.2010.152).

Silva, E. S., Hassani, H., and Madsen, D. Ø. (2019). "Big Data in fashion: transforming the retail sector," *Journal of Business Strategy* (41:4), pp. 21-27 (doi: 10.1108/JBS-04-2019-0062).

Slaats, T., Debois, S., and Hildebrandt, T. (2018). "Open to Change: A Theory for Iterative Test-Driven Modelling," in *Business Process Management*, M. Weske, M. Montali, I. Weber and J. Vom Brocke (eds.), Cham: Springer International Publishing, pp. 31-47 (doi: 10.1007/978-3-319-98648-7_3).

Sommerville, I. (2007). *Software Engineering, eighth edition*, Addison-Wesley.

Staegemann, D., Feuersenger, H., Volk, M., Liedtke, P., Arndt, H.-K., and Turowski, K. (2022a). "Investigating the Incorporation of Big Data in Management Information Systems," in *Business Information Systems Workshops*, W. Abramowicz, S. Auer and M. Stróżyna (eds.), Cham: Springer International Publishing, pp. 109-120 (doi: 10.1007/978-3-031-04216-4_12).

Staegemann, D., Volk, M., Jamous, N., and Turowski, K. (2019). "Understanding Issues in Big Data Applications - A Multidimensional Endeavor," in *Proceedings of the Twenty-fifth Americas Conference on Information Systems,* Cancun, Mexico. 15.08.2019 - 17.08.2019.

Staegemann, D., Volk, M., Jamous, N., and Turowski, K. (2020). "Exploring the Applicability of Test Driven Development in the Big Data Domain," in *Proceedings of the ACIS 2020,* Wellington, New Zealand. 01.12.2020 - 04.12.2020.

Staegemann, D., Volk, M., Lautenschlager, E., Pohl, M., Abdallah, M., and Turowski, K. (2021a). "Applying Test Driven Development in the Big Data Domain – Lessons From the Literature," in *2021 International Conference on Information Technology (ICIT),* Amman, Jordan. 14.07.2021 - 15.07.2021, IEEE, pp. 511-516 (doi: 10.1109/ICIT52682.2021.9491728).

Staegemann, D., Volk, M., Saxena, A., Pohl, M., Nahhas, A., Häusler, R., Abdallah, M., Bosse, S., Jamous, N., and Turowski, K. (2021b). "Challenges in Data Acquisition and Management in Big Data Environments," in *Proceedings of the 6th International Conference on Internet of Things, Big Data and Security,* Prague,Czech/Online Streaming. 23.04.2021 - 25.04.2021, SCITEPRESS - Science and Technology Publications, pp. 193-204 (doi: 10.5220/0010429001930204).

Staegemann, D., Volk, M., and Turowski, K. (2021c). "Quality Assurance in Big Data Engineering - A Metareview," *Complex Systems Informatics and Modeling Quarterly* (28), pp. 1-14 (doi: 10.7250/csimq.2021-28.01).

Staegemann, D., Volk, M., and Turowski, K. (2022b). "Adapting the (Big) Data Science Engineering Process to the Application of Test Driven Development," in *Proceedings of the 19th International Conference on Smart Business Technologies,* Lisbon, Portugal. 14.07.2022 - 16.07.2022, SCITEPRESS - Science and Technology Publications, pp. 120-129 (doi: 10.5220/0011289200003280).

Statista. (2021). "Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025," available at https://www.statista.com/statistics/ 871513/worldwide-data-created/, accessed on Feb 13 2022.

Volk, M., Staegemann, D., Bosse, S., Nahhas, A., and Turowski, K. (2020a). "Towards a Decision Support System for Big Data Projects," in *WI2020 Zentrale Tracks*, N. Gronau, M. Heine, K. Poustcchi and H. Krasnova (eds.), GITO Verlag, pp. 357-368 (doi: 10.30844/wi_2020_c11-volk).

Volk, M., Staegemann, D., Pohl, M., and Turowski, K. (2019). "Challenging Big Data Engineering: Positioning of Current and Future Development," in *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security,* Heraklion, Crete, Greece. 02.05.2019 - 04.05.2019, SCITEPRESS - Science and Technology Publications, pp. 351-358 (doi: 10.5220/0007748803510358).

Volk, M., Staegemann, D., and Turowski, K. (2020b). "Big Data," in *Handbuch Digitale Wirtschaft*, T. Kollmann (ed.), Wiesbaden: Springer Fachmedien Wiesbaden, pp. 1-18 (doi: 10.1007/978-3-658-17345-6_71-1).

Vom Brocke, J., and Maedche, A. (2019). "The DSR Grid: Six Core Dimensions for Effective Capturing of DSR Projects," *Electronic Markets* (July), pp. 1-17.

Williams, L., Maximilien, E. M., and Vouk, M. (2003). "Test-driven development as a defect-reduction practice," in *Proceedings of the 14th ISSRE,* Denver, Colorado, USA. 17.11.2003 - 20.11.2003, IEEE, pp. 34-45 (doi: 10.1109/ISSRE.2003.1251029).

Yin, S., and Kaynak, O. (2015). "Big Data for Modern Industry: Challenges and Trends [Point of View]," *Proceedings of the IEEE* (103:2), pp. 143-146 (doi: 10.1109/JPROC.2015.2388958).