A Tool for Developing and Evaluating Methods (TDEM)

Murad Huseynli, Michael Chima Ogbuachi and Udo Bub
Faculty of Informatics, Eötvös Loránd University (ELTE), Pázmany Péter sétány 1/C, H-1117 Budapest, Hungary

Keywords: Information Systems, Digital Innovation, Design Science Research, Method Engineering, Method Design, Method Evaluation, Process Evaluation, Microservices.

Abstract: Digital Innovation (DI) is the use of digital technologies during the process of innovation or as the result of innovation. While we see a huge amount of interest in DI research from IS scholars, there is still scarcity in terms of using Design Science Research (DSR) for the engineering of DI, specifically from a method perspective and a possible integration of both fields. In this paper, we propose a tool for developing and evaluating methods to build method design theories, not just in the DI field, but also generally applicable to method engineering in other fields as well. Thus, we contribute systematically to the body of knowledge for information systems with a tool that shall be an effective assistant in method engineering. The tool builds on the eight components of a design theory first introduced by (Gregor and Jones, 2007) and conceptualized by (Offermann et al., 2010a) for the artifact type "method". Finally, we analyze the role and utility of the tool in detail, concentrating on a method for the engineering of DI, in a DI project related to the microservices architecture, and lastly, we show generalizability of the tool in terms of design process evaluation, not specifically following a DSR paradigm.

1 INTRODUCTION

Design science research is generally applied to classes of artifacts, including algorithms, computer or human interfaces, design methods and techniques, and process models. It is primarily applied in engineering and computer science, but is not limited to these fields and can be found in many other disciplines and fields. Since the research paradigm of design science focuses on the creation of an innovative artifact, it contributes to the field of digital innovation from both a research and practical perspective. Digital innovation describes the use of digital technologies as an aid during the innovation process, or as an outcome of innovation. Digital innovation is increasingly becoming a dominant research focus in the field of information systems. Although the pursuit of innovation is crucial for forward-looking companies (Bub, 2018), little attention is paid to the systematic development of digital innovation in related fields such as IS and computer science, which play a key role in digital transformation. Innovation plays an important role in economic progress and consequently, in human well-being (Baumol, 2002). Information technologies are developing at a rapid pace, so digital innovation based on digital technologies has become the focus of attention. Basically, digital innovation combines physical and digital entities and creates new products on this basis (Hevner and Gregor, 2020). As digitized products and services are among the key drivers of innovation, digital innovation is now practiced by a growing number of companies (Berntsson-Svensson and Taghavianfar, 2015). At its core, digital innovation is about novel digital technologies, entrepreneurship and innovation management, and digitization of information and services.

At its essence, DSR in Information Systems (IS) is about DI (Hevner and Gregor, 2020). While this much synergy exists between both fields, not many studies have been conducted regarding method engineering for DI, using DSR. This gap was the motivation for this research. As we looked into existing literature we found that we could not identify a scientific tool that allowed researchers to perform proper method engineering, not only in DI, but also in any other IS fields where DSR methodologies can make a huge impact (considering research processes and outcomes). DSR is a generally accepted paradigm of Information Systems (IS) research (Hevner et al., 2004; Peffers et al., 2008). Among the artifact types that are subject to design in IS, "method" is of high relevance and subject to intensive research work in the IS community (Hevner et al., 2004; Bucher and Winter, 2008; Offermann et al., 2010a). Reviews of ex-

Huseynli, M., Ogbuachi, M. and Bub, U.
A Tool for Developing and Evaluating Methods (TDEM).
DOI: 10.5220/0011077700003179
In *Proceedings of the 24th International Conference on Enterprise Information Systems (ICEIS 2022) - Volume 2*, pages 543-552
ISBN: 978-989-758-569-2; ISSN: 2184-4992

isting literature (Huseynli et al., 2021) showed that there is still scarcity regarding methods for the engineering of DI using DSR. (Bub, 2018) takes the initiative for the development of such a method and puts the basis for it. This effort, though, needs further validation and improvement. Designing such a method requires a thorough, comprehensive knowledge base with both scientific rigor and practical relevance. Researchers face challenges in identifying academic insight and progress in comparison to the state of the art, due to the complex nature and diverse ways of describing method artifacts. While it is unavoidable to put much effort into identifying how to design and evaluate methods, to support these activities, we introduce a tool for use in method development and evaluation. Our proposal is scientifically based on IS Design Theories (Gregor, 2006) and preliminary practical considerations (Bub, 2020). In their seminal work, (Gregor and Jones, 2007) have identified eight components for an IS Design Theory: "Purpose and Scope", "Constructs", "Principles of Form and Function", "Artifact Mutability", "Testable Propositions", "Justificatory Knowledge", "Principles of Implementation", and "Expository Instantiation". Their work targets components of a Design Theory of Design and Action, meant to be independent of any artifact types and that require further detailing for concrete implementation (i.e. to define method artifacts). (Offermann et al., 2010a) have built on their work and have proposed detailed advice on how to describe methods in a scientific manner, by leveraging the aforementioned components to build method Design Theories. We further took the identified concepts from both research works and built a tool for developing and evaluating methods based on them. We abbreviate this tool as TDEM (A Tool for Developing and Evaluating Methods) and we illustrate the utility of TDEM with a thorough analysis.

We structured the overall process as follows. The concept of TDEM for Design Science in IS is introduced and its components are thoroughly explained. Subsequently, an analysis of the potential uses of TDEM is provided and two aspects are investigated: the analytical and ideational points of view. For the first one, an example is shown by providing a brief study (evaluation) on a scientific paper, while for the latter, a design project of DI in microservices was developed and is introduced, which discusses how TDEM participates in the ideation process and helps establish scientific knowledge in Design Science Research. Finally, we show the generalizability of TDEM on a non-DSR paper in terms of process evaluation.

2 BACKGROUND

In this section, we closely look into DI, DSR, and Method Engineering from a theoretical point of view.

2.1 Digital Innovation

Digital technologies transform organizations worldwide to a great extent. DI can be seen as the application of digital technologies to business problems. (Hevner et al., 2019) explains digital innovation as the appropriation of digital technologies during the process of and as the outcome of innovation. DI encompasses several directives just like developing a novel technology or technology strategy in a business context, adopting, and implementing technology resources, and digitizing work activities within an organization. Organizations that innovate or transform come across with several benefits, such as streamlining business processes by automating and integrating different technological artifacts and cutting costs as a result of having digital solutions, having a better return on investment, and boosting revenue streams.

2.2 Design Science Research

Design Science Research (DSR) has seen a vast amount of interest in Information Systems (IS), as an outcome-based research methodology that seeks to create novel technological artifacts. In its essence, Design Science (DS) pursues exploring, comprehending, and rectifying potential components, constructs, and methods to create and position original artifacts which are capable of solving an organizational problem in a natural setting (Baskerville, 2008). DSR aims to further improve the designed artifacts through its iterative approach, focusing on their development and evaluation. The output artifact resulting from DSR can be software, an algorithm, a mathematical equation, frameworks, process models, and so on. DSR outputs can be categorized in constructs, models, methods, implementations, architectures, design principles, frameworks, instantiations, and theories as described by researchers (Chatterjee, 2015; Vaishnavi and Kuechler, 2004). The all-round learning and establishment of knowledge, where it is essential to deliver innovative artifacts, is at the heart of Design Science Research. The actual creation process generally also covers the improvement of design artifacts and processes in their setting. As the iterative evaluation is the main part of DSR, it provides a finer explication of the problem and provides feedback to improve the quality and efficacy of the artifact and its design process.

2.3 Method Engineering

Initially, (Bergstra et al., 1985) came up with the term "Method Engineering" (ME), then (van Slooten and Brinkkemper, 1993) and (Brinkkemper, 1996) further supported the idea of ME. ME in IS focuses on designing, constructing, and evaluating methods, procedures, tools, and techniques to enhance information systems development (Brinkkemper, 1996). While computer engineering deals with hardware and firmware aspects, ME focuses on methods, procedures, and techniques in all engineering activities, since ME can relate to other research areas including project management, software configuration management, software engineering environments, and software process modelling (Brinkkemper, 1996). In addition, methods are the fundament of goal-driven and tactical actions in various application fields. A Method incorporates several parts that make the information systems development process more manageable (Sunyaev et al., 2008). There are several concepts regarding methods, such as Method Chains and Alliances (Nilsson, 1999), which are necessary when performing integration of methods and methods parts. Among the latter, we have: *Method Components* (Karlsson and Wistrand, 2003), which are defined as being part of a method, consisting of concept, notation, and process; *Method Fragments* (Harmsen et al., 1994), which are building blocks of methods; *Method Chunks* (Ralyté and Rolland, 2001), which are the combination of the existing process and product fragments.

3 A TOOL FOR DEVELOPING AND EVALUATING METHODS (TDEM)

(Gregor and Jones, 2007) established a design theory meant for information systems research, constituted by eight components. These eight components were taken by (Offermann et al., 2010a) and refined, characterizing an implementation that was specific for methods. In fact, it is important to remember that the original components were meant to define a process or structure for building Design Theory, which means they were meant for a much more generic scope. The aforementioned refinements added important operational detail in the structure, mostly in two forms: attributes of a component and evaluation criteria.

The attributes give added focus and structure and help define the elements that are most important for the definition of a method. A simple way of mentally visualizing these, which is particularly close to people working in Information Systems, would be by thinking of them like the attributes that define a class in Object Oriented Programming (OOP), which are the elements that constitute the internal properties of a type, thus characterize it (where a class is one of the eight components by (Gregor and Jones, 2007)). And again, as with the OOP class attributes, these can also be either singletons or collections of elements (some attributes may be specified by a plurality of elements, depending on the type of method that is being designed). See Figure 1 for the structure of TDEM.

The evaluation criteria constitute an inbound analytical tool for the process and are expressed in the form of questions. Each component has its set of questions (criteria), which allow a segmented study of the method that is being created, specific for each component. The questions are meant to be able to admit both Boolean (of the yes/no type) and complex answers. This level of detail, considering segmentation and structure of the questions, is very important, as it makes it possible to understand if the process for building a method is being utilized "correctly", or at least in a coherent manner. These questions were selected through a hermeneutical analysis of the components and revisited several times (Offermann et al., 2010a). They are posed in a way that would push the user to deeply assess the strategy that is being used to structure the method.

So, to briefly recapitulate, the process to structure a method should include the use of these eight components of Design Theory, tailored to be used within the area of Design Science Research. Each of these components is structured in such a way that they have attributes, to define them, and evaluation criteria, which help make an in-depth analysis of the method that is being built.

We gather all this important preparatory work as input for creating a new tool, TDEM. This can be fundamentally explained as a one-pager tool that gives space to all the augmented versions of the eight components of Design Theory, as explained in the previous paragraphs. This characterization as a tool makes it easier to understand the usage of the components and provides an easy way to arrange, also at a graphical level, such an abstract analytical process. Specifically, this can be seen as a tool to perform an in-depth study of a method and the artefacts that are produced as a result of the Design Science Research process. Further in this section, the eight components are shown in detail.

Purpose and Scope - This component describes what a method is meant to achieve and what area of a research field in Design Science it should cover. Purpose statements should provide information about the

A Tool for Developing and Evaluating Methods (TDEM): <Name of the Method>

Purpose and Scope

- Project type
- Project context
- Lifecycle coverage
- Activity coverage

Evaluation criteria: Given a problem, is the purpose described in a way to tell if the method solves the problem? Is the product of the method clear? Given a problem context, is the scope described in a way to tell if the context falls within the scope? Are all relevant dimensions specified within the scope? Are the covered lifecycle phases, roles and activities specified?

Constructs

- Method specific constructs, meta-model
- Output-specific constructs
- Concepts in the application context

Evaluation criteria: Are terms and concepts presented to describe the method? ... describe the resulting structure? ... describe the application context? ... describe the purpose and scope? ... create the testable propositions? Are the concepts introduced comprehensively in one location and derived systematically?

Testable Propositions

- Utility statement about theory
- Optional: truth statements about match between method design and requirements of method output

Evaluation criteria: Are testable propositions concerning the method utility given? Are the propositions sufficiently well operationalised to allow testing by other scientists?

Principles of Form and Function

- Description of the method (according to the meta-model selected above)

Evaluation criteria: Is the method described extensively enough to be useful and transferable? For each possible role, is it clear which activities have to be done in what order?

Justificatory Knowledge

- Design theories that this method is based on
- Theories about the application context
- Other aspects of interest

Evaluation Criteria: Are theories presented for key method features not covered by testable propositions? ... to support the testable propositions? ... regarding the method product? ... regarding the method setting?

Artifact Mutability

- Foreseeable changes (which part of the method, method base or method instance, kind of change)
- Optional: support for change offered by method design paradigm (i.e. situational, method rationale)

Evaluation criteria: Is the method described extensively enough to understand the functioning of each component? Are conditions for changes described? Does it become clear, which parts change and what they change into? Are method tailoring mechanisms defined?

Principles of Implementation

- Tailoring/assembly advice
- Advice regarding introduction into real-life setting

Evaluation criteria: Is the tailoring/assembly advice comprehensive with regard to the combinatorial possibility? Does the advice for introducing the method cover different settings within the scope, or is at least clear for which it applies?

Expository Instantiation

- Example of an instantiated method
- Report on a real-life case study in which an instantiated method has been enacted

Evaluation criteria: Does the example cover a case within the scope and is it covering the main concepts of the method? Is the example specific enough to be illustrative but not too idiosyncratic to be incomprehensible?

Figure 1: TDEM based on (Offermann et al., 2010a; Gregor and Jones, 2007).

type of output that the usage of a method should yield, the properties of the output, and some statements regarding the method itself. On top of that, the scope of the method is also described from an external and an internal point of view.

Scientific Role: Purpose and scope give the basis for the scientific significance of a theory or method, as they provide a clear vision of the end goal for which they are being developed, and describe what the research area should "look like". This strongly contributes to providing the generalisability that should be typical of a proper method or theory. When the scope is well defined, it is easier to generalize the validity of the scientific process.

Constructs - Constructs are properties that can be identified either from the method itself, from the structure of its output, or the context of the enactment of such method. One of the concepts that concretely participate in the constructs is that of Method itself, as methods can be generated from other method models or meta-models. Other constructs might come from product classes or justificatory knowledge as well.

Scientific Role: The constructs do not provide information that would directly help the evaluation of a theory or method, but they do help in setting the fundament for the description of all the other components of the process.

Principles of Form and Function: This component describes what the abstract architecture of the artefact should look like, especially when the artefact is a system. In case the artefact is a method, then there is no specification for different architectural models, but situational context is considered. A general method is described, which can be seen as a set of individual solutions. The principles of form and function establish a strict relation between the elements of the method and the meta-model that defined the internal structures. The way such elements are described must comply with the representation coming from the metamodel and having justificatory knowledge would be an added point.

Scientific Role: This component is strictly necessary for a method to be useful. In fact, it provides the information that allows the understanding of the purpose of the method, which is necessary for its coherent use.

Artefact Mutability: Mutability during a research process can happen both on the design stage of the method, or on a particular instance of the method itself. For methods, specifically, two types of mutability are identifiable. The first kind identifies the mutability, changes on the method itself, that the designer can predict. The second kind identifies the mutability that can be expected and is described in the original method design, but happens during the instantiation of the method.

Scientific Role: This is another component that greatly helps increase the possibility of generalization of a method, such as the principles of form and function. Understanding what changes (and in what manner) a method can undergo also helps to understand whether it is possible to transfer it across research domains.

Testable Propositions: Here it is possible to make an analysis and have a complete view of what are the statements made about the method or artefact of the process in general, that can be measured and tested. For simplicity, an overview of what these propositions are can be obtained using the following three questions: What are the statements about? What is asserted by the statement, utility or truth? How complete should the collection of testable propositions be?

Scientific Role: Testable propositions have the purpose of supporting the validity of the method, especially considering its utility statement. As in (Offermann et al., 2010a), "...a method is valid if it is useful in respect to its purpose.". Of course, an objective and testable analysis can only be made if the hypotheses are made in such a way that they can be reproduced by someone else.

Justificatory Knowledge: This component is meant to clearly state all the possible theories supporting the method, the product, and the context in which the method is being applied. As in (Gregor, 2009), there are different kinds of theory that can be selected, based on specific situations.

Scientific role: Justificatory knowledge is meant to support the possibility of transferring a method into different research domains and to prove its validity. It provides elements that can be used to repeat a proof of validity for the method in different contexts.

Principles of Implementation: This component helps explore what the necessary steps are during the implementation phase, which usually imply adapting a method to the specific scope (situation or organization) it is being used in. This is also known as tailoring process.

Scientific role: These principles help again support the statements about the utility of a method and define how such a method is transferable into different scopes.

Expository Instantiation: This component is meant to show a real-life application of the method that is being designed. This can be represented either as a description of the method applied in a specific context or as information about the execution of the method.

Scientific role: These principles, as the previous one, also support transferability and generalization, but provide more concrete information about the results coming from an actual tentative instantiation and use of the method.

4 USAGE DUALITY OF TDEM

In this section, we explore the possibilities that TDEM offers in Design Science. Specifically, two distinct modes are taken into consideration, which we named the analytical (evaluation) and the ideational (development) processes. Both processes are explored and illustrated by giving a specific example for each.

4.1 TDEM as an Analytical Tool

From what concerns the analytical point of view, TDEM proved to be a tool that is effective in the evaluation of artifacts of Design Science, specifically methods. In an earlier work, we explored a potential usage of TDEM as a framework component for process evaluation (Ogbuachi et al., 2021). In this subsection, the evaluation process is shown through the analysis of a method for the engineering of DI by (Bub, 2018), which describes design activities, design outcomes, and the necessary roles for organizations that deliver innovation. The essential construct is the process model, which shows the sequence of design activities, essentially concentrating on method engineering.

The analysis conducted on the approach using TDEM, revealed several interesting matters, which are illustrated here for each of the eight components of Design Science Research.

Purpose and Scope: The research was carried out within the scope of method engineering for digital innovation. The purpose of the paper is to develop a method for the engineering of DI and integrating a DSR process into the process model.

Constructs: Fundamental constructs can be easily identified from the method, such as an enhanced meta-model for situational method engineering, a stage-gate-oriented idea-to-launch process model, a stage-gate-oriented process model integrated with a DSR process, and a combined process model.

Testable Propositions: The utility of the proposed method is shown in the paper, through a description of how the method is applied in a university-industry co-innovation lab. The research process and implementation are also described.

Principles of Form and Function: The process model is clearly described in the paper. Design activities are in sequential order and they have to be performed "from left to right", completing each stage one after the other.

Justificatory Knowledge: Method design theories are presented, such as theories about the application context (including innovation management) and theories about DSR.

Artifact Mutability: After application in the original method setting, situational tailoring for project types (and similar adaptations) is encouraged for other settings, since the number of gates for innovation processes can be adjusted to specific needs of the respective enterprise.

Principles of Implementation: The method requires an already implemented idea-to-launch process, or willingness to implement it at the enterprise.

Expository Instantiation: The paper is illustrating a case study in which an instantiated method has been enacted.

4.2 TDEM as an Ideation Tool

TDEM worked well also as a creational tool for Design Science, thus representing a sort of duality if compared to the analytical approach that was described in the previous section. This new point of view is illustrated here, by a study that was conducted in a DI internal project, from the structuring of its fundamental parts to the verification of its completion. The project aimed at representing the subdivision of a simple monolithic application into a structure that tried to replicate the microservices architecture.

The project is based on the Stencil Parallel Pattern. This pattern suits specifically areas like image processing. In Stencil, every step of computation is done starting from a memory address (or an array index) and operating on offsets from that address. The result is then saved to an output array, specifically to an index that corresponds to the starting memory location for the input. The image processing program consists of a set of algorithms that run consecutively on a given image dataset, using this pattern.

The pipeline is constituted by execution units (functions) that are executed in sequence. Given the structure of the application, the plan consisted in analysing what the functional units were, that could be defined independently from all the others, and transforming those into actual applications that would run as small services.

As mentioned earlier, in this specific case the functional units are the functions that implement the image processing algorithms. Those functions are

therefore encapsulated into fine-grained services that are able to retrieve an image as an input, process it by applying one specific filter to the image, and give back an output as a result, having only a few minor requirements on the input image. Each service is able to operate independently from who sends the input, thus fulfilling one of the core characteristics of a microservices-based architecture.

It is important to remember, though, that in this specific example the functionality of the full monolithic application should also be preserved, after the transition to a microservices-based architecture. Despite the structure being very different now from an architectural point of view, the original functionality could still be preserved by keeping the temporal point of view unchanged. By this, we mean that the original order of operations was sequential in the original monolithic application, as it is typical of the general imperative programming approach, but that same order could be replicated in the new architecture as well, by enforcing the sequentiality of the operations through a structured series of service calls and data transfers. The intended sequence then looks as shown in Figure 2.

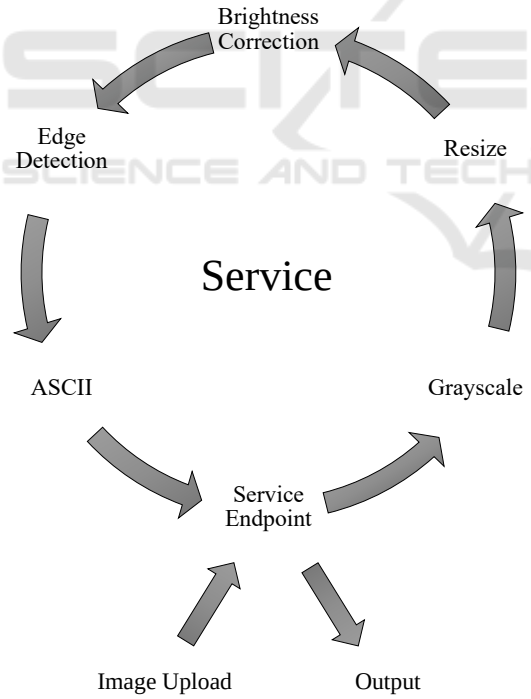

Figure 2: Sequential order of operations.

Therefore, the initial image is now uploaded to an endpoint of the service, which acts as the only component the user is allowed to directly interact with. This endpoint should then send the image that was uploaded by the user to the first step of the image filtering process, which is the Grayscale function. After the execution is complete, the output gray image is then sent to the next component, Resize, and so on, until the last one, ASCII text-based rendering, completes its elaboration and returns the image as a file to the user, as a response from the same aforementioned service endpoint. In our set-up, each function provides an input to the next one, completing the pipeline. By restructuring the application in a microservice architecture, we explored the possibility of enforcing synchronous behavior (originating from the monolithic application) in an environment (in this case, the network) that is inherently asynchronous. Having a guideline to define the development process, helped us identify the potential issues (such as communication issues, segregation of data, scalability of workload, etc.) that could arise when building a service that strongly relies on asynchronous interactions.

As before, an analysis based on the eight components of TDEM is provided, with the different focus presented in this section.

Purpose and Scope: The purpose of the project can be explained easily as the creation of a method to define microservices from an existing monolithic application. Trying to identify the context, the best option is software engineering, specifically image processing and networking. The definition of a lifecycle coverage process was not entirely defined, as it was partly out of the scope of the process (only the design phase is considered). While structuring the method, we described the purpose in a way that can tell if the method solves the proposed problem, and it does, especially the problem of splitting a pre-existing application into smaller parts. The approach we adopted is general enough to be adaptable to the specific problem.

Constructs: The construct that comes as an output of this project is a modelling approach to the definition of a microservice from a pre-existing application. Thus, the main construct is a purely theoretical high-level approach. The application context would be Autonomous processing services and Microservice candidates. We described the basic concepts needed to present the method, the resulting structure, and the application context.

Testable Propositions: As a utility statement, we agreed that this approach should allow an easier definition for the subdivision of components when creating a new microservice system by splitting existing systems and, in this specific case, the method tries to fulfil completely the problem it was used for. The work that has been done should be enough information to allow at least the unit testing of components, in most cases, testable propositions about the method

utility are also given.

Principles of Form and Function: The method described in this paper is meant to be a starting point for the creation of microservices based on the segmentation of an application. The method we produced was extensively described, and the amount of information provided is enough to understand that it is adaptable. For each role in the project, the necessary activities and their order were clearly specified. In fact, the method is described with step-by-step clarity (keeping the descriptions very high-level).

Justificatory Knowledge: The proposed method is based on Domain Driven Design and Microservices. It is still general, but the instantiation is image processing as a domain. The main theory is that monolithic, computationally expensive applications can be split into smaller parts, to which we added attention to the fact that the knowledge base of the applications can be singular or plural, both before and after the splitting process, and both approaches could actually be beneficial (indirectly), especially while considering efficiency.

Artifact Mutability: Considering the foreseeable changes, we estimated that, because of the generic nature of this method, which is also based on very simple patterns, such changes are minor and would not extensively change the defined structure. More specifically, the foreseeable changes are mostly based on small adaptations to the application scopes. The only method tailoring mechanism that was defined was for the given example, trying to put it in a more generic context.

Principles of Implementation: About the tailoring or assembly advice we provided, most of it is about the definition of "autonomous entities", and mostly about the use in the suggested scope, without excluding the consideration of several other areas. Particular attention has been put in making it clear for the proposed scope.

Expository Instantiation: The method is instantiated only within the provided example. There is no report of the sort of real-life case study in the paper. Mostly the use we made was for laboratory experiments in a controlled environment. The provided example defined the operational scope and it is specific enough to be illustrative about the applied method.

5 GENERALIZABILITY OF TDEM

TDEM can also be used as a tool for evaluating processes, which are not being necessarily developed using DSR paradigm. We show this through the analysis of a design approach structured by O'Connor, Elger, and Clarke in (O'Connor et al., 2016), which is meant to help enhance development (especially in IT-related fields) by taking into consideration the so-called situational context of a project.

As described in the paper, situational context can be of utmost importance when designing a process. It can be defined as the set of elements that may have a direct influence on a project and its outcome, from the Enterprise level to the actual development phases. In particular, (O'Connor et al., 2016) show that the elements identified as components of this context are the personnel, requirements, application (the software project itself), technology, organization, operation, management, and business. The first four elements are the ones having the highest impact on the design stages. The personnel is identified as the non-managerial characters participating first-hand in the development of the software application (the software engineers), while the requirements are all the implementation needs and rules as defined in the software engineering standard ISO 12207 (Singh, 1996), which can be characterized by the commissioner or the development team. The technology that is available in the company affects directly the capabilities of the aforementioned personnel and the required development time.

The analysis conducted on this approach using TDEM, revealed several interesting matters, which are illustrated here for each of the eight components.

Purpose and Scope: The type, context, lifecycle, and activity coverage of the project could be explained fairly easily. The project was carried out within the scope of software engineering for microservices, and the lifecycle management corresponds to what is often seen in the paradigm known as "agile development". The purpose of the paper was not to show a method, specifically, but rather showing through a concrete example how situational context affects project management.

Constructs: The "products" of the project can be plainly pointed out. Specifically, the output-specific construct here is a framework for situational analysis which is fundamental to understand the necessary adaptation steps required for general methods, when applying them to any concrete project. One notable remark is that the use of terms and concepts in the paper has been made in such a way that allows the entire framework to be easily understandable, to the point that an entire section was dedicated to the definition of concepts and their utility.

Testable Propositions: The utility of the proposed theory is illustrated across the paper and can be summarized with one proposition: the suggested

analysis should provide a better view of how to analyse the effects of context on development. Although there are no specifically defined measures, the information provided by the authors includes the results of actual tests conducted in an enterprise.

Principles of Form and Function: This paper provides a framework to analyze the environment in which a microservices-based software system is being developed, to easily adapt generic methods to the needs of specific cases. The suggested approach is described very extensively and the amount of information provided is enough to understand how the adaptation process works. The framework is described with precise detail and great clarity (as mentioned earlier, a use case example is also shown).

Justificatory Knowledge: This component helps identify some important properties regarding the background knowledge needed to justify the theories that are discussed in the paper. Starting from the design theories, which are based on the agile and lean development principles and the microservices architecture, it is understandable that the majorly observed scope is that of web applications and full-stack operations. Other aspects of interest are the precision of the framework and the possibility of extending it. Theories supporting the testable propositions and regarding the product of the research are also presented.

Artifact Mutability: Foreseeable changes are to be taken into consideration. In fact, because of the generic nature of this framework, which is also based on very simple discernment strategies, it is correct to say that the foreseeable changes are minor and wouldn't extensively change the defined structure of the framework itself. It can also be added that the foreseeable changes are mostly situational, based on adaptations to the application scopes, and they are supposed to be like that. The framework resulting from this work is described extensively to allow its use as a method component and, since change is considered to be a fundamental factor through the whole process, tailoring mechanisms have been defined for the given example and for more generic contexts.

Principles of Implementation: It would not be wrong to affirm that the paper is meant to emphasize and prove the importance of tailoring as a principle. Some advice regarding the introduction of the approach into real-life settings is also included, together with an extensive example conducted in a real company. Generally speaking, the advice is mostly clear for the setting in which it is applied, but also helps a general understanding for an adaptation into different scopes.

Expository Instantiation: An expository instantiation is not missing, as the approach has been instantiated in the provided example, which is practically an enactment within a real-life case study. The example covers the main concepts illustrated by the proposed approach and tackles the issues of a case that fully belongs to the predefined scope.

6 CONCLUSION AND FUTURE WORK

There exists a huge amount of interest in DI research from IS scholars. As of our previous research (Huseynli et al., 2021), we found out that there is still scarcity in terms of using Design Science Research (DSR) for the engineering of DI, especially from a method perspective, and a possible integration of both fields. While getting motivation for developing a method for the engineering of DI, using and integrating DSR, we came across early research works - (Gregor and Jones, 2007) and (Offermann et al., 2010a) - and based on the concepts illustrated in both works, we proposed a tool for developing and evaluating methods which we called TDEM. We showed the utility of TDEM as an evaluation tool by analysing a paper from the DI-DSR research area and also illustrated its generalizability capabilities in non-DSR works, on a paper based on continuous software engineering from a microservices perspective. TDEM is also useful as a development tool for methods, and we described this through a DI internal project. We believe that TDEM is a useful contribution to the body of knowledge for information systems and shall be utilized while developing and evaluating methods.

Our work focuses on the very important artifact type "method". (March and Smith, 1995) identify four artifact types from design science: constructs, models, methods, and instantiations. Moreover, (Offermann et al., 2010b) identified eight different artifact types from an extensive literature review of design science publications: Systems Design, Method, Language/Notation, Algorithm, Guideline, Requirements, Pattern, Metric. In future work, we would like to develop similar tools also for artifact types other than methods, possibly starting from system design.

REFERENCES

Baskerville, R. (2008). What design science is not. *European Journal of Information Systems*, 17:441–443.

Baumol, W. J. (2002). The free-market innovation machine - analyzing the growth miracle of capitalism. *Journal of Economics*, 82:93–97.

Bergstra, J., Jonkers, H., and Obbink, J. (1985). A software

development model for method engineering. *Esprit*, 84:84–94.

Berntsson-Svensson, R. and Taghavianfar, M. (2015). Selecting creativity techniques for creative requirements: An evaluation of four techniques using creativity workshops. *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pages 66–75.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Inf. Softw. Technol.*, 38:275–280.

Bub, U. (2018). Towards an integrated method for the engineering of digital innovation and design science research. In *European Conference on Advances in Databases and Information Systems*, pages 327–338. Springer.

Bub, U. (2020). Methodenvergleich und methodenentwicklung mit einem canvas. *Wirtschaftsinformatik & Management*, 12:146–157.

Bucher, T. and Winter, R. (2008). Dissemination and importance of the "method" artifact in the context of design research for information systems. In Vaishnavi, V. and Baskerville, R., editors, *Proceedings of the Third International Conference on Design Science Research in Information Systems and Technology (DESRIST 2008)*, pages 39–59, Atlanta, GA. Georgia State University.

Chatterjee, S. (2015). Writing my next design science research master-piece: But how do i make a theoretical contribution to dsr? In *ECIS 2015 Completed Research Papers*.

Gregor, S. (2009). Building theory in the sciences of the artificial. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology*, DESRIST '09, New York, NY, USA. Association for Computing Machinery.

Gregor, S. D. (2006). The nature of theory in information systems. *MIS Q.*, 30:611–642.

Gregor, S. D. and Jones, D. (2007). The anatomy of a design theory. *J. Assoc. Inf. Syst.*, 8:19.

Harmsen, F., Brinkkemper, S., and Oei, J. L. H. (1994). Situational method engineering for informational system project approaches. In *Methods and Associated Tools for the Information Systems Life Cycle*.

Hevner, A. R. and Gregor, S. D. (2020). Envisioning entrepreneurship and digital innovation through a design science research lens: A matrix approach. *Information & Management*, page 103350.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Q.*, 28:75–105.

Hevner, A. R., vom Brocke, J., and Maedche, A. (2019). Roles of digital innovation in design science research. *Business & Information Systems Engineering*, 61:3–8.

Huseynli, M., Podder, I., Ogbuachi, M. C., and Bub, U. (2021). Digital innovation and design science research: The state-of-the-art review and integrated innovation strategies framework. In *Proceedings of the 10th Jubilee Interdisciplinary Doctoral Conference 2021*, pages 355 – 368.

Karlsson, F. and Wistrand, K. (2003). Method components - extending the vision. In *Proceedings of the 26th Information Systems Research Seminar in Scandinavia (IRIS 26), Haikko Manor, Finland*.

March, S. T. and Smith, G. F. (1995). Design and natural science research on information technology. *Decis. Support Syst.*, 15:251–266.

Nilsson, A. G. (1999). The business developer's toolbox: Chains and alliances between established methods. In *Perspectives on Business Modelling*, pages 217–241. Springer.

O'Connor, R. V., Elger, P., and Clarke, P. M. (2016). Exploring the impact of situational context — a case study of a software development process for a microservices architecture. *2016 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, pages 6–10.

Offermann, P., Blom, S., Levina, O., and Bub, U. (2010a). Vorschlag für komponenten von methodendesigntheorien. *WIRTSCHAFTSINFORMATIK*, 52:287–297.

Offermann, P., Blom, S., Schönherr, M., and Bub, U. (2010b). Artifact types in information systems design science - a literature review. In *DESRIST*.

Ogbuachi, M. C., Podder, I., Bub, U., and Huseynli, M. (2021). A framework for quantifiable process improvement through method fragments in situational method engineering. In *2021 3rd International Conference on Advanced Information Science and System (AISS 2021)*, AISS 2021, New York, NY, USA. Association for Computing Machinery.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2008). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24:45 – 77.

Ralyté, J. and Rolland, C. (2001). An approach for method reengineering. In *International Conference on Conceptual Modeling*, pages 471–484. Springer.

Singh, R. (1996). International standard iso/iec 12207 software life cycle processes. *Software Process: Improvement and Practice*, 2:35–50.

Sunyaev, A., Hansen, M., and Krcmar, H. (2008). Method engineering: A formal description. In *ISD*.

Vaishnavi, V. K. and Kuechler, W. L. (2004). Design Science Research in Information Systems. *Ais*, pages 1–45.

van Slooten, K. and Brinkkemper, S. (1993). A method engineering approach to information systems development. In *Information System Development Process*.