Organizing Empirical Studies as Learning Iterations in Design Science Research Projects

Monalessa P. Barcellos
Ontology & Conceptual Modeling Research Group (NEMO), Computer Science Department, Federal University of Espírito Santo
Vitória, ES, Brazil
monalessa@inf.ufes.br

Gleison Santos
Federal University of the State of Rio de Janeiro
Rio de Janeiro, RJ, Brazil
gleison.santos@uniriotec.br

Tayana U. Conte
Federal University of Amazonas
Manaus, AM, Brazil
tayana@icomp.ufam.edu.br

Bianca Trinkenreich
Northern of Arizona University
Flagstaff, AZ, USA
bianca_trinkenreich@nau.edu

Patricia G. F. Matsubara
Federal University of Amazonas
Manaus, AM, & Federal University of Mato Grosso do Sul
Campo Grande, MS, Brazil
patriciagfm@icomp.ufam.edu.br

ABSTRACT

Software Quality is a relevant topic that interests both Academy and Industry. Hence, research on this topic should be aligned with the Industry needs, which demands the adoption of research approaches that enable closer interaction between researchers and practitioners. In this context, Design Science Research (DSR) stands out as a way to reduce the gap between theory and practice. DSR is a methodological approach to building innovative artifacts to solve real-world problems and, at the same time, making a scientific contribution. As a problem-oriented research method, DSR seeks to understand the problem, build and evaluate artifacts that allow transforming situations, changing their conditions to better or desirable states. In DSR projects, empirical studies have been usually applied to evaluate the proposed artifact. However, they can also be used to support other activities. Over the last eight years, we have successfully used empirical studies with different purposes in DSR projects. We organized the studies as learning iterations that provide useful knowledge to understand the problem, ground the artifact, develop, evaluate and improve it. As a result, we have experienced a more fluid DSR process and the proposed artifacts have been better grounded and suitable for solving the aimed problem. In this paper, we share our experience by discussing how we have used empirical studies as learning iterations in DSR projects, presenting our approach to organizing empirical studies in a DSR project according to the study purpose and the knowledge it intends to capture, and summarizing two DSR projects that address software quality issues and were developed by using such approach.

SBQS '22, Nov 7–10, 2022, Curitiba, Brazil
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9999-9/22/11…$15.00
https://doi.org/10.1145/3571473.3571474

CCS CONCEPTS

• **General and reference → Empirical studies**.

KEYWORDS

Design Science, Empirical Study, Empirical Software Engineering

ACM Reference Format:
Monalessa P. Barcellos, Gleison Santos, Tayana U. Conte, Bianca Trinkenreich, and Patricia G. F. Matsubara. 2022. Organizing Empirical Studies as Learning Iterations in Design Science Research Projects. In *SBQS '22: Brazilian Symposium on Software Quality, Nov 7–10, 2022, Curitiba, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3571473.3571474

1 INTRODUCTION

Knowledge about how to achieve software quality has increased over the years as a result of the work of researchers that investigate how software is or should be developed. However, technologies (e.g., procedures, methods, techniques, tools) proposed in the academic universe do not always find acceptance by the industry. This may be influenced by a possible lack of alignment between the research and the industry needs. The alignment between academy and industry is indispensable for researchers to produce useful results that enable the industry to overcome challenges today and in the future. Such results typically come from the observation of real problems and scenarios, which provides inputs to propose new solutions [14].

In the last years, the interest in better aligning Software Engineering (and, thus, Software Quality) research with industry needs has increased [8]. Moreover, the adoption of Design Science Research (DSR) has been recognized as a suitable methodological approach for developing artifacts to solve real-world problems [50]. Researchers from the Software Quality community have adopted DSR to support the creation of artifacts in diverse topics, such as technical debt prioritization [34] and continuous planning [37].

DSR addresses the design and investigation of artifacts in context. The artifacts are designed to interact with a problem context in order to improve something in that context [48]. As a science, design science has to do with the systematic creation of knowledge about, and with, design. It extends to the scientific study of design and the use of design processes in the scientific creation of knowledge [6].

According to Hevner [15], DSR has an iterative process containing three interconnected cycles. In a nutshell, the Relevance cycle initiates the research with the identification of the problem, application context, requirements for the research and acceptance criteria for the ultimate evaluation of the research results. The Design cycle iterates between the construction of an artifact, its evaluation, and subsequent feedback to refine the design further. The Rigor cycle provides past knowledge to the project to ensure its innovation and additions to the knowledge base as a result of the research.

Several authors (e.g., [48] and [6]) have suggested the use of empirical studies in DSR projects to properly evaluate the proposed artifact. Although the use of empirical studies in DSR projects has focused on the Design cycle, particularly on the artifact evaluation activity, they can also be used to support other activities related to that and the other cycles of the DSR process. For example, preliminary studies can be carried out in the Relevance cycle to better understand the problem and establish the research requirements. In the Design cycle, studies can also be performed to support decisions on the artifact design aiming to meet the established requirements properly. Even though some authors (e.g., [48]) state that empirical studies can be performed in different moments of a DSR project, there is still a lack of clear guidance on how to conduct such studies with different purposes throughout a DSR project.

Over the last eight years, we have successfully used empirical studies with different purposes in DSR projects. We organized the studies as *learning iterations* (LIs) —i.e., studies performed in iterations that allow the researcher to learn something— which provide useful knowledge to understand the problem, develop the artifact, evaluate and improve it. As a result, we have experienced a more fluid DSR process, which harmonizes several studies in iterations, contributing to research soundness. Moreover, the studies also contribute to the development of better-grounded artifacts, which are expected to be more suitable for solving the target problem.

Hevner [15] claims that it is vital that we as a research community provide clear and consistent definitions, guidelines and deliverables for the design and execution of high-quality DSR projects. Understanding and communicating the DSR process is essential not only to support acceptance among professionals but also to establish the credibility of DSR among the larger body of design science researchers. In this sense, in this paper, we share our experience by discussing how we have used empirical studies as LIs in DSR projects. The main contributions of this paper are (i) the approach that we have adopted in DSR projects to organize and harmonize empirical studies considering the DSR cycles proposed by Hevner [15]; (ii) the discussion about the use of empirical studies in DSR projects; and (iii) examples of DSR projects where empirical studies were applied as LIs aiming to build suitable and innovative artifacts. By portraying empirical studies more explicitly in DSR cycles, we also contribute to emphasizing DSR projects as scientific endeavors. Through empirical studies, DSR projects go beyond solving a relevant industry problem. They also advance our comprehension of relevant phenomena, through a conscious pursuit of scientific evidence [49].

The paper is organized as follows. Section 2 presents the background for the paper and discusses some related work. In Section 3, we introduce the approach that we have used to organize empirical studies in a DSR project according to the study purpose and the knowledge it intends to capture. Section 4 briefly presents two DSR projects developed using our approach. In Section 5, we make a discussion on the harmonized use of empirical studies throughout DSR projects. Finally, in Section 6, we make final considerations, discuss limitations of our work and reflect on future research.

2 BACKGROUND

2.1 Design Science Research

Design science aims at creating and evaluating artifacts intended to solve real-world problems. It involves a rigorous process to design artifacts to solve observed problems, make research contributions, evaluate the designs, and communicate the results. Such artifacts may include constructs, models, methods and instantiations [16]. They may also include social innovations [3] or new properties of technical, social or informational resources [17].

According to Hevner et al. [16], the fundamental principle of DSR is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artifact. That is, DSR requires the creation of an innovative, purposeful artifact for a specified problem domain. Because the artifact is purposeful, it must yield utility for the specified problem. Hence, a thorough evaluation of the artifact is crucial. Novelty is similarly crucial since the artifact must be innovative, solving a heretofore unsolved problem or solving a known problem in a more effective or efficient manner. In this way, DSR differentiates from the practice of design. The artifact itself must be rigorously defined, formally represented, coherent and internally consistent. The process by which it is created (and often the artifact itself) incorporates or enables a search process whereby a problem space is constructed and a mechanism is posed or enacted to find an effective solution. Finally, the results must be communicated effectively both to a technical audience (researchers who will extend them and practitioners who will implement them) and to a managerial audience (researchers who will study them in context and practitioners who will decide if they should be implemented within their organizations).

In the literature, there are some works defining processes, frameworks or guidelines for DSR. Hevner [15], for example, based on the framework combining Design Science and Behavioral Science proposed in [16], presents DSR as a set of three closely related cycles. The Relevance cycle bridges the contextual environment of the research project with the design science activities. The Rigor cycle, in turn, connects the design science activities with the knowledge base of scientific foundations, experience, and expertise that grounds the research project. The Design cycle iterates between the core activities of building and evaluating the design artifacts and processes of the research.

Peffers et al. [32] defined a process composed of six activities, namely: Problem Identification and Motivation, Definition of the Objectives for a Solution, Design and Development, Demonstration, Evaluation, and Communication. Wieringa [48], in turn, claims that DSR involves two parts: design and investigation. Each part is addressed in a specific cycle. Design is treated in the Design cycle, which is responsible for artifact development and evaluation. It is part of a larger cycle, called Engineering cycle, in which the result of the Design cycle (i.e., a validated artifact) is transferred to the real world, used, and evaluated. Investigation is addressed in the Empirical cycle, which is a rational way to answer scientific knowledge questions. Runeson et al. [35] propose the use of the design science paradigm as a frame for articulating and communicating prescriptive software engineering research contributions. In their perspective, design science embraces problem conceptualization, solution (or artifact) design, and validation of solution proposals, with recommendations for practice phrased as technological rules.

Although these works provide useful guidelines for performing DSR projects, some aspects still need clarification and additional guidelines. Concerning empirical studies, Peffers et al. [32], Hevner [15] and Hevner et al. [16] focus their use only on supporting artifact evaluation activities. Wieringa [48], in turn, besides advocating the use of empirical studies to evaluate the artifact in context, considers that such studies can also be performed to investigate questions aiming to get relevant knowledge to build the artifact. Even so, there are some gaps about which study should be performed at different moments of a DSR project.

Acknowledging the need for more concrete guidance on how to conduct studies in DSR projects, Offermann et al. [31] proposed a design science process organized in three phases (problem identification, solution design, and evaluation) and, in each phase, the authors propose the use of specific research methods. For instance, for problem identification, they recommend researchers carry out a literature review and expert interviews to build a solid foundation for the remaining of the research process. Recently, this approach has been regarded as well-aligned and fit for supporting research in industry-academia collaboration in the Software Engineering context [50]. Our work is in line with the proposal by Offermann et al. [31], but we generalize from specific research methods that can be adopted in each phase to the use of studies as LIs, i.e., empirical studies that can use any research method the researcher considers the best to iteratively and increasingly *learn* more about the problem or the solution. Moreover, we make the LIs explicit in the DSR cycles of Hevner [15].

2.2 Empirical Studies

Empirical studies enable researchers to produce new evidence as well as capture relevant past evidence recorded in the literature and other knowledge sources. The most basic source of empirical evidence are primary studies, in which we make direct measurements of the objects of interest [22]. In these studies, we can apply research methods such as controlled experiments, case studies, surveys, among others. A controlled experiment investigates a testable research hypothesis through the manipulation of a set of independent variables to measure their effects on a set of dependent variables. A case study investigates a contemporary phenomenon in its context, providing an in-depth understanding of how or why it occurs. A survey aims at identifying the characteristics of a broad population by generalizing from the data collected from a representative sample of individuals [10]. The execution of an empirical study using any of such methods involves performing a set of well-defined activities, ending up with the reporting of results and the packaging of study material and data [11], so that other researchers and practitioners can obtain knowledge about relevant outcomes.

Another source of empirical evidence are secondary studies, which are research efforts that do not generate data from direct measurements [22] but aggregate increasingly relevant evidence from primary studies [11][1]. The most common types of secondary studies are Systematic Literature Reviews (SLRs) and Systematic Literature Mappings (SLMs) [22]. The former aims at identifying and synthesizing all relevant material regarding a given topic through the use of objective, analytical, and repeatable procedures [22]. The latter also relies on systematic procedures for data search, collection, and analysis. However, the aim is to provide an overview of a research area, through classifications and categorizations, highlighting well-explored topics and gaps [33]. Primary and secondary studies are complementary sources of evidence [24] that can be used as part of DSR projects [31].

3 USING EMPIRICAL STUDIES AS LEARNING ITERATIONS IN DSR

Over the last years, we have used empirical studies throughout DSR projects by considering two main DSR foundations: the three cycles defined by Hevner [15] (Relevance, Design and Rigor) and the two parts of DSR considered by Wieringa [48] (Design and Investigation). The three cycles [15] encompass the DSR activities and make clear the relations among different components involved in a DSR project. The two parts of DSR [48], in turn, refer to two kinds of research problems. Design problems call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals. A solution is a design, and there are usually many different solutions. There may even be as not one single best solution. Investigation problems, in turn, are based on knowledge questions, which ask for knowledge about the world [48].

Like Wieringa [48], we also advocate the use of empirical studies throughout DSR projects to answer knowledge questions. By experiencing that in several DSR projects, we noticed that such studies differ mainly in their purpose and target knowledge. The studies came in for building knowledge—even related to the artifact design or evaluation—thus, we adopted them as *learning iterations*, i.e., studies performed iteratively and, in each iteration, the researcher could "learn something" (i.e., acquire knowledge) from the obtained results, which were based on the knowledge questions that motivated the study.

LIs can take place in any of the DSR cycles (Relevance, Design and Rigor). The cycle where the study is performed directly influences the study purpose and target knowledge. Considering that, we have defined five categories and organized the studies as LIs that support (i) *problem investigation*; (ii) *artifact foundation*; (iii) *artifact design*; (iv) *artifact evaluation*; and (v) *artifact evolution*. Figure 1 adapts Hevner's proposal [15] and shows empirical studies as LIs that serve as a basis for DSR, providing useful knowledge for developing the aimed artifact. In the figure, the grey area represents the DSR project. The yellow, green and pink rectangles refer to information related to Hevner's cycles (shown in blue). Empirical studies appear at the bottom as LIs, serving as the basis of evidence to the research project. They can be performed in the context of any cycle, according to the aimed knowledge questions.

[1]Similarly, tertiary studies aggregate evidence from multiple secondary studies [22].

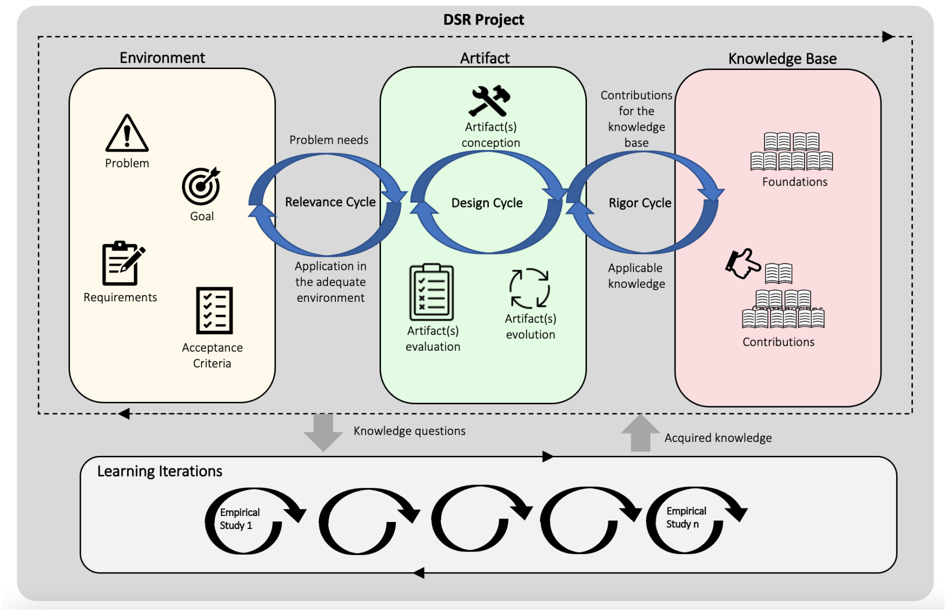

Figure 1: Learning Iterations in DSR

Each study is a LI in the sense that as the researcher performs studies, he/she learns more about the research and the aimed artifact. Figure 2 depicts a LI. The study purpose and the knowledge questions help define the empirical study to be performed (e.g., a systematic literature review, a case study, a controlled experiment) and its scope. After performing the study, the researcher gets knowledge about the problem or to ground, develop or evolve the artifact(s). New knowledge questions can also arise and lead to another study in a new LI. A LI contains the LI plan, the empirical study package, and the LI report. The plan establishes the LI purpose and knowledge questions, identifies which empirical study will be carried out and how it connects to the DSR project as a whole. The execution of the LI involves activities related to the referred empirical study, which include defining the research protocol to be followed, running it to collect and analyze data, and recording the study results. The LI is concluded with a report that connects the study results with the DSR project as a whole.

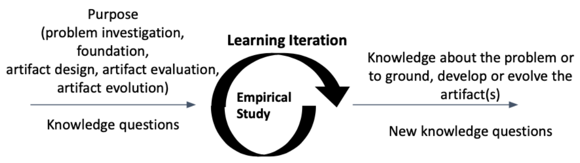
Purpose (problem investigation, foundation, artifact design, artifact evaluation, artifact evolution)
Knowledge questions
Learning Iteration
Empirical Study
Knowledge about the problem or to ground, develop or evolve the artifact(s)
New knowledge questions

Figure 2: Learning Iteration

Back to Figure 1, in the Relevance cycle, the researcher defines the problem to be addressed in the DSR project, the goal to be achieved, the requirements that the proposed artifact should meet and the acceptance criteria that will be used to evaluate the artifact. In this cycle, LIs are used to learn about the problem, the solution requirements and the application domain (i.e., the environment where the artifact will be applied). In this sense, knowledge questions to be answered in the studies aim to get to know the problem (e.g., What is the problem? Why does it happen? When? Who/what does it affect? Which are its effects/consequences? Was the problem treated before? Which treatments have been applied? Where have they succeeded? Where have they failed?), identify the research requirements (e.g., What is necessary in the artifact, so that it will be able to solve the problem? Which are the stakeholders' needs? Which are the technical constraints? Which are the acceptance criteria?) and understand the environment (e.g., Which are the stakeholders? What are the domain characteristics? What are the main constraints and rules?).

In the Rigor cycle, the researcher selects knowledge available in the knowledge base and that will help him/her conduct the research project (e.g., by providing information about which techniques and methods can be applied and how to apply them). Moreover, such knowledge is also useful to ground the artifact to be developed. In this sense, the researcher can perform studies to learn about the artifact's foundations and also to identify methods, techniques, theories, tools, etc. that can be applied so that the artifact meets the established requirements. Studies performed in this cycle aim to answer knowledge questions such as Which existing methods, theories, tools, etc. are suitable for supporting the research project execution? How to apply them?

In the Design cycle, the researcher designs, builds, evaluates and evolves the aimed artifact. This cycle coincides with the design part of DSR referred by Wieringa [48]. Although the focus is on design, we can also have knowledge questions related to it. Thus, studies performed in this cycle seek to provide knowledge about the artifact itself. In other words, in this cycle, the studies are LIs that aim to grow knowledge about the artifact in order to produce a sound and suitable solution. First, the researcher designs the artifact. In this context, he/she wants to learn about how to meet the established requirements. Thus, he/she needs to answer knowledge questions such as How to meet the requirements? How to address the constraints? How to evaluate if the acceptance criteria are met?. Once the artifact is built, the knowledge needs are related to its use. Hence, the researcher performs studies to answer questions like Does the artifact work in context? Does the artifact meet the acceptance criteria? Which does and which does not work properly? The artifact can, thus, evolve. Studies aiming at artifact evolution aim to provide knowledge about improvement opportunities. In this sense, the researcher needs to learn about what can be improved in the artifact and how to make the improvements. This involves knowledge questions such as Which improvements are needed? How to improve the artifact? Can the artifact be adapted to a new context? Which modifications should be made?

As argued by Hevner [15], the three cycles are closely related. Therefore, some studies provide knowledge for more than one cycle. For example, in the Rigor cycle, the researcher can perform a study to identify findings to help meet the requirements established in the Relevance cycle. The acquired knowledge is relevant for designing the artifact, which occurs in the Design cycle.

By performing the studies as LIs, the researcher is guided by knowledge needs and grows knowledge towards the development of the artifact in context. Moreover, LIs help harmonize the use of empirical studies in DSR projects by organizing and combining several studies towards a common goal – the DSR project goal. Table 1 summarizes the LIs categories, the cycle where they occur, some of the general knowledge questions the studies are aimed to answer and some examples of empirical studies that can help answer the knowledge questions. In the next section, we present two DSR projects that adopted empirical studies as LIs.

4 APPLYING THE PROPOSED APPROACH

In this section, we briefly share our experience of using LIs in two DSR projects related to Software Quality. The first one regards an artifact to aid organizations in IT service measurement and its further extension to address service level agreement (SLA). The second one proposes an artifact to support the defense of software estimates when estimators face pressure over them, which can result in unrealistic commitments and affect software quality. In these works, empirical studies were performed iteratively, based on knowledge needs and towards the aimed artifact.

4.1 SINIS - A Framework for Selecting Indicators for IT Services

SINIS (Select Indicators for IT Services) was created to support IT service organizations to define indicators in multiple levels and aligned to the business goals [44]. It was conceived using DSR and through nine LIs. The first iteration was performed in the Relevance cycle and aimed *Problem Investigation* (i.e., it held to learn about the problem). It comprised an exploratory case study on industry to investigate the process of defining IT service indicators [40]. From this iteration, it was possible to learn that although indicators were often defined aligned to the business goals, there was not a formal process to identify the proper indicators to be used, neither to cascade and keep track of the business goals into indicators at different organizational levels. Moreover, data collection was often executed by only one person (who had intrinsically owned the process), indicators were not properly defined, many of those could not be automatically measured, demanded manual observation, and were susceptible to subjective interpretation. Hence, there was a need to improve the process to define IT service indicators.

Once the problem was understood, a second LI was performed to obtain knowledge to *Artifact Foundation*. The iteration consisted of a mapping study that investigated IT service indicators recorded in the literature [42]. From this iteration, the researchers learned that there is a set of IT service indicators available in the literature but many of them have incomplete definitions. In this way, although the study was performed in the Rigor cycle, its results also provided useful knowledge to refine the problem in the Relevance cycle. By knowing that many IT service indicators were not defined in a complete and unambiguous way, the researchers added a new requirement to be met by the aimed artifact (it should support defining IT service indicators clearly). Besides the acquired knowledge, this iteration added a catalog of indicators to the knowledge base (i.e., a new contribution, which other researchers can use).

The second LI rose a new knowledge question about the use of the found IT service indicators, which was addressed in the third LI. It comprised a new case study that investigated if the indicators found in the literature would be helpful to IT service industry [41]. In the study, managers were provided with the catalog of indicators (resulting from the second iteration) to select the ones they found useful. The researchers learned that the catalog was helpful, but the managers still needed a detailed process to select indicators. The study was performed in the Relevance cycle and supported the researchers in *Problem Investigation* by consolidating knowledge about the problem and helping delineate the research goal, requirements and preliminary acceptance criteria.

Aiming at *Artifact Design*, three studies were carried out in the industry in the Design cycle to apply, learn, and receive feedback about design choices made to develop the artifact. In the fourth iteration, we learned that business process modeling and process mining could be used to support finding the sub-process that acted as a bottleneck of incident resolution and impacted the business goals, to then define indicators to control that specific sub-process and strategies to improve them [46]. In the fifth iteration we learned to apply GQM+Strategies [5] to support the definition of goals, indicators and strategies at different organization levels, while keeping track of the connection between the multi-level indicators [39]. In the sixth iteration, we evolved the idea from the fourth study of investigating the root cause analysis. We learned how to use the cause-and-effect diagram and the 5-why's technique to help cascade the problems being faced to achieve business goals into actionable strategies (and respective indicators) that could help to mitigate or solve those business problems achieved [45]. Each of these three studies aimed to verify if the design choice made was adequate and contributed to meeting the established requirements. In this way, besides providing useful knowledge for developing the artifact, the acquired knowledge also enabled us to review both requirements and acceptance criteria and improve the artifact foundation, since the design choices considered knowledge available in the knowledge base (e.g., knowledge about cause-effect diagram).

Table 1: Summary of Learning Iterations

Category	Cycle	General Knowledge Questions	Examples of Empirical Studies
Problem Investigation	Relevance	- What is the problem? Why does it happen? When? Who/what does it affect? Which are its effects/consequences? Was the problem treated before? Which treatments have been applied? Where have they succeeded? Where have they failed? - Which are the stakeholders? What are the domain characteristics? What are the main constraints? - What is necessary in the artifact, so that it will be able to solve the problem? Which are the stakeholders needs? Which are the technical constraints? Which are the acceptance criteria?	- A SLM to investigate a research topic and identify gaps that the current solutions are not able to cover. - A survey to identify problems in the application domain. - A survey to identify stakeholders' requirements. - An exploratory study to get to know the application domain. - An exploratory study to foster the understanding of how practitioners perceive or deal with pitfalls and other findings from a SLM.
Artifact Foundation	Rigor	- Which methods, theories, tools, etc. are suitable for for supporting the research execution? How to apply them?	- A SLM to investigate which methods, tools, techniques, theories, etc. are already available in the literature and are suitable for supporting the execution of the research project. - A controlled experiment or an exploratory study to investigate if a particular method, tool, technique, theory, etc. is suitable for the research project.
Artifact Design	Design	- How to meet the requirements? How to address the constraints? How to evaluate if the acceptance criteria are met? - How to package in an artifact the solutions to address different stakeholders' needs?	- A case study or a controlled experiment to verify if a particular design decision meets some of the artifact requirements. - An action research to define an approach to tackle the stakeholders' requirements while proposing the first version of the artifact.
Artifact Evaluation	Design	- Does the artifact work in context? Does the artifact meet the acceptance criteria? Which does and which does not work properly?	- A case study to evaluate the use of the artifact in the application domain.
Artifact Evolution	Design	- Which improvements are needed? How to improve the artifact? Can the artifact be adapted to a new context? Which modifications should be made?	- A survey with the artifact users to identify improvement opportunities. - A controlled experiment to prioritize improvement opportunities.

Taking the knowledge acquired over all the LIs into account, the artifact (SINIS) was developed and a participative case study was performed in the seventh LI [43] aiming at *Artifact Evaluation*. In this iteration, SINIS was used in industry and evaluated in context, bringing lessons learned as knowledge to improve the artifact.

After being applied in context, we noticed that SINIS could be extended to satisfy new requirements. Thus, we performed two studies in the eighth and ninth LIs for *Artifact Evolution*. In the eighth LI, we applied SINIS and evaluated whether it was suitable for supporting the definition of goals, indicators and strategies of IT Information Security to the business goals. In the ninth iteration, we used SINIS in another participative case study [7] to identify and explain the relationship between business goals, indicators, and strategies in a large public IT service organization while evaluating how the framework could be extended to consider SLAs alignment explicitly [12]. Both studies provided us valuable insights on how the framework could be extended to new contexts besides confirming its capacity of supporting IT business alignment. The later study led to the creation of a new version of SINIS [13], which comprised other LIs as well. The seventh, eighth and ninth LIs were based on participative case studies because the researchers were members of the organizations, observed a particular group of organization' subjects, and participated in the process being observed [7].

Figure 3 summarizes the LIs performed in this DSR project. For simplification and better visualization, in the figure we represent some studies in a single iteration.

4.2 SwEDeL - A Set of Defense Lenses for Software Estimates

SwEDeL (Software Estimates' Defense Lenses) comprises a guide in the form of lenses to support the defense of software estimates when estimators face pressure over them [29]. The artifact was built through the use of five LIs [26]. At the beginning of the project, the research overall interest was in the problem of distortions of software estimates: changes (increases or decreases) of software estimates that make them deviate from the most likely usage of effort for a given task or project, given the available information [25]. More specifically, we were interested in intentional distortions, i. e., changes to estimates to fulfill objectives outside the estimation context [25], such as to attain business goals when estimates collide with them [9], leading to unrealistic commitments. Therefore, we also started the project with an overall goal in mind: supporting estimators in reaching realistic commitments. We performed the first two LIs in parallel in the Relevance cycle with the main purpose of *Problem Investigation*, to elaborate more on the problem and understand more of our environment from the practitioners' and researchers' perspectives.

The first LI aimed to investigate the problem from the practitioners' perspective. It consisted of a case study to answer the knowledge question: how are software estimates used to establish software development commitments in the software industry? We approached the problem indirectly, investigating how pressure over software estimates can emerge in the process and its results regarding changes to estimates [28]. This study contributed with confirming evidence about the existence of changes to software estimates. It also enlightened us that instead of defending their estimates, software practitioners were padding some of their tasks to compensate for the pressure over other tasks that resulted in tight deadlines and decreases of software estimates (i.e., intentional distortions) [28]—thus connecting with our overall problem.

The second LI was a SLM [27] guided by the knowledge question: What are the factors affecting expert-judgment software estimates?

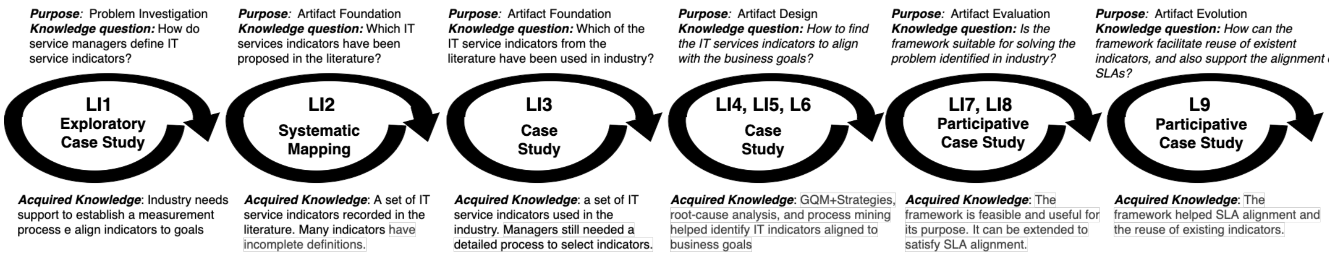
Purpose: Problem Investigation
Knowledge question: How do service managers define IT service indicators?
LI1
Exploratory Case Study
Acquired Knowledge: Industry needs support to establish a measurement process e align indicators to goals
Purpose: Artifact Foundation
Knowledge question: Which IT services indicators have been proposed in the literature?
LI2
Systematic Mapping
Acquired Knowledge: A set of IT service indicators recorded in the literature. Many indicators have incomplete definitions.
Purpose: Artifact Foundation
Knowledge question: Which of the IT service indicators from the literature have been used in industry?
LI3
Case Study
Acquired Knowledge: a set of IT service indicators used in the industry. Managers still needed a detailed process to select indicators.
Purpose: Artifact Design
Knowledge question: How to find the IT services indicators to align with the business goals?
LI4, LI5, L6
Case Study
Acquired Knowledge: GQM+Strategies, root-cause analysis, and process mining helped identify IT indicators aligned to business goals
Purpose: Artifact Evaluation
Knowledge question: Is the framework suitable for solving the problem identified in industry?
LI7, LI8
Participative Case Study
Acquired Knowledge: The framework is feasible and useful for its purpose. It can be extended to satisfy SLA alignment.
Purpose: Artifact Evolution
Knowledge question: How can the framework facilitate reuse of existent indicators, and also support the alignment of SLAs?
L9
Participative Case Study
Acquired Knowledge: The framework helped SLA alignment and the reuse of existing indicators.

Figure 3: Learning Iterations to build SINIS

This LI served two different purposes. First, it helped us in *Problem Investigation* by focusing on the literature. Second, it allowed us to systematically grow the *Artifact Foundation* by identifying other factors related to pressure and, possibly, approaches already attempted by other researchers. The study resulted in a broad and well-organized map that readers can use to navigate through the factors affecting software estimates, along with measurement strategies to understand the impact of such factors, the research strategies used to investigate them, among other things [27]. Thus, we evolved the research problem: estimators change their estimates and accept unrealistic commitments to deal with human and social issues in software estimation—e.g., resistance of other stakeholders to update early estimates, pressure to decrease estimates, existence of business goals, among others.

Together, these first two iterations enabled us to update the research goal: empower software estimators in defending estimates and support them to reach realistic commitments—instead of yielding to pressure either by changing their estimates or padding other tasks. Furthermore, inspired by the results of these iterations and previous literature suggesting that negotiation could be used to defend software estimates [30], we raised a new knowledge question: what negotiation theories and methods are relevant and can be adapted to the software estimation context?

Even before finishing the SLM, we started the third LI, which took place in the Rigor cycle. We focused on gaining knowledge for the *Artifact Foundation*, to help us devise a solution to aid software practitioners to deal with pressure over their estimates. We studied theories about negotiation styles and methods, as this was a promising idea proposed earlier, although not explored to its full extent —for instance, it was never empirically evaluated. Thus, we enlarged our foundations with knowledge from disciplines other than Software Engineering. We employed a snowballing strategy to find relevant literature [26].

Based on the acquired knowledge, in the Design cycle we developed the first version of SwEDeL, a guide consisting of a set of defense lenses to change the perspective of software practitioners about how to react in face of pressure over software estimates: defending them, instead of yielding to pressure [29]. Thus, we performed the fourth LI, a judgment study [38] applying a focus group with software practitioners [29]. The purpose was *Artifact Evaluation*, with the knowledge question: What is the perceived usefulness of the artifact? We were also interested in *Artifact Evolution*, so we also asked: What are the improvement opportunities for the artifact? The acquired knowledge allowed us to generate a second version of SwEDeL. This iteration also highlighted an improvement opportunity: the presentation of the lenses as part of a guide was time-consuming and not attractive to some participants. This led us to another relevant design decision: to build a complementary artifact, in the form of a digital simulation, to help practitioners to develop the negotiation skills embedded in the lenses in a more dynamic way. The digital simulation was implemented as interactive videos that present each one of the defense lenses in the context of pressure scenarios (identified from the knowledge acquired in the first two LIs), thus making more concrete the application of the negotiation principles in real-life situations.

Currently, we are executing the fifth LI: one more empirical study for *Artifact Evaluation*. It is a controlled experiment with industry practitioners to assess the impact of the digital simulation, focusing on two knowledge questions: (i) What is the impact of the artifact on practitioners' intentions to defend their software estimates? and (ii) What is the perceived usefulness of the artifact? Additionally, we were again interested in *Artifact Evolution*, and the knowledge question: What are the improvement opportunities? As part of this iteration, we enlarged our foundations once more. We included knowledge of the Theory of Planned Behavior [19], which states that intentions are the immediate antecedent of a behavior of interest [2]—in our case, the behavior is the defense of the software estimates. We intend to compare participants from one experimental group—exposed to the digital simulation—to participants in a control group—exposed to questions to make them reflect on the pressure scenarios they face over their estimates and how they impact relevant outcomes, such as product quality, overtime work, and others. Therefore, we expect to learn whether digital simulation is a better alternative to the simpler solution of making people reflect.

Figure 4 gives an overview of the DSR project, summarizing information about the problem, research goal, requirements and acceptance criteria, proposed artifact, main foundations and contributions of the research, and the five performed LIs.

5 DISCUSSION

The Software Engineering research community has a history of adapting research methods, methodologies, and paradigms from other disciplines. This is also perceived in Software Quality research, which has applied such approaches and also provided advances to better adapt them to our field. For instance, on an analogy with Evidence-Based Medicine, SLRs have been used in Software Engineering as a method for the integration of evidence from multiple primary studies [23]. However, the adoption took time and required

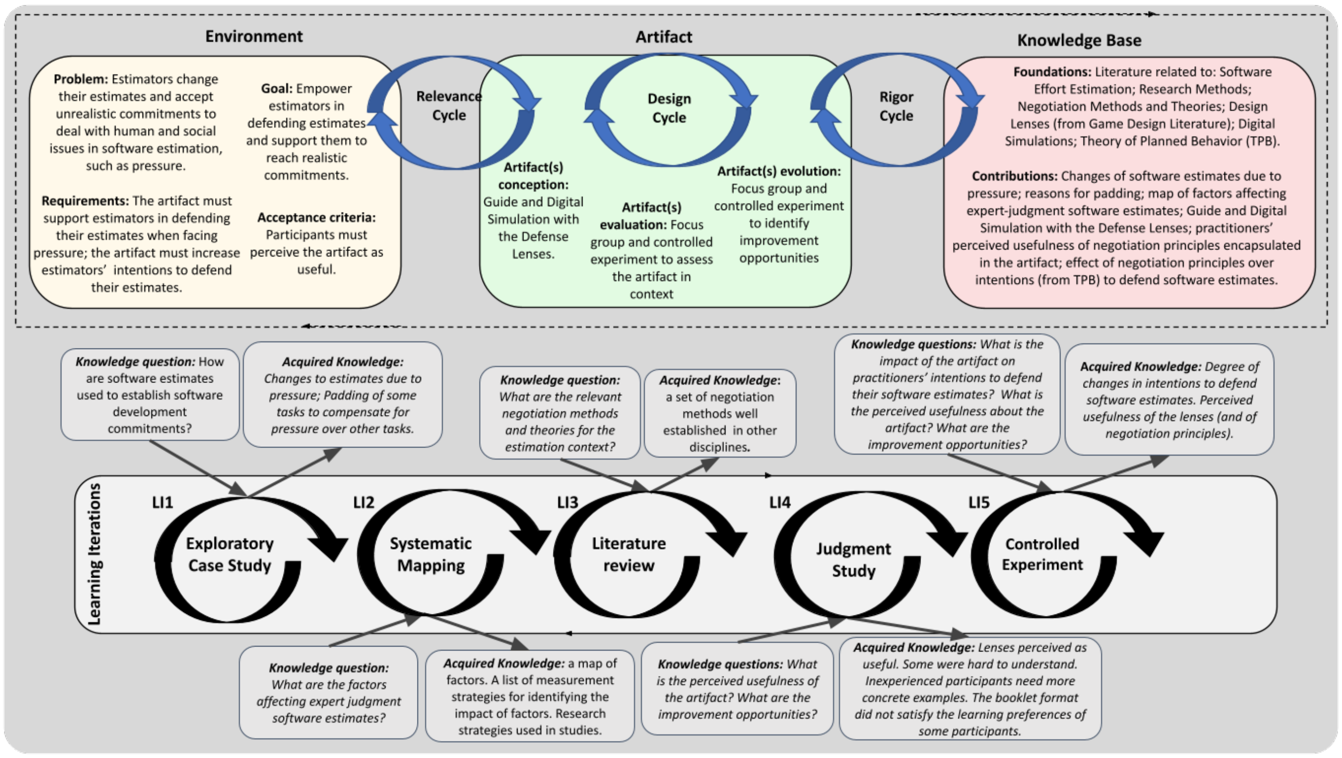
Environment
Artifact
Knowledge Base
Problem: Estimators change their estimates and accept unrealistic commitments to deal with human and social issues in software estimation, such as pressure.
Goal: Empower estimators in defending estimates and support them to reach realistic commitments.
Requirements: The artifact must support estimators in defending their estimates when facing pressure; the artifact must increase estimators' intentions to defend their estimates.
Acceptance criteria: Participants must perceive the artifact as useful.
Relevance Cycle
Design Cycle
Rigor Cycle
Artifact(s) conception: Guide and Digital Simulation with the Defense Lenses.
Artifact(s) evaluation: Focus group and controlled experiment to assess the artifact in context
Artifact(s) evolution: Focus group and controlled experiment to identify improvement opportunities
Foundations: Literature related to: Software Effort Estimation; Research Methods; Negotiation Methods and Theories; Design Lenses (from Game Design Literature); Digital Simulations; Theory of Planned Behavior (TPB).
Contributions: Changes of software estimates due to pressure; reasons for padding; map of factors affecting expert-judgment software estimates; Guide and Digital Simulation with the Defense Lenses; practitioners' perceived usefulness of negotiation principles encapsulated in the artifact; effect of negotiation principles over intentions (from TPB) to defend software estimates.
Knowledge question: How are software estimates used to establish software development commitments?
Acquired Knowledge: Changes to estimates due to pressure; Padding of some tasks to compensate for pressure over other tasks.
Knowledge question: What are the relevant negotiation methods and theories for the estimation context?
Acquired Knowledge: a set of negotiation methods well established in other disciplines.
Knowledge questions: What is the impact of the artifact on practitioners' intentions to defend their software estimates? What is the perceived usefulness about the artifact? What are the improvement opportunities?
Acquired Knowledge: Degree of changes in intentions to defend software estimates. Perceived usefulness of the lenses (and of negotiation principles).
Learning Iterations
LI1
Exploratory Case Study
LI2
Systematic Mapping
LI3
Literature review
LI4
Judgment Study
LI5
Controlled Experiment
Knowledge question: What are the factors affecting expert judgment software estimates?
Acquired Knowledge: a map of factors. A list of measurement strategies for identifying the impact of factors. Research strategies used in studies.
Knowledge questions: What is the perceived usefulness of the artifact? What are the improvement opportunities?
Acquired Knowledge: Lenses perceived as useful. Some were hard to understand. Inexperienced participants need more concrete examples. The booklet format did not satisfy the learning preferences of some participants.

Figure 4: Overview of the DSR project and Learning Iterations to build SwEDeL

refinements. The main guidelines on SLRs were first written in 2004 [21] and were revised in the following years, such as in 2007 [20] and 2015 [22]. It also required adaptations along the way. For instance, the current research practice in our field led to the definition of specific procedures for SLMs [33], which has been more used than SLRs [1]. Researchers also are increasingly using gray literature in their reviews [18], given the importance of the practitioners' voices in our field.

Likewise, we also need refinement and adaptation of DSR, as we acknowledge it as a frame for Empirical Software Engineering [35] and empirical studies as a valuable means to advance in software quality issues. Runeson et al. [35] propose a few refinements, such as the use of technological rules, i.e., statements that capture knowledge about mappings between instances of problems and solutions, helping articulate knowledge produced during a DSR project. We argue that another relevant refinement is a better understanding of how empirical studies can be used in a combined and harmonized way throughout the DSR cycles [15], contributing to the researcher continuously *learn* about the research problem and proposed solution. In this sense, our approach provides guidance and examples on how to use empirical studies as LIs in DSR projects. We must clarify that we do not mean to replace current DSR guidelines but enrich them by guiding how to position empirical studies in DSR projects. Like in [48] and [31], in our approach, empirical studies are part of the DSR project, but now they are more explicit, categorized and are defined according to the knowledge they intend to capture. Therefore, we make explicit knowledge about how to position empirical studies throughout a DSR project. We have noticed that such knowledge can be specially relevant for novice researchers, who might find it difficult to understand how to integrate different studies in the DSR framework. Based on our experience, many Ph.D. students fail to understand how the Relevance, Rigor and Design cycles should be conducted. The proposed structure of LIs, guided by knowledge questions, helps novice researchers to plan their research.

Moreover, the Software Quality community has sought to foster academia and industry collaboration (e.g., by exploring the use of Action Research) [4]. DSR is a solution-oriented research methodology [50] to add to the Software Quality researchers' toolbox to increase research relevance through collaboration with the software industry. Research relevance should not be aimed at the expense of research rigor. Hence, we argue that using LIs in DSR projects aids researchers in keeping both relevance and rigor. This happens because each LI acts as a unitary gear, gathering a minimal set of elements necessary to make DSR projects' cycles move forward in a scientific manner: a set of knowledge questions, one empirical study to answer them, and new knowledge acquired about either the problem or the solution—also, potentially new knowledge questions to be answered in coming LIs. This unitary gear metaphor also helps the researcher navigate the nested problem-solving hierarchy present in DSR projects, where a higher level practical problem can be decomposed in many other practical or knowledge problems (also called knowledge questions) [47]. By planning and monitoring the DSR project from the perspective of the needed LIs to solve the higher-level practice problem, the researcher can experience the research process more smoothly, instead of getting entangled in the nested structure. This happens because the LIs make the researcher reflect on the necessary knowledge and connect his/her questions to the aimed knowledge throughout the project.

6 FINAL CONSIDERATIONS

The concern with the need for more industry-academia collaboration to improve the research relevance has increased. In fact, the collaboration between industry and academia is by many viewed as essential [36]. One challenge in such collaboration is identifying a suitable research approach that provides credible evidence [49]. Solution-oriented research methodologies can be useful in this matter. DSR has stood out as a research approach conducted in the context of practice to generate contributions to practice [50].

DSR stresses the research to be built on existing knowledge or theory, and create/generate/synthesize design knowledge. This duality is represented in Hevner's cycles [15] and it is put forward by Wieringa [48], who summarizes DSR as an 'act of producing knowledge by designing useful things' [50]. DSR is a research method strongly based on knowledge.

Although DSR has been used in many research projects in Software Engineering as a whole and Software Quality in particular, there is still a need for growing knowledge about how DSR can be applied in this context. In this paper, we aimed to give a step in this direction by sharing knowledge that we have obtained over the last eight years by applying DSR in several projects. In our experience, we have organized empirical studies as LIs based on knowledge questions that guide the researcher toward the development of the aimed artifact. For that, the researcher identifies his/her knowledge needs and the purpose the knowledge is intended to fulfill. Thus, he/she selects the empirical study he/she considers more suitable for answering the knowledge questions, performs it and uses the acquired knowledge to advance the research. In this way, it is possible to harmonize several empirical studies and leverage experimentation in DSR projects by using such studies not only to evaluate the artifact but also to understand the problem, establish the research goal, requirements and acceptance criteria, as well as improve the artifact foundations. The researcher should follow the approach until the aimed artifact is produced and the DSR project goal is achieved. If there are resource or time constraints, the artifact produced in a DSR project can be further evolved in a new DSR project.

Over the last eight years, we have conducted several projects by organizing empirical studies as LIs in DSR projects focused on increasing the quality of software products or processes. From this experience, we can point out some benefits and challenges. As for the benefits, we observed that by organizing empirical studies as LIs in DSR projects we can experience a more fluid process because the researcher reflects on the necessary knowledge and connects his/her questions to the aimed knowledge throughout the project. We also noticed that our approach encourages the researcher to better understand the problem and better ground the artifact before developing it. Moreover, the researcher is stimulated to evaluate the design choices iteratively until the artifact is ready. This contributes to evolving the artifact gradually, making decisions to ensure that it is suitable for the purpose it intends to fulfill. Finally, the researcher can improve the artifact based on lessons learned from its use in context and can further evolve it into a new version, which increases the capacities of the previous one.

As for the challenges, we noticed that it is necessary to reflect on the number of LIs to be performed in the project because it must fit the available time and resources. Moreover, DSR projects involve a lot of tacit knowledge. The LI categories and suggested knowledge questions help decide on which studies to carry out, but there are other issues that need to be better supported (e.g., how to instantiate the knowledge questions to a particular project, how to establish the best order to perform the studies, how to prioritize studies according to time and effort constraints, when performing studies in parallel). In this sense, we believe that a set of detailed guidelines on how to apply our approach can be useful for other researchers.

Our purpose with this paper is to introduce and exemplify the idea of using empirical studies as LIs in DSR projects. This is our first step toward aiding other researchers in applying DSR in Software Quality and Software Engineering research. There is a lot of tacit knowledge that still needs to be addressed. Thus, in future work, we intend to elaborate guidelines on how to organize and conduct empirical studies as LIs in DSR projects, provide some templates, and make them available for other researchers in a Wiki. We will also conduct other projects using our approach to improve it, bring more examples and enrich the guidelines.

Some limitations should be considered together with the experience shared in this paper. At least one of the authors has participated in the projects conducted using our approach. This may have influenced the perceived results. Moreover, the research projects involved researchers from only three different research groups, which is a small sample. Thus, the results are based on our perceptions of using the approach. To address this limitation, we intend to perform qualitative evaluation, through interviews and observation with different researchers using the approach.

ACKNOWLEDGMENTS

This work is supported by CAPES - Financing Code 001, FAPERJ (210.231/2021, 211.437/2021), UNIRIO (PPQ-UNIRIO 02/2021), CNPq (314174/2020-6), FAPEAM (062.00150/2020), and grant #2020/05191-2 São Paulo Research Foundation (FAPESP). The present work also is the result of the Research and Development (R&D) project 001/2020, signed with Federal University of Amazonas and FAEPI, Brazil, which has funding from Samsung, using resources from the Informatics Law for the Western Amazon (Federal Law nº 8.387/1991), and its disclosure is in accordance with article 39 of Decree No. 10.521/2020.

REFERENCES

[1] 2022. Short communication: Evolution of secondary studies in software engineering. *Inf Softw Technol* 145 (2022), 106840.
[2] Icek Ajzen. 2020. The theory of planned behavior: Frequently asked questions. *Human Behavior and Emerging Technologies* 2, 4 (2020), 314–324. https://doi.org/10.1002/hbe2.195 Preprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/hbe2.195.
[3] J.E. Aken, van. 2004. Management research based on the paradigm of the design sciences: the quest for field-tested and grounded technological rules. *J. Manag. Stud.* 41, 2 (2004), 219–246. https://doi.org/10.1111/j.1467-6486.2004.00430.x
[4] Aline Barbosa, Geraldo Galindo, Maria Lencastre, Roberta Fagundes, and Wylliams Santos. 2020. Fostering Industry-Academia Collaboration in Software

Engineering using Action Research: A Case Study. In *Anais do XIX Simpósio Brasileiro de Qualidade de Software* (São Luiz do Maranhão). SBC, Porto Alegre, RS, Brasil, 411–419. https://sol.sbc.org.br/index.php/sbqs/view/14239

[5] Victor Basili, Adam Trendowicz, Martin Kowalczyk, Jens Heidrich, Carolyn Seaman, Jürgen Münch, and Dieter Rombach. 2014. GQM+ Strategies in a Nutshell. In *Aligning Organizations Through Measurement*. Springer, 9–17.

[6] Richard Baskerville. 2008. What design science is not. *European Journal of Information Systems* 17, 5 (2008), 441–443. https://doi.org/10.1057/ejis.2008.45

[7] Richard L Baskerville. 1997. Distinguishing action research from participative case studies. *Journal of systems and information technology* (1997).

[8] Joelma Choma, Luciana A. M. Zaina, and Tiago Silva Da Silva. 2015. Towards an Approach Matching CMD and DSR to Improve the Academia-Industry Software Development Partnership: A Case of Agile and UX Integration. In *Proceedings of the 29th Brazilian Symposium on Software Engineering*. IEEE Computer Society, USA, 51–60. https://doi.org/10.1109/SBES.2015.18

[9] Aldo Dagnino. 2013. Estimating software-intensive projects in the absence of historical data. In *2013 35th International Conference on Software Engineering (ICSE)*. 941–950. https://doi.org/10.1109/ICSE.2013.6606643 ISSN: 1558-1225.

[10] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. *Selecting Empirical Methods for Software Engineering Research*. Springer London, London, 285–311. https://doi.org/10.1007/978-1-84800-044-5_11

[11] Michael Felderer and Guilherme Horta Travassos. 2020. *The Evolution of Empirical Methods in Software Engineering*. Springer International Publishing, Cham, 1–24. https://doi.org/10.1007/978-3-030-32489-6_1

[12] Eduardo Ferreira, Bianca Trinkenreich, Monalessa Perini Barcellos, and Gleison Santos. 2018. Using SINIS and GQM+Strategies to Align Organizational Goals and Service Level Agreement Indicators. In *Proceedings of the 17th Brazilian Symposium on Software Quality*. ACM, New York, NY, USA, 324–333. https://doi.org/10.1145/3275245.3275274

[13] Eduardo Ferreira, Bianca Trinkenreich, Monalessa Perini Barcellos, and Gleison Santos. 2021. SINIS-LA Method for IT Alignment Considering Service Level Management. In *XVII Brazilian Symposium on Information Systems* (Uberlândia, Brazil) *(SBSI 2021)*. ACM, New York, NY, USA. https://doi.org/10.1145/3466933.3466977

[14] Tony Gorschek and Krzysztof Wnuk. 2020. *Third Generation Industrial Co-production in Software Engineering*. Springer Int. Publishing, Cham, 503–525.

[15] A. R. Hevner. 2007. A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems* (2007), 87–92.

[16] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design Science in Information Systems Research. *MIS Quarterly* 28, 1 (mar 2004), 75–105.

[17] Pertti Järvinen. 2007. Action Research is Similar to Design Science. *Quality & Quantity* 41 (2007), 37–54. Issue 1. https://doi.org/10.1007/s11135-005-5427-1

[18] Fernando Kamei, Igor Wiese, Gustavo Pinto, Márcio Ribeiro, and Sérgio Soares. 2020. On the Use of Grey Literature: A Survey with the Brazilian Software Engineering Research Community. In *Proc. of the 34th Brazilian Symposium on Software Engineering (SBES '20)*. ACM, New York, NY, USA, 183–192.

[19] Matthew P. H. Kan and Leandre R. Fabrigar. 2017. Theory of Planned Behavior. In *Encyclopedia of Personality and Individual Differences*, Virgil Zeigler-Hill and Todd K. Shackelford (Eds.). Springer International Publishing, Cham, 1–8. https://doi.org/10.1007/978-3-319-28099-8_1191-1

[20] Staffs Keele et al. 2007. *Guidelines for performing systematic literature reviews in software engineering*. Technical Report EBSE-2007-01. Keele University & University of Durham.

[21] Barbara A. Kitchenham. 2004. *Procedures for Performing Systematic Reviews*. Joint Technical Report TR/SE-0401. Keele University & National ICT Australia Ltd., Keele, UK.

[22] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. 2015. *Evidence-Based Software Engineering and Systematic Reviews* (1 ed.). CRC Press.

[23] Barbara A. Kitchenham, Tore Dyba, and Magne Jorgensen. 2004. Evidence-Based Software Engineering. In *Proceedings of the 26th International Conference on Software Engineering (ICSE '04)*. IEEE Computer Society, USA, 273–281.

[24] Sômulo Nogueira Mafra, Rafael Ferreira Barcelos, and Guilherme Horta Travassos. 2006. Aplicando uma metodologia baseada em evidência na definição de novas tecnologias de software. In *Proceedings of the 20th Brazilian Symposium on Software Engineering (SBES 2006)*, Vol. 1. 239–254.

[25] Ana Magazinius, Sofia Börjesson, and Robert Feldt. 2012. Investigating Intentional Distortions in Software Cost Estimation - An Exploratory Study. *J Syst Softw* 85, 8 (aug 2012), 1770–1781. https://doi.org/10.1016/j.jss.2012.03.026

[26] Patricia Matsubara. 2019. Dealing with Software Estimates Distortions from the Perspective of Negotiation Theories. *SIGSOFT Softw. Eng. Notes* 44, 3 (nov 2019), 22. https://doi.org/10.1145/3356773.3356794

[27] Patricia Matsubara, Bruno Gadelha, Igor Steinmacher, and Tayana Conte. 2022. SEXTAMT: A systematic map to navigate the wide seas of factors affecting expert judgment software estimates. *J Syst Softw* 185 (2022), 111148. https://doi.org/10.1016/j.jss.2021.111148

[28] Patricia Matsubara, Igor Steinmacher, Bruno Gadelha, and Tayana Conte. 2021. Buying time in software development: how estimates become commitments?. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, Madrid, Spain, 61–70. https://doi.org/10.1109/CHASE52884.2021.00015

[29] Patricia Matsubara, Igor Steinmacher, Bruno Gadelha, and Tayana Conte. 2022. The best defense is a good defense: adapting negotiation methods for tackling pressure over software project estimates. In *Proceedings of the 44th International Conference in Software Engineering — New Ideas and Emerging Results Track*. IEEE, Pittsburgh, Pennsylvania. https://doi.org/10.1145/3510455.3512775

[30] Steve McConnell. 2006. Politics, Negotiation, and Problem Solving. In *Software Estimation: Demystifying the Black Art*. Microsoft Press, Redmond, 259–270.

[31] Philipp Offermann, Olga Levina, Marten Schönherr, and Udo Bub. 2009. Outline of a Design Science Research Process. In *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology* (Philadelphia, Pennsylvania) *(DESRIST '09)*. ACM, New York, NY, USA. https://doi.org/10.1145/1555619.1555629

[32] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. 2007. A Design Science Research Methodology for Information Systems Research. *J Manag Inf Syst* 24, 3 (2007), 45–77. https://doi.org/10.2753/MIS0742-1222240302

[33] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. 2015. Guidelines for Conducting Systematic Mapping Studies in Software Engineering. *Inf. Softw. Technol.* 64, C (aug 2015), 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

[34] Rodrigo Rebouças de Almeida, Christoph Treude, and Uirá Kulesza. 2019. Tracy: A Business-Driven Technical Debt Prioritization Framework. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 181–185.

[35] Per Runeson, Emelie Engström, and Margaret-Anne Storey. 2020. *The Design Science Paradigm as a Frame for Empirical Software Engineering*. Springer International Publishing, Cham, 127–147. https://doi.org/10.1007/978-3-030-32489-6_5

[36] Per Runeson, Sten Minör, and Johan Svenér. 2014. Get the Cogs in Synch: Time Horizon Aspects of Industry–Academia Collaboration. In *Proceedings of the 2014 International Workshop on Long-Term Industrial Collaboration on Software Engineering* (Vasteras, Sweden) *(WISE '14)*. ACM, New York, NY, USA, 25–28. https://doi.org/10.1145/2647648.2647652

[37] Rafaela Sampaio, Cristina Teles Cerdeiral, and Gleison Santos. 2021. *A Method to Support Continuous Planning at the Team Level*. ACM, New York, NY, USA. https://doi.org/10.1145/3493244.3493257

[38] Klaas-Jan Stol and Brian Fitzgerald. 2020. *Guidelines for Conducting Software Engineering Research*. Springer International Publishing, Cham, 27–62. https://doi.org/10.1007/978-3-030-32489-6_2

[39] Bianca Trinkenreich and Gleison Santos. 2015. Avaliação da Gerência de Incidentes sob a Luz do MR-MPS-SV e Medição para Apoiar a Melhoria da Qualidade do Serviço de TI. In *Proceedings of the 14th Brazilian Symposium on Software Quality*. SBC, 220–227.

[40] Bianca Trinkenreich and Gleison Santos. 2015. Avaliação do Processo de Medição para Serviços de TI em uma Empresa Global à Luz do MR-MPS-SV. *iSys-Brazilian Journal of Information Systems* 8, 2 (2015), 58–77.

[41] Bianca Trinkenreich. and Gleison Santos. 2015. Metrics to Support It Service Maturity Models - A Case Study. In *Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 1: ICEIS,*. INSTICC, SciTePress, 395–403. https://doi.org/10.5220/0005398003950403

[42] Bianca Trinkenreich, Gleison Santos, and Monalessa Perini Barcellos. 2015. Metrics to Support IT Service Maturity Models. In *Proc. 17th Int. Conf. Enterp. Inf. Syst.(ICEIS 2015)*. 1–8.

[43] Bianca Trinkenreich, Gleison Santos, and Monalessa Perini Barcellos. 2015. SINIS: a method to select indicators for IT services. In *International Conference on Product-Focused Software Process Improvement*. Springer, 68–86.

[44] Bianca Trinkenreich, Gleison Santos, and Monalessa Perini Barcellos. 2018. SINIS: A GQM+Strategies-based approach for identifying goals, strategies and indicators for IT services. *Inf Softw Technol* 100 (2018), 147–164.

[45] Bianca Trinkenreich, Gleison Santos, Monalessa Perini Barcellos, and Tayana Conte. 2017. Eliciting strategies for the GQM+ strategies approach in IT service measurement initiatives. In *Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 374–383.

[46] Bianca Trinkenreich, Gleison Santos, Valdemar TF Confort, and Flávia Maria Santoro. 2015. Toward using Business Process Intelligence to Support Incident Management Metrics Selection and Service Improvement. In *SEKE*. 522–527.

[47] Roel Wieringa. 2009. Design Science as Nested Problem Solving. In *Proc. of the 4th Int. Conf. on Design Science Research in Information Systems and Technology* (Philadelphia, Pennsylvania) *(DESRIST '09)*. ACM, New York, NY, USA. https://doi.org/10.1145/1555619.1555630

[48] Roelf J. Wieringa. 2014. *Design science methodology for information systems and software engineering*. Springer, Netherlands. https://doi.org/10.1007/978-3-662-43839-8 10.1007/978-3-662-43839-8.

[49] Claes Wohlin and Austen Rainer. 2021. Challenges and recommendations to publishing and using credible evidence in software engineering. *Inf Softw Technol* 134 (June 2021), 106555. https://doi.org/10.1016/j.infsof.2021.106555

[50] Claes Wohlin and Per Runeson. 2021. Guiding the selection of research methodology in industry–academia collaboration in software engineering. *Inf Softw Technol* 140 (2021), 106678. https://doi.org/10.1016/j.infsof.2021.106678