INSIGHTS ON TRANSFERRING SOFTWARE ENGINEERING SCIENTIFIC
KNOWLEDGE TO PRACTICE

Talita Vieira Ribeiro

Tese de Doutorado apresentada ao Programa de
Pós-Graduação em Engenharia de Sistemas e
Computação, COPPE, da Universidade Federal do
Rio de Janeiro, como parte dos requisitos
necessários à obtenção do título de Doutor em
Engenharia de Sistemas e Computação

Orientadores: Guilherme Horta Travassos

Jeffrey Clark Carver

Rio de Janeiro

September 2022

INSIGHTS ON TRANSFERRING SOFTWARE ENGINEERING SCIENTIFIC
KNOWLEDGE TO PRACTICE

Talita Vieira Ribeiro

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO
LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR
EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Guilherme Horta Travassos

Jeffrey Clark Carver

Examinada por:

_____

Prof. Guilherme Horta Travassos, D.Sc.

_____

Prof. Toacy Cavalcante de Oliveira, D.Sc.

_____

Prof. Manoel Gomes de Mendonça Neto, Ph.D.

_____

Prof. Marcos Kalinowski, D.Sc.

_____

Prof. Sérgio Castelo Branco Soares, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

SETEMBRO DE 2022

REFLEXÕES SOBRE A TRANSFERÊNCIA DE CONHECIMENTO CIENTÍFICO DE ENGENHARIA DE SOFTWARE PARA A PRÁTICA

Talita Vieira Ribeiro

Setembro/2022

Orientadores: Guilherme Horta Travassos

Jeffrey Clark Carver

Programa: Engenharia de Sistemas e Computação

CONTEXTO. Fazer a ponte entre a pesquisa e a prática é um desafio antigo na área de engenharia de software. Enquanto os pesquisadores esperam que os profissionais usem as produções científicas, os profissionais esperam que os pesquisadores lidem com problemas reais enfrentados regularmente na indústria de software. OBJETIVO. Apoiar os pesquisadores na identificação e na condução de pesquisa científica a respeito de questões práticas relevantes capturadas em repositórios de conhecimentos práticos de engenharia de software. Em seguida, apoiar os pesquisadores a tornar o conhecimento científico de engenharia de software disponível para os profissionais de uma forma que eles possam encontrar, entender e avaliar esse conhecimento mais facilmente. MÉTODO. Realizamos uma série de estudos experimentais para identificar as razões da lacuna existente entre a pesquisa e a prática que dificulta a busca e o uso do conhecimento científico pelos profissionais. RESULTADOS. Por meio de experimentação, reunimos evidências sobre as dificuldades que os profissionais encontram ao buscar, entender e avaliar o conhecimento científico da engenharia de software. Com base nos achados, propusemos o Hermes, um apoio para pesquisadores na identificação de questões práticas de engenharia de software que podem ser usadas para conduzir pesquisas práticas na área. O Hermes

compreende oito heurísticas –duas melhoradas a partir de avaliações – para serem usadas durante a realização de pesquisas práticas e uma infraestrutura computacional que lida com tarefas morosas envolvendo a aplicação das heurísticas. CONCLUSÕES. Junto com Hermes, esta tese fornece uma série de reflexões importantes com base em evidências que podem ser usadas em diferentes trabalhos científicos para auxiliar na aproximação da pesquisa e da prática em engenharia de software.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

INSIGHTS ON TRANSFERRING SOFTWARE ENGINEERING SCIENTIFIC KNOWLEDGE TO PRACTICE

Talita Vieira Ribeiro

Setembro/2022

Advisors: Guilherme Horta Travassos

Jeffrey Clark Carver

Department: Computer Science and Systems Engineering

CONTEXT. Bridging the gap between research and practice is an old challenge in the software engineering field. While researchers expect practitioners to use scientific productions, practitioners expect researchers to deal with real problems regularly faced in the software industry. GOAL. To support researchers in identifying and researching relevant practical questions from software engineering practical knowledge repositories. Next, to support researchers in making software engineering scientific knowledge available to practitioners in a way they can easily find, understand, and appraise it. METHOD. We performed a series of empirical studies to identify the reasons for the gap between research and practice that make it difficult for practitioners to look for and use scientific knowledge. RESULTS. Through experimentation, we gathered evidence on practitioners' difficulties when searching, understanding, and appraising software engineering scientific knowledge. Based on the findings, we proposed Hermes, support for researchers in identifying software engineering practical issues that can be used to conduct practical research in the field. Hermes comprises eight heuristics – two improved thorough assessments – to be used while performing practical research and computational infrastructure that deals with time-consuming tasks involving applying the heuristics. CONCLUSIONS. Along with

Hermes, this thesis provides important insights based on evidence that can be used in different works to support bridging the gap between research and practice.

Index

Index of Figures

Index of Tables

1 Introduction

"Bringing science to the people brings people into science." – Stephen
Hawking

1.1 Motivation

While doing research, researchers make assumptions, follow systematic
procedures, and use empirical tools to increase their understanding of phenomena or
events (MCGRATH, 1995). Software engineering (SE) research should not be different
from any other science. Although SE researchers have not had any well-understood
guidance about either the research process they should follow or the standards for
judging the quality of the scientific results in the early years (SHAW, 2002), the SE
community has improved the way it does science in the past years. Many SE
researchers grew their interest in experimentation, and many empirical studies – primary
and secondary ones – have been published in the past years on different SE topics
(BASILI, 2013) (FELDERER and TRAVASSOS, 2020).

Empirical studies have supported evaluating, predicting, understanding,
controlling, and improving software development processes and products for over 30
years (BASILI, SELBY, and HUTCHENS, 1986) (GLASS, VESSEY, and RAMESH,
2002) (FELDERER and TRAVASSOS, 2020). Different software technologies[1] have
been proposed to assist various development process areas, from requirements
elicitation to software deployment. During this time, one of the most important lessons
we learned was the recognition that to achieve high-quality and relevant study results
and proposals, a collaboration between the SE research community and the industry
must exist (SJøBERG, DYBÅ, and JøRGESEN, 2007).

Not only can practice be used to support research, but research can also be used
to support practice. Some academic software technologies have become largely used
in practice from conception until today – not exactly the original conceptions. For
example, object-oriented programming had roots in Simula; a discrete event simulation
language considered the first object-oriented programming language. Simula was
created in the context of a project led by researchers at the Norwegian Computing
Center in the 1960s (DAHL, 2002). Moreover, software quality prediction with software

---

[1] We use software technology to mean "any method, technique, tool, procedure or paradigm used in software
development or maintenance", following the definition of (PFLEEGER, 1999).

metrics is now part of different IDEs and source code analysis tools for various programming languages, and academics first reported them in the 70s (AKIYAMA, 1971) (MCCABE, 1976) (HALSTEAD, 1977). Software design patterns were envisioned in the 80s-90s (BECK and CUNNINGHAM, 1987) (GAMMA *et al.*, 1994) and are examples of academic productions that successfully met practice. More recently, it was the moment for REST, a product from a Ph.D. thesis (FIELDING, 2000), to gain popularity among SE practitioners. Osterweil et al. discuss other impacts of SE research on practice (OSTERWEIL *et al.*, 2008).

This way, "theory leads to practice," and "practice is itself the source of theory," as it happens in many different scientific fields (CHECKLAND, 1985). This way, as researchers, we understand that the disregard for scientific evidence in practice can result in making neither suitable nor applicable technology adoptions to software project needs (RAINER, HALL, and BADDOO, 2003) (DYBÅ, KITCHENHAM, and JøRGENSEN, 2005) (DEVANBU, ZIMMERMANN, and BIRD, 2016). For this reason, the SE community has increased its interest in producing better scientific results for practice (WEYUKER, 2011), thus, guidelines, methods, frameworks, and techniques have been proposed to guide researchers while researching the SE field (BORGES *et al.*, 2015) (MOLLÉRI, PETERSEN, and MENDES, 2018).

SE is difficult to research since its product (software) cannot be manufactured, and its production depends on different human abilities, such as creativity and intellectual sense (BASILI, 2013) that are hard to control. Also, SE research produces both quantitative and qualitative data that are hard to combine. For this reason, most of the research methodologies we use in SE were adapted from other fields to match the SE research context. Therefore, different research methodologies and guidelines were proposed to support the use of scientific methods to research and combine evidence on SE, promoting evidence-based software engineering (EBSE) (MOLLÉRI, PETERSEN, and MENDES, 2018) (FELDERER and TRAVASSOS, 2020) (KITCHENHAM and BUDGEN, 2022).

One of these proposals was the Evidence Factory, computational infrastructure support for the Structured Synthesis Method (SSM), which is a method for evidence aggregation that also comprises a representation of SE evidence (SANTOS, 2015). One of the contributions of SSM and the Evidence Factory to the EBSE was the possibility of aggregating quantitative and qualitative results, producing a synthesis of many different types of studies in SE. The idea behind Evidence Factory is to be a repository of research-synthesized evidence that researchers and practitioners can use further in SE.

The work of Santos (SANTOS, 2015) motivated this research thesis. Santos focused his thesis on the research side of the research-practice collaboration. His work focused on supporting researchers to represent and aggregate different types of scientific knowledge, providing a repository of synthesized SE scientific evidence to be used by researchers and practitioners while performing knowledge translation. We believe that to provide a broader solution for bridging the gap between research and practice, not only scientific knowledge should be combined, but also practical knowledge should be considered in a SE knowledge repository. This thesis complements the work proposed by Santos with the practice side, providing ways of capturing practitioners' needs to be used in research so that research can offer more inline SE knowledge to practice.

1.2 Problem Definition

Despite the mutual benefit, from around 1969 (the early years of SE) until nowadays, bridging the gap between research and practice has still been a challenge in SE (GAROUSI, PETERSEN, and OZKAN, 2016). Nevertheless, several attempts have arisen to exchange knowledge between these two groups during this time. For example, using surveys (PFLEEGER and KITCHENHAM, 2001) with practitioners and ethnography (SHARP, DE SOUZA, and DITTRICH, 2010) in software companies helped researchers understand the real problems faced in software development projects. Likewise, the introduction of design science (HEVNER and CHATTERJEE, 2010) and action-research in software development projects supported practitioners in solving local issues while employing researchers. Moreover, literature review guidelines (KITCHENHAM and BUDGEN, 2022) were created and used in industrial settings to combine different study results and provide insights for software projects. Even so, using these approaches for knowledge exchange is not an everyday practice in SE.

Scientific productions and results are rarely used to support the decision-making of software companies (JEDLITSCHKA, JURISTO, and ROMBACH, 2014), at least in a solo endeavor by practitioners. Still, it is inadequate to state that software engineers make uninformed decisions in the software industry. For example, since its launch in 2008, Stack Overflow (ATWOOD and SPOLSKY, 2008) has been a technological platform for practitioners to ask and answer questions about computer programming, share software development experiences, and promote software technologies. According to Alexa's rank, these facts reflect directly on its popularity – the platform is among the 100 most visited websites worldwide (KAHLE and GILLIAT, 1996). Moreover, SE blogs represent other essential sources of information that practitioners usually go

for whenever seeking knowledge in the field. Almost 700 SE blogs (CHOI, 2015) written in English provide a communication channel for casually exchanging experiences among SE practitioners.

From this scenario, we can conjecture that the scientific production in the way it has been presented to practitioners has been neither adequate nor sufficient to fulfill their information needs appropriately, a situation that has also been reported in other research areas (STRAUS, TETROE, and GRAHAM, 2013). Also, the approaches aiming at putting the research knowledge into practice have not been accomplishing their goals on a larger scale (Chapter 2).

During an investigation of the issues presented in the context of searching for scientific knowledge, it was identified several pitfalls incurred from the search for scientific knowledge by those inexperienced in research procedures, which helps explain the reasons for the lack of popularity of scientific knowledge search among practitioners (Chapter 3). Moreover, different investigations allowed us to observe many barriers when understanding/using scientific production to support decision-making in SE (Chapter 3). From all studies, we identified that the pitfalls and barriers were related to challenges in the SE field that need to be tackled before expecting SE practitioners' efforts to pull scientific knowledge from scientific repositories.

More than persuading practitioners to search and use scientific productions, researchers shall provide means for practitioners to use scientific knowledge more naturally, as with less strict information sources. This way, scientific knowledge can be more largely used to ground decision-making during SE activities, and practitioners might do the EBSE actively.

Since the efforts to take scientific knowledge from producers to users must be employed by at least one of the involved parts (see Figure 1.1) and considering those issues mentioned above, we understand that the gap between research and practice needs to be solved by bringing the perspective of SE practitioners' information needs to the scientific production creation and reporting. Moreover, the communication channel to achieve better communication should be more aligned with what practitioners use. Table 1.1 summarizes the problem definition of research work.

Table 1.1 - Problem Definition

IDEAL SCENARIO: To guarantee the use of scientific knowledge during decision-making in the software industry, an efficient way of communication between researchers and practitioners must exist.

THE REALITY: The scientific production in the way it has been presented to practitioners has hampered its search, understanding, assessment, and use in industrial settings.

CONSEQUENCES: Software engineers rarely look for/use scientific approaches and results while building software or selecting software technologies to solve real problems in their software projects.

PROPOSED RESEARCH: This doctoral thesis is meant to support researching SE practical issues, which considers practitioners' real inquiries on SE and their perspectives about what should be considered while conducting practical research and reporting their results to practice.

1.3 Research Questions

Upon the motivation and problems stated, during this doctoral research, we investigated the following primary research question:

What to consider while planning, executing, and reporting empirical studies in software engineering to reach practitioners?

Table 1.2 presents the secondary research questions stated to support answering the primary one:

Table 1.2 – Secondary Research Questions

SRQ1: What are the problems faced by those who search for and use SE scientific evidence in practice? (Chapter 3)

The answer to this question allowed us to identify the:

- issues possible to be dealt with to promote the search and use of scientific productions in SE practice
- people responsible for taking the scientific evidence to practice: researchers or practitioners.

SRQ2: What are the practitioners' information needs that can be used to guide practical research on SE? (Chapter 4)

The answer to this question allowed us to identify the:

- relevant SE practical issues and common types of issues raised by practitioners
- key contextual information used by practitioners to share practical knowledge and common types of information important to practitioners assess shared knowledge

1.4 Research Goals

Figure 1.1 depicts an overview of the scenario in which the primary motivation of this research is placed, showing the knowledge dialog capacity to take scientific

productions from research into practice. Figure 1.2 emphasizes the other scenario that needs to be addressed in this research to accomplish this thesis's primary motivation: to understand existing SE practical issues to be researched and the information practitioners require while using the information to make decisions in industrial settings.
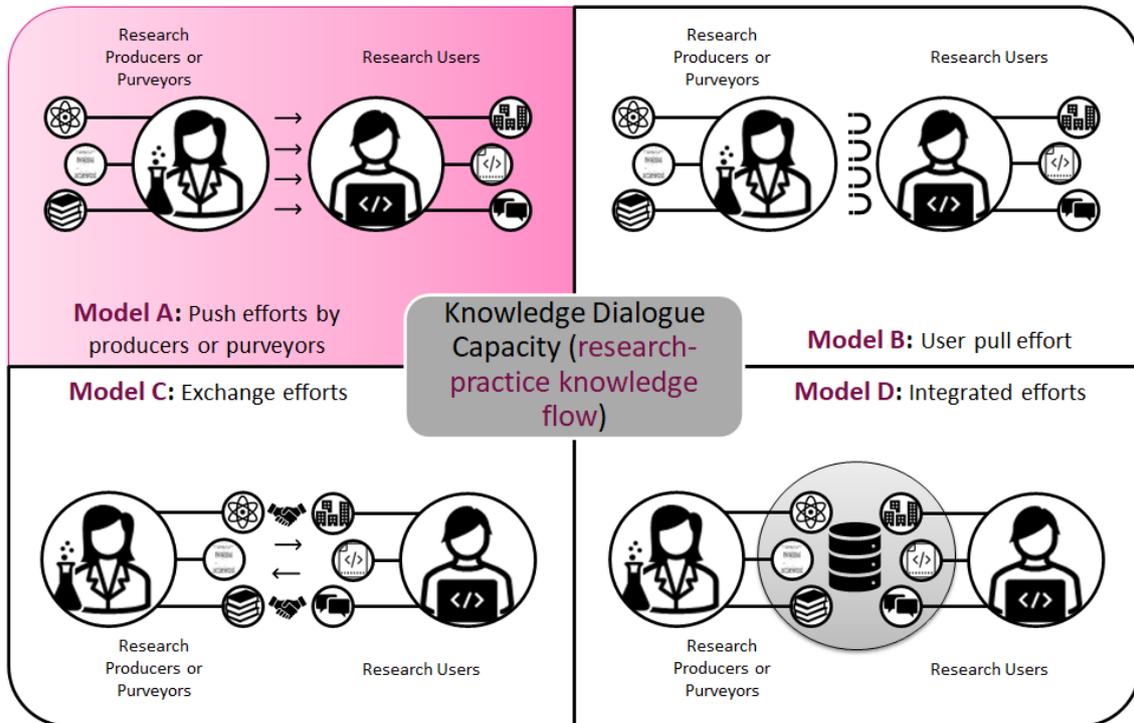
Figure 1.1 – Knowledge Dialogue Capacity in SE (Research to Practice Flow) – adapted from (LAVIS *et al.*, 2006) and (BENNETT and JESSANI, 2011)

The knowledge exchange flow can happen from research to research, from practice to practice, from practice to research, and from research to practice (details in Chapter 2). In each of these scenarios, there are different types of knowledge and efforts in taking this knowledge from the producer to its user. The target of this thesis is the knowledge flow related to taking scientific knowledge from research to practice (Figure 1.1) – which somehow requires getting to know knowledge from practice and taking them to research (Figure 1.2). Regarding the effort, we understand that an effort from the research side is more reasonable than an effort from the practice side – see discussions in Chapter 3 – even though integrated efforts should be the ideal goal. We believe that a pushing effort by scientific knowledge producers (**Model A** from Figure 1.1) alongside a pulling effort by practical knowledge users (**Model B** from Figure 1.2) is an initial step towards the ideal goal (**Model D** from Figure 1.1 and Figure 1.2) in which there is no SE research knowledge or SE practical knowledge, but SE knowledge. Hence, adopting the perspective of SE practitioners' information needs (Figure 1.2) while transferring scientific knowledge from research into practice is required.
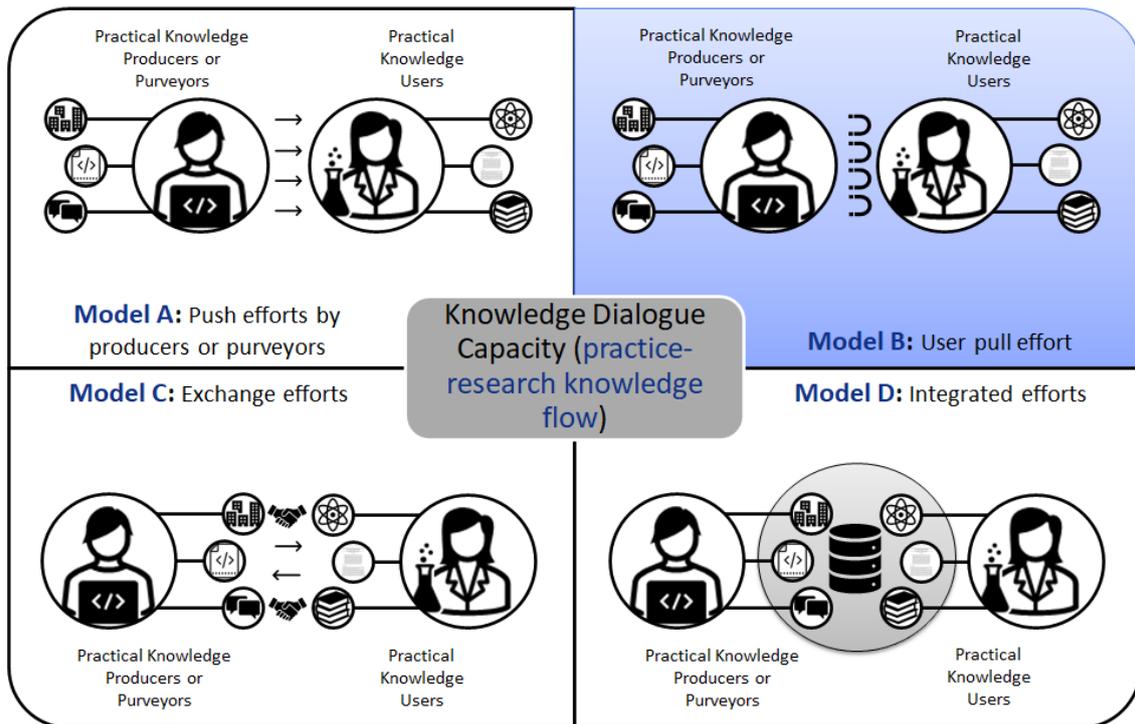
Figure 1.2 – Knowledge Dialogue Capacity in SE (Practice to Research Flow) – adapted from (LAVIS *et al.*, 2006) and (BENNETT and JESSANI, 2011)

In summary, this doctoral thesis intends to provide insights into the issues encountered while transferring SE scientific knowledge to practice that can ground different proposals for dealing with the challenge of bridging the gap between research and practice. Also, based on these insights, we intend to propose a solution to support researchers while researching SE topics to make them consider the main concerns regarding information needs from practice so that the scientific knowledge produced can reach practitioners.

The following goals need to be accomplished to achieve the primary purpose of this research. They were identified as goals based on the four main reasons that make users not use information: they do not know it exists; do not understand it; do not find it relevant, and they do not agree with it (BENNETT and JESSANI, 2011):

1. Identify the challenges to searching for SE scientific knowledge (Chapter 3).

2. Identify the challenges to understanding/using SE scientific knowledge with practical purposes (Chapter 3).

3. Understand what makes knowledge relevant to SE practitioners (Chapter 4).

4. Understand what makes SE practitioners trust/agree with knowledge (Chapter 4).

5. Provide a solution for researchers to research relevant issues to SE practitioners and report relevant information to SE practitioners (Chapter 5).

1.5 Research Methods

Before providing a solution for transferring scientific knowledge to practice, we must understand what prevents it. We performed an exploratory study to investigate issues in the channel used for communicating scientific knowledge that might represent challenges for practitioners in getting to know SE scientific knowledge. In addition, we conducted a family of studies to investigate issues involved in the message while using SE evidence in practice that might represent challenges for practitioners to understand and use SE scientific knowledge. These studies, with other published works that investigated the relevance and credibility of research results to practice, helped us identify the problems involved in taking scientific knowledge to practice and, thus, plan support for diminishing such problems.

The scientific knowledge engineering (SKE) approach (SANTOS and TRAVASSOS, 2016) depicted in Figure 1.3 was selected to conduct this thesis. We have chosen this methodology because it suits the need for knowledge structuring, and we applied it to practical and scientific knowledge, as explained next.
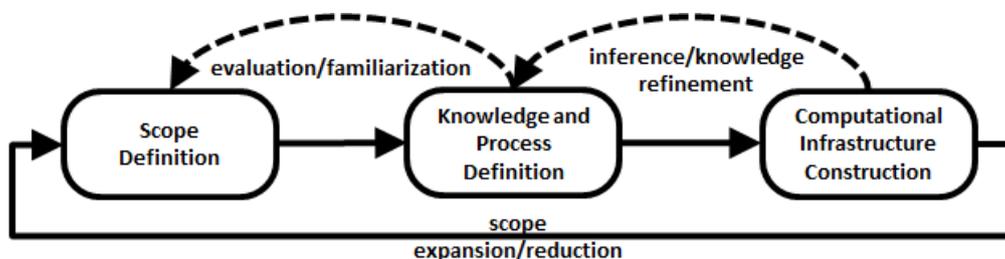
Figure 1.3 – Scientific Knowledge Engineering Phases – from (SANTOS and TRAVASSOS, 2016)

1. Scope Definition – This phase comprises activities related to knowledge familiarization and the first organization in which the knowledge engineer explores the knowledge heterogeneity and tries to organize and classify the concepts and their relationships. The problem and scope are defined in this phase. For this phase, we applied thematic analysis to questions and answers from an important practical SE Q&A forum to identify topics, side information, and presentation pattern that make questions and answers relevant to SE practitioners.

2. Knowledge and Process Definition – This phase comprises the activities related to domain conceptualization, knowledge representation definition, and

knowledge utilization process in which the knowledge engineer formalizes the obtained knowledge by determining objects, relationships, observations, and events representations, resolving conflicting ways of communicating the same knowledge. A process for extracting knowledge from materials to allow their modeling using the knowledge representation is also needed in this phase. For this phase, we used the analysis from the previous work on the questions and answers from the SE practical Q&A forum to build heuristics to support researchers using practical information during research activities. The heuristics support different research activities related to knowledge creation, such as problem identification, research question formulation, study design, and publication of the results.

3. Computational Infrastructure Construction – This phase comprises activities related to inference strategy specification and computerized infrastructure implementation in which the knowledge engineer specifies an inference strategy to understand how the obtained and organized knowledge is produced and used. A computational infrastructure is expected to be engineered in this phase. We built a prototype tool for this phase to support researchers with the repetitive and error-prone activities involved in using some proposed heuristics. The tool is not supposed to replace the cognitive tasks required in the research but to provide computational support for them to be performed promptly. The combination of the heuristics and the computational infrastructure we call Hermes.

To assess this thesis, we performed two observational studies on the use of two out of the eight proposed heuristics, together with data from Q&A forums, to support practical research. Their assessment provided means for improvements of the heuristics and the computational infrastructure that support their application on practical data. However, more assessments are required concerning the remaining heuristics and the computational infrastructure.

1.6 Structure of this Thesis

After presenting the motivations and problems involved in this work and defining the research questions, objectives, and methods, the next chapters present the investigations conducted to build the foundation for Hermes. The chapters are organized as follows:

Chapter 2 → Knowledge Diffusion, Transfer and Translation in Software Engineering. This chapter presents an overview of the meta-research proposals related to knowledge diffusion, transfer, and translation in SE, showing works for different

knowledge flows: research-research, practice-practice, practice-research, and research-practice.

Chapter 3 → Challenges to Taking Scientific Knowledge to Practice. This chapter presents the challenges of searching and understanding/using SE scientific knowledge. The identified findings were used to ground our understanding of the issues involved in the scientific knowledge that hampered its use by practitioners. In addition, it helped us to understand who should be responsible for transferring scientific knowledge to practice.

Chapter 4 → Understanding the Information Needs of Software Engineering Practitioners. This chapter presents the studies performed in Stack Exchange forums on identifying patterns of relevance and credibility of practical knowledge. It supported our understanding of what should be considered while researching practical issues.

Chapter 5 → Hermes – A Support for Researching on Real Software Engineering Issues. This chapter presents Hermes, the product of this thesis, which comprises eight heuristics and computational infrastructure to support practical research activities. Hermes was developed grounded on the results and conclusions from all five studies performed during this thesis, along with the findings from related works.

Chapter 6 → Hermes Assessment. This chapter presents an overview of the different ways of assessing Hermes and presents the two observational studies planned and executed to assess two of the eight Hermes heuristics.

Chapter 7 → Final Considerations. This chapter presents the final considerations, contributions, limitations, and future works that can be envisioned with this thesis.

2 Knowledge Diffusion, Transfer, and Translation in Software Engineering

"Knowledge is and will be produced to be sold. It is and will be consumed to be valorized in a new production: in both cases, the goal is to exchange." – Jean-Francois Lyotard

2.1 Introduction

In this chapter, we present the context in which this research is placed by providing an overview of the knowledge flows in SE and the applicability of some well-known meta-research[2] proposals to support transferring knowledge from producers to its users. Through looking at the meta-research proposals in SE reported by Molléri et al. (MOLLÉRI, PETERSEN, and MENDES, 2018), and presented more recently in Felderer and Travassos (FELDERER and TRAVASSOS, 2020) alongside new publications in the field, we identified those that can provide some support on the knowledge transferring from research to research, practice to practice, practice to research, and research to practice. This chapter does not intend to provide an extensive list of works but rather an overview of the main ones.

2.2 Knowledge Diffusion, Transfer and Translation: Concepts and Supports

From knowledge creation to its application, three important concepts support our understanding of the whole process of taking knowledge from those who produce it to those who use it: knowledge diffusion, transfer, and translation. These concepts are associated and commonly used interchangeably in the technical literature because they are related to moving knowledge from its producer (transferor) to its user (transferee).

While knowledge diffusion is related to knowledge spreading – usually freely and passively – to a target population of users, knowledge transfer requires an agreement between the involved parties and an active and intentional process for disseminating and acquiring knowledge (HAMERI, 1996). Moreover, the success of a knowledge transfer is accomplished once the receiver can adequately use knowledge in their

---

[2] In this work, we consider meta-research the investigations and proposals of research practices to promote evidence-based improvements in the quality and effectiveness of research.

environment – a need that is not necessary for knowledge diffusion (RAMANATHAN, 2008).

Budgen, Kitchenham, and Brereton borrowed the knowledge translation concept from medicine to SE. It is defined as "the exchange, synthesis and ethically-sound application of knowledge – within a complex system of interactions between researchers and users – to accelerate the capture of the benefits of research through better quality software and software development processes" (BUDGEN, KITCHENHAM, and BRERETON, 2013). Knowledge translation differentiates from knowledge transfer primarily because of its message characteristics. In medicine and SE, the authors advocate for the message to be quality assessed, synthesized, and aggregated before its transfer to reduce bias from individual works. Thus, a solution for knowledge translation in SE should consider all steps among gathering knowledge until its selection for use.

Before accomplishing a complete knowledge translation, a couple of questions should be answered by either transferors or transferees, according to Lavis et al. (LAVIS *et al.,* 2003) and other authors in the field (SUDSAWAD, 2007) (BENNETT and JESSANI, 2011). These questions intend to identify correctly the knowledge that should be translated, the people involved in the process, and the process itself:

   i.   What should be transferred (knowledge)?

  ii.   By whom should knowledge be transferred (transferors)?

 iii.   To whom should knowledge be transferred (transferees)?

  iv.   With what effect should knowledge be transferred (knowledge transfer goal)?

   v.   How should knowledge be transferred (the knowledge transfer process and communication infrastructure support)?

These questions guide the presentation of meta-research proposals for knowledge diffusion, transfer, and translation in SE. Also, we discuss the sources of efforts (pushing efforts by the transferor, pulling efforts by the transferee, or exchange efforts from both) to make the transfer or translation of knowledge happen.

Figure 2.1 depicts knowledge production environments (research and practice) and storage (knowledge sharing repositories). As mentioned by Bennett and Jessani, "different contexts require different strategies for transferring knowledge […] the choice of 'push' tools will differ, as will the opportunities for facilitating pull and creating linkage and exchange partnerships" (BENNETT and JESSANI, 2011). Therefore, the presented proposals do not necessarily support knowledge diffusion, transfer, or translation.

However, discussing their applicability to support knowledge transfer in SE might help the reader understand the related works to this thesis.
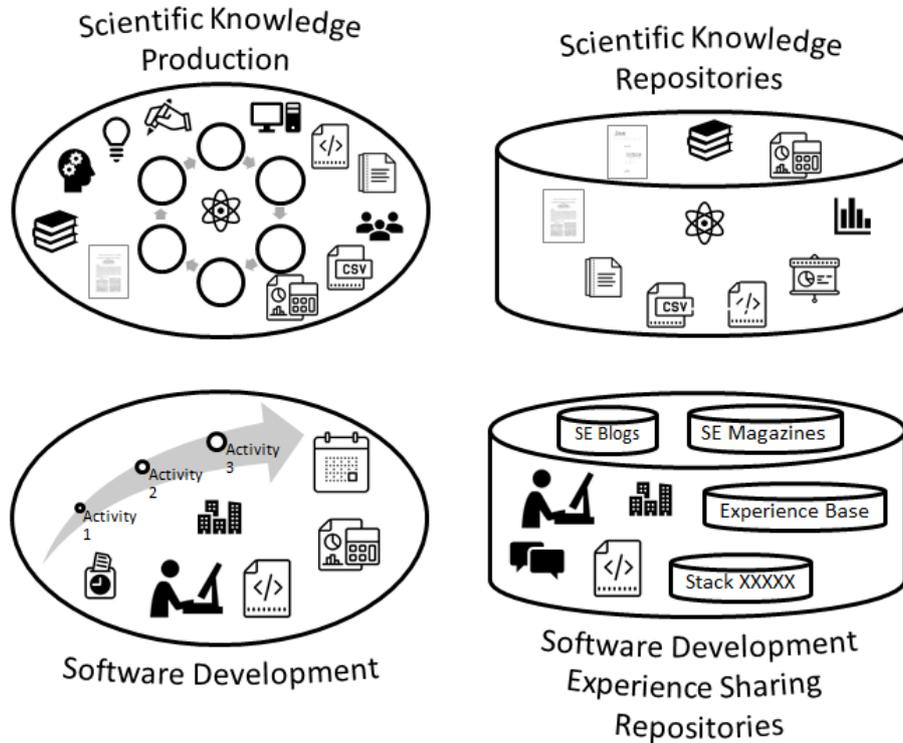
Figure 2.1 – Knowledge Production Environments and their Repositories

2.2.1 The Research-Research and Practice-Practice Knowledge Flows

Figure 2.2 presents the first two knowledge flows: knowledge produced in research and transferred/translated to research; and knowledge produced in practice and transferred/translated to practice. As one might expect, producers can efficiently use their productions, which is explained by the fact that they are used to the type of knowledge, argumentation, and format used to share knowledge through their main communication channels (e.g., repositories).
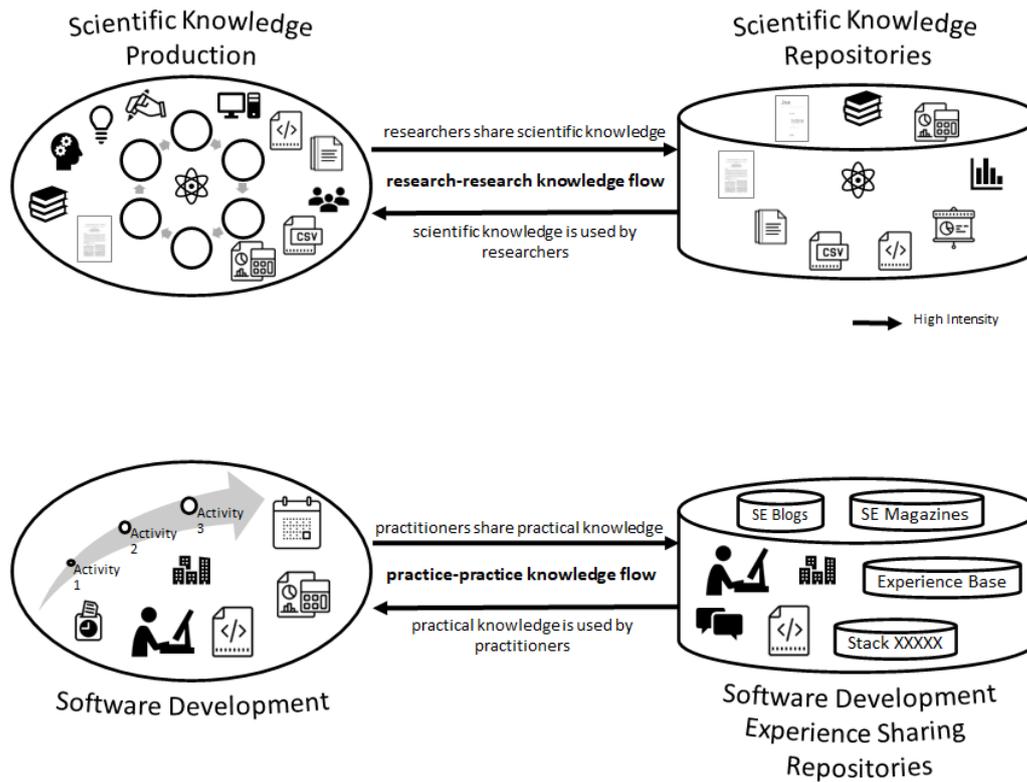
Figure 2.2 – Research-Research and Practice-Practice Knowledge Flows

2.2.1.1 The Research-Research Knowledge Flow

The knowledge transferred in the research-research knowledge flow is usually **scientific knowledge reporting software technologies**, associated **empirical studies**, and other **investigations regarding SE phenomena**. Since science is continuously evolving, there are intense efforts to push scientific knowledge to research repositories and pull it from the repositories, and commonly these efforts have been done through research articles and literature reviews (SANTOS, 2015).

Concerning the **pushing efforts**, different guidelines for SE were proposed to support the writing of scientific abstracts (BUDGEN *et al.*, 2008), the reporting of experiments (JEDLITSCHKA, CIOLKOWSKI, and PFAHL, 2008), case studies (RUNESON and HÖST, 2009), experimental replications (CARVER, 2010), simulation-based studies (FRANÇA and TRAVASSOS, 2014), surveys (DE MELLO and TRAVASSOS, 2015), and others. Moreover, more generic proposals were presented to support the preparation of empirical study reports in general for evidence synthesis (WOHLIN, 2014), which might include the identification and reporting of contextual information necessary to the understanding of scientific productions (GALLIS, ARISHOLM, and DYBÅ, 2003) (HÖST, WOHLIN, and THELIN, 2005) (KARAHASANOVIC *et al.*, 2005) (DYBÅ, SJøBERG, and CRUZES, 2012) (BALDASSARRE, FRANÇA, and SILVA, 2016) (PETERSEN *et al.*, 2021). These

proposals represent ways of improving the quality of scientific reports stored in scientific databases for future retrieval and use by other researchers in the field.

Concerning the **pulling efforts**, *ad-hoc* literature reviews have been used by researchers to withdraw scientific knowledge since the beginning of scientific investigations. More recently, guidelines for conducting literature reviews were proposed to support the retrieval of appropriate works from scientific databases (BIOLCHINI *et al.*, 2005) (KITCHENHAM and CHARTERS, 2007) (PETERSEN *et al.*, 2008) (PETERSEN, VAKKALANKA, and KUZNIARZ, 2015) (KITCHENHAM, BUGDEN, and BRERETON, 2016) (CARTAXO, PINTO, and SOARES, 2020) (KITCHENHAM and BUDGEN, 2022), and work references (WOHLIN, 2014) (MOURÃO *et al.*, 2020) (WOHLIN *et al.*, 2020). Figure 2.3 shows the steps usually followed in systematic literature reviews, such as the ones proposed in those guidelines.
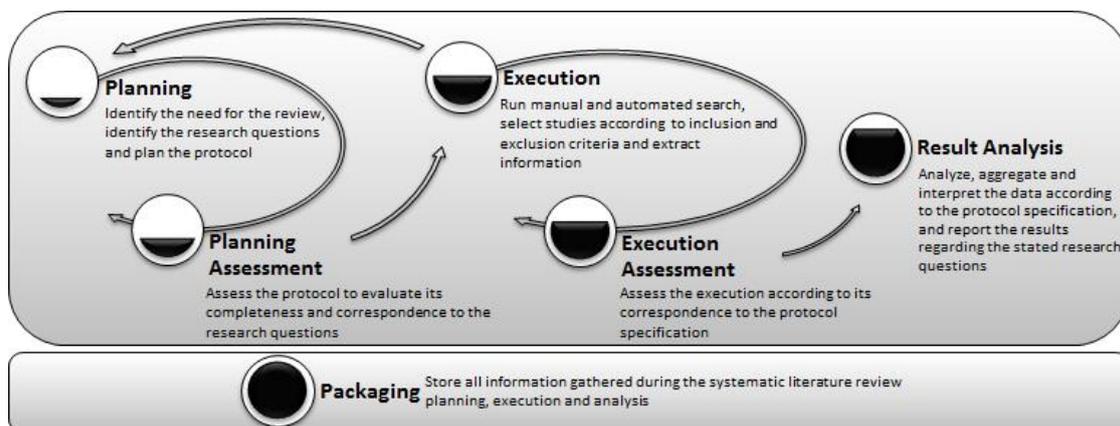
Figure 2.3 – Steps to Systematically Survey Scientific Productions in SE

Apart from the support for reporting and retrieving information, different methods for synthesizing and aggregating evidence were brought to SE from other fields of science as additional mechanisms for organizing scientific knowledge before its reporting and after its collection. There are many examples of meta-research in this regard outside SE that are used in the SE field, such as grounded theory (GLASER and STRAUSS, 1967), meta-ethnography (NOBLIT and HARE, 1988), and content analysis (KRIPPENDORFF, 2004). Some of them were adapted specifically for SE, such as thematic analysis (CRUZES and DYBA, 2011), meta-analysis (SHEPPERD, 2013), and case surveys (PETERSEN, 2020). Moreover, new aggregation methods were created, such as the Structured Synthesis Method (SANTOS, 2015), besides supports that can guide the choice of these methods (CRUZES and DYBÅ, 2011) (DIESTE *et al.*, 2011). Table 2.1 provides an overview of research-research knowledge flow efforts in SE.

Table 2.1 – Research to Research Knowledge Flow Efforts

Knowledge: scientific knowledge reports, papers, books, and data about software technologies, their associated empirical studies, and other investigations on SE phenomena	
Transferors: Researchers	**Transferees:** Researchers
Knowledge Transfer Goal (transferors' perspective): communicate scientific findings on software technologies	**Knowledge Transfer Goal (transferees' perspective):** identify research gaps and opportunities; replicate empirical studies; aggregate studies; organize scientific knowledge

Pushing Efforts by Transferors		Pulling Efforts by Transferors	
Knowledge Organization/Report: use of → context models for characterizing evidence; scientific methods for knowledge assessment, synthesis, and aggregation; reporting guidelines and templates	**Knowledge Diffusion:** presentation of works in → scientific conferences; scientific journals; scientific databases; researchers' blogs	**Knowledge Searching/ Gathering:** use of → literature review guidelines, scientific search engines	**Knowledge Organization/Use:** use of → scientific methods for knowledge assessment, synthesis, and aggregation

2.2.1.2 The Practice-Practice Knowledge Flow

In the practice-practice knowledge flow, the knowledge transferred usually regards **software development technologies and experiences about their use** in real settings. In addition, practitioners use previous **software development experiences** (feedback, lessons learned, deliverables, among others) to support the planning and improvement of future software developments. Although there are no meta-researchers to support this process since it involves activities that start and end in the industry, some proposals in SE research provide support for practitioners to learn from their own experiences in practice.

Learning from experience is not new in the SE field. For instance, the quality improvement paradigm (QIP) (BASILI, 1985) is one of the first approaches to reusing all types of experiences from different software projects in an organization. The QIP is a goal-driven feedback-oriented improvement paradigm that advocates the need for project characterization and quantifiable goal definition to select appropriate process models, methods, and tools for this project.

In 1994, the Experience Factory infrastructure (see Figure 2.4) was created using the QIP as a fundamental methodological instrument for pushing and pulling

experiences in software organizations. Apart from the previous steps defined in QIP, the Experience Factory added additional steps to support the generalization of experiences from finalized projects and their tailoring to new ones based on the projects' contextual characteristics. The project characterization is one of the central infrastructure parts since it needs to be unambiguous to enable future comparison among projects. Basili, Caldiera, and Rombach provide some hints of the information that should be taken into account while characterizing the projects, such as "the number of people, level of expertise, group organization, problem experience, process experience, […] application domain, newness to state of the art, susceptibility to change, problem constraints, […] life cycle models, methods, techniques, tools, programming language, other notations, […] deliverables, system size, required qualities, […] target and development machines, calendar time, budget, existing software" (BASILI, CALDIERA, and ROMBACH, 1994).

After the Experience Factory infrastructure conception, different characterization approaches were proposed to identify the factors present in software development and maintenance that might affect the selection of software technologies such as (KITCHENHAM *et al.*, 1999), (PETERSEN and WOHLIN, 2009) and (CLARKE and O'CONNOR, 2012).
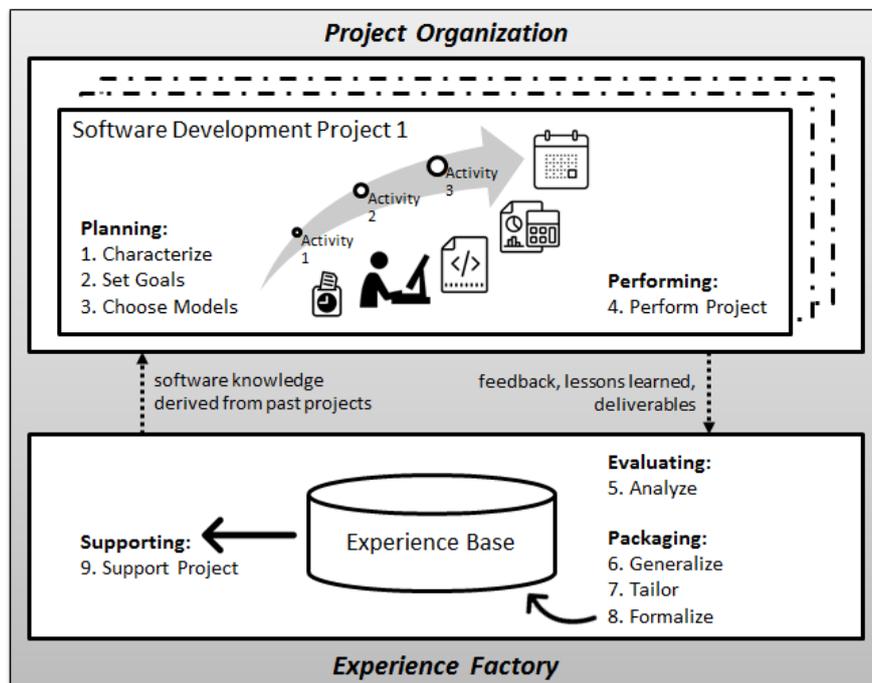
Figure 2.4 – Experience Factory – adapted from (BASILI, CALDIERA, and ROMBACH, 1994)

At the time of its development, the Experience Factory infrastructure had a more intra-organization improvement intention. Nowadays, SE practitioners are self-organized to share inter-organization experiences, mainly concerning software

development technologies. However, the followed process to accomplish such knowledge sharing (pushing and pulling experiences) is not necessarily as rigorous as the one established by the Experience Factory infrastructure, and we are yet to discover the factors that influence their decisions on software technology adoptions. Table 2.2 provides an overview of practice-practice knowledge flow efforts in SE.

Table 2.2 – Practice to Practice Knowledge Flow Efforts

Knowledge: software development technologies, experience reports, and data			
Transferors: practitioners		**Transferees:** practitioners	
Knowledge Transfer Goal (transferors' perspective): promote and/or sell software development technologies; help the community of software engineering practitioners		**Knowledge Transfer Goal (transferees' perspective):** identify software development technologies and improvement opportunities; collect information for decision-making support	
Pushing Efforts by Transferors		**Pulling Efforts by Transferors**	
Knowledge Organization/Report: use of → policies and rules from diffusion supports; Experience Factory process; context models for characterizing projects, software technologies, and experiences	**Knowledge Diffusion:** presentation of works in → industry conferences and forums; practitioners' blogs and knowledge-sharing databases; software technology repositories	**Knowledge Searching/ Gathering:** use of → Experience Factory process; practitioners' blogs and knowledge sharing databases; software technology repositories	**Knowledge Organization/Use:** documentation of → software development technologies and experiences

2.2.2 The Practice-Research and the Research-Practice Knowledge Flows

Figure 2.5 and Figure 2.6 present the other two knowledge flows discussed in this work: knowledge produced in practice and transferred/translated to research and knowledge produced in research and transferred/translated to practice. Unlike the other two presented knowledge flows, these flows are not as natural as one might expect, meaning neither practical knowledge is transferred to research nor scientific knowledge is transferred to practice by itself (BADAMPUDI, WOHLIN, and GORSCHEK, 2019). The knowledge, argumentation, and format used to share knowledge and the differences among sharing knowledge repositories represent challenges to be overcome while transferring knowledge between researchers and practitioners. The next sections

explain the main meta-research proposals that aim to bridge the gap between research and practice.
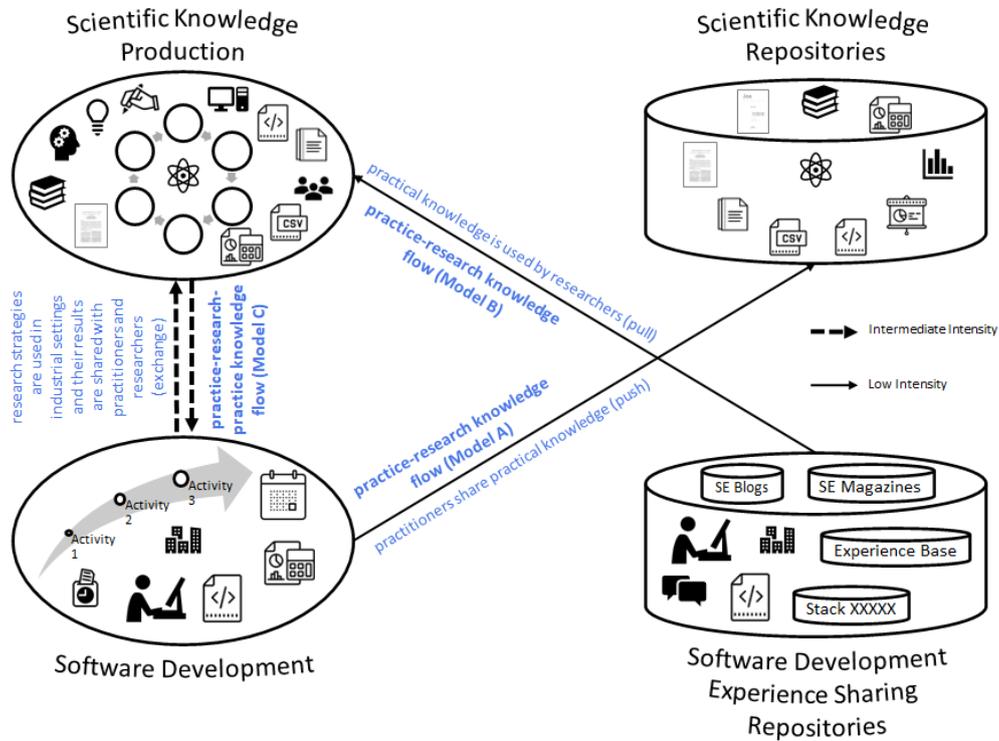
Figure 2.5 – Practice-Research Knowledge Flows – Pulling, Pushing and Exchange Efforts Involved
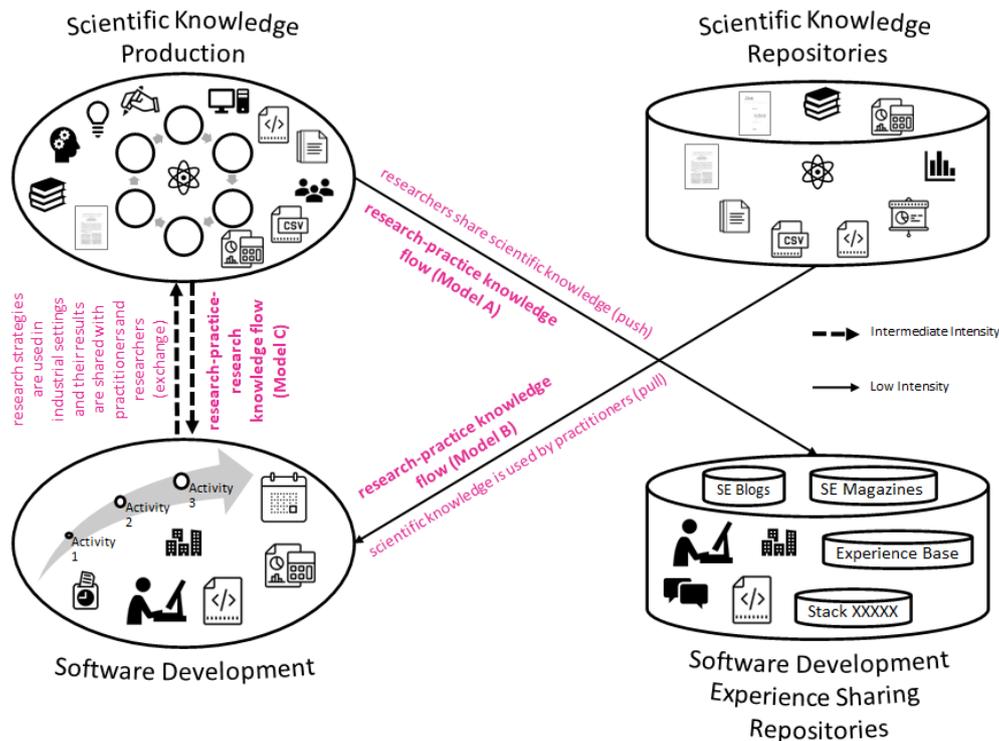
Figure 2.6 – Research-Practice Knowledge Flows – Pulling, Pushing and Exchange Efforts Involved

2.2.2.1 The Practice-Research Knowledge Flow

In the practice-research knowledge flow, the knowledge transferred is usually **software development technologies and experiences and practitioners' opinions and perceptions on software technologies and SE phenomena**. Different meta-research proposals have the intention of taking knowledge from practice to be used in research, although there are more approaches related to pulling efforts by researchers than to pushing efforts by practitioners.

Concerning the **pushing efforts** (Model A of Figure 2.5), we could not find any specific research to support practitioners in pushing knowledge to researchers, although we can point out that the same reporting guidelines used by researchers (see 2.2.1.1) can be used by practitioners when writing for scientific conferences to share their practical knowledge. Also, works providing writing tips for scientific conferences, such as the one proposed by Mary Shaw (SHAW, 2002) (SHAW, 2003) represent ways of helping practitioners to share knowledge in academic settings, especially in industry forums and tracks of scientific conferences such as ICSE (International Conference on Software Engineering), FSE (Symposium on the Foundations on Software Engineering), ASE (International Conference on Automated Software Engineering) and others.

Concerning **pulling efforts** (Model B of Figure 2.5), some strategies for gathering practitioners' experiences with software development, such as practitioners' repository mining and blog post-analysis, are provided. Examples of works using mining techniques to investigate software development practices from real software projects are (HASSAN, 2006) (CHOWDHURY and ZULKERNINE, 2011) (SCANDARIATO *et al.*, 2014) (DAM *et al.*, 2021). In some sense, they are included in data science works (MENZIES, WILLIAMS, and ZIMMERMANN, 2016) (SCOTT, MILANI, and PFAHL, 2020) in which a large amount of data is used to base research and theories.

Using blog posts as evidence for software engineering research (RAINER and WILLIAMS, 2019) is an example of gathering practitioners' experiences. Moreover, Garousi, Felderer, and Mantyla recently proposed a guideline for combining scientific knowledge from the technical literature, called white literature, with sources of information from practitioners available in magazines, Q&A sites, and blogs, among others, called grey literature (GAROUSI, FELDERER, and MÄNTYLÄ, 2019). The idea, in this case, is to collect practical software development knowledge and experiences from their sources of knowledge sharing (see Figure 2.7). Other works in this context are the ones that provide ways of assessing the quality of the knowledge collected from

grey literature to provide a more rigorous approach while using them to base scientific discussions (RAINER and WILLIAMS, 2019).
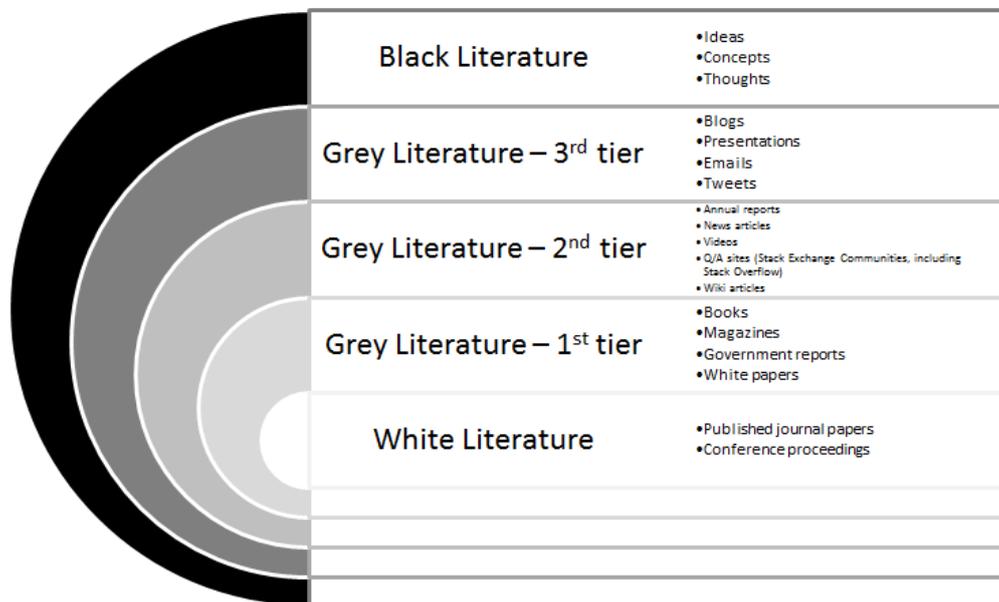
Figure 2.7 – Multivocal Literature Review Sources and their Credibility Dimensions

Concerning the **exchange efforts** (Model C of Figure 2.5), interviews (HOVE and ANDA, 2005), ethnography (ROBINSON, SEGAL, and SHARP, 2007) (SHARP, DE SOUZA, and DITTRICH, 2010), and survey (LINÅKER *et al.*, 2015) (MOLLÉRI, PETERSEN, and MENDES, 2016) are the most used research methods to acquire knowledge from practitioners which require more collaboration from practitioners than the works presented in the pulling efforts by researchers. While ethnography focus on observing real software development settings without interference, the interview and survey focus on capturing information by asking practitioners questions on software technologies and phenomena. The results of such studies can return to practice as a way of providing diagnosis for industry on a specific topic, aside from providing interesting insight to research on a particular topic.

A more recent strategy for getting knowledge from practitioners is using surveys combined with interactive posters presented to practitioners at industrial events. The idea is to provide a way of collecting additional information from practitioners in environments where they feel more comfortable sharing knowledge (DIEBOLD *et al.*, 2017). All these strategies require effort from researchers and practitioners to succeed with knowledge gathering from practice to research.

Table 2.3 gives an overview of the practice-research knowledge flow efforts in SE.

Table 2.3 – Practice to Research Knowledge Flow Efforts

Knowledge: software technologies, software development experience reports and data, practitioners' opinions and perceptions of software technologies and phenomena			
Transferors: practitioners		**Transferees:** Researchers	
Knowledge Transfer Goal (transferors' perspective): promote and/or sell software development technologies; help researchers understand real software development problems		**Knowledge Transfer Goal (transferees' perspective):** investigate real software development phenomena; identify software development problems; collect practitioners' evaluation of software technologies	
Pushing Efforts by Transferors		**Pulling Efforts by Transferors**	
Knowledge Organization/Report: use of → writing support for practitioners in scientific conferences	**Knowledge Diffusion:** participation in →interviews, ethnography, and surveys; presentation of works in → scientific conferences; industrial conferences and forums; practitioners' blogs; practical knowledge sharing databases; software technology repositories	**Knowledge Searching/ Gathering:** use of → interview, ethnography, and survey guidelines; multivocal literature review guidelines	**Knowledge Organization/Use:** use of → scientific methods for knowledge assessment, synthesis, and aggregation

2.2.2.2 The Research-Practice Knowledge Flow

In the research-practice knowledge flow, the knowledge that is transferred is usually **scientific knowledge reporting software technologies and their associated empirical studies**. Our intention as researchers is to provide evidence to practitioners to make informed decisions during software developments, while practitioners (might) expect evidence to make these informed decisions during software development.

Concerning the **pushing efforts** (Model A of Figure 2.6), some works intend to support the organization and presentation of scientific knowledge and its diffusion to practitioners. We can organize these works into at least three categories: i) works that intend to provide models for knowledge diffusion (see Figure 2.8) (PFLEEGER, 1999) (PFLEEGER and MENEZES, 2000); ii) works that intend to identify the knowledge information that should be transferred to practitioners (JEDLITSCHKA *et al.*, 2007) (JEDLITSCHKA, CIOLKOWSKI, and PFAHL, 2008) (IVARSSON and GORSCHEK,

2011) (JEDLITSCHKA, JURISTO, and ROMBACH, 2014) (PETERSEN *et al.*, 2021), and iii) works that intend to provide mediums through which knowledge can be transferred to practitioners (GRIGOLEIT *et al.*, 2015) (CARTAXO *et al.*, 2016) (STOREY *et al.*, 2017). While using these proposals, researchers are responsible for organizing the scientific knowledge in a way that is better for practitioners' consumption, without mentioning which channels of communication to use with practitioners other than the ones the research already uses.
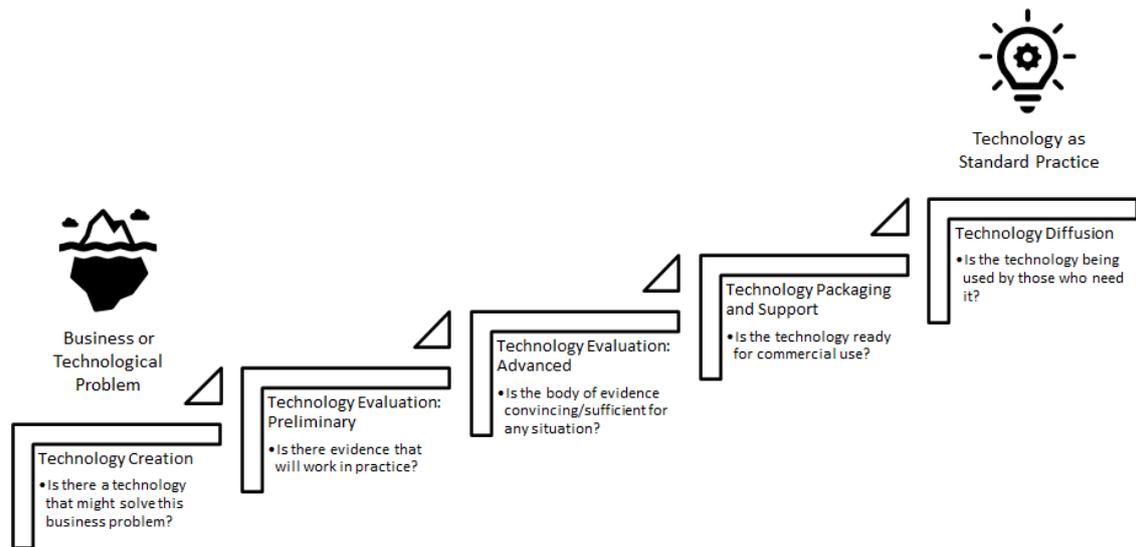
Figure 2.8 – From Problem to Standard Practice – adapted from (PFLEEGER, 1999)

The evidence-based software engineering (EBSE) brought the expectation that practitioners can be responsible for employing the main effort in knowledge transferring from research into practice by **pulling scientific knowledge from its sources** (Model B of Figure 2.6). Kitchenham, Dyba, and Jorgensen borrowed the concepts and methods from evidence-based medicine and defined EBSE as a rigorous process for systematically collecting and combining scientific evidence with practical experiences to support technology adoption decisions by practitioners in the software industry (KITCHENHAM, DYBÅ, and JøRGENSEN, 2004) (DYBÅ, KITCHENHAM, and JøRGENSEN, 2005). This approach uses systematic literature reviews and knowledge translation concepts for capturing, assessing, synthesizing, aggregating, and employing scientific knowledge in software organizations. It is a research framework that requires several research strategies to translate knowledge (considered to be aggregated evidence) from research into practice, including those to characterize the context of the retrieved evidence and the intended application environment. Figure 2.9 overviews the five steps required during an evidence-based practice.
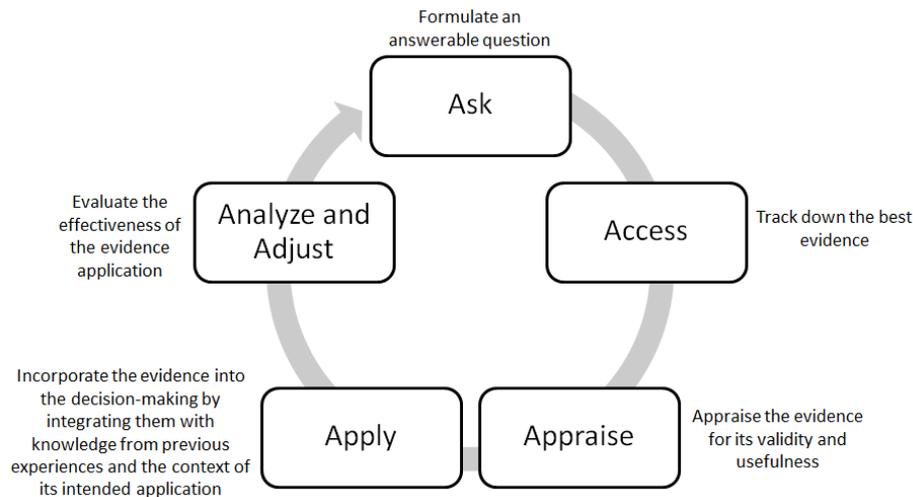
Figure 2.9 – The Five Steps of Evidence-based Practice – adapted from (SACKETT *et al.*, 2000)

So far, the **exchange efforts** from both sides are the ones that have shown to be the most effective way of taking scientific knowledge from research into practice (Model C of Figure 2.6). They usually take advantage of the EBSE approach, but instead of expecting practitioners to use it, researchers are responsible for dealing with research concerns. Examples of methodologies in this scenario are action-research (SUSMAN and EVERED, 1978) (SANTOS and TRAVASSOS, 2011) and design science (HEVNER and CHATTERJEE, 2010) (RUNESON, ENGSTRÖM, and STOREY, 2020).

Action-research is a type of research based on the search for actions to be taken (also called intervention) to solve a real industrial problem. The methodology requires synergy among researchers and practitioners to i) identify the problem; ii) search, select and assess appropriate actions to solve the problem; and iii) spread the results. Figure 2.10 depicts the action-research cyclic process of five phases commonly used when applying this research strategy inside an organization. In each phase of its cycle, different research methods can be employed, such as interviews, ethnographies, and surveys during diagnostic, systematic literature reviews (KITCHENHAM and CHARTERS, 2007) (KITCHENHAM and BUDGEN, 2022), mapping studies (PETERSEN *et al.*, 2008) (PETERSEN, VAKKALANKA, and KUZNIARZ, 2015), and rapid reviews (CARTAXO, PINTO, and SOARES, 2018) (CARTAXO, PINTO, and SOARES, 2020) during the planning, etc. Consonant to action-research, design science advocates using scientific methods to identify and solve real problems in the industry, requiring the design of a solution for the problem. This way, its process, which is also cyclic, requires a problem conceptualization, a solution design, and empirical validation (RUNESON, ENGSTRÖM, and STOREY, 2020).
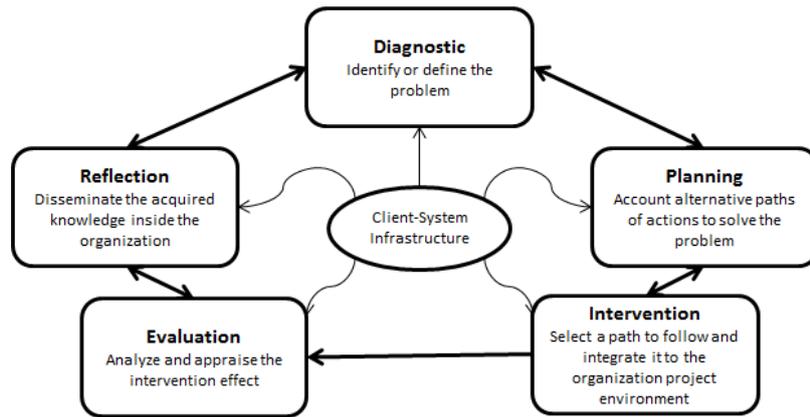
Figure 2.10 – Action Research Canonical Process – adapted from (SANTOS and TRAVASSOS, 2011) and based on (SUSMAN and EVERED, 1978)

Still in the line of works related to taking knowledge from research to practice requiring exchange efforts from researchers and practitioners are the ones that propose models for research-practice collaborations such as the Model for Technology Transfer in Practice (GORSCHEK *et al.*, 2006) (GORSCHEK and WNUK, 2020). While action-research and design science use a cyclic process when trying to solve industrial problems, the model depicted in Figure 2.11 provides seven steps from identifying an industrial problem/issue until the release of its solution. The main difference presented in this model is the requirement for three forms of validation, one in academia and two in industry. These validations increase the chances of successfully satisfying a solution to the stated problem inside the industry.
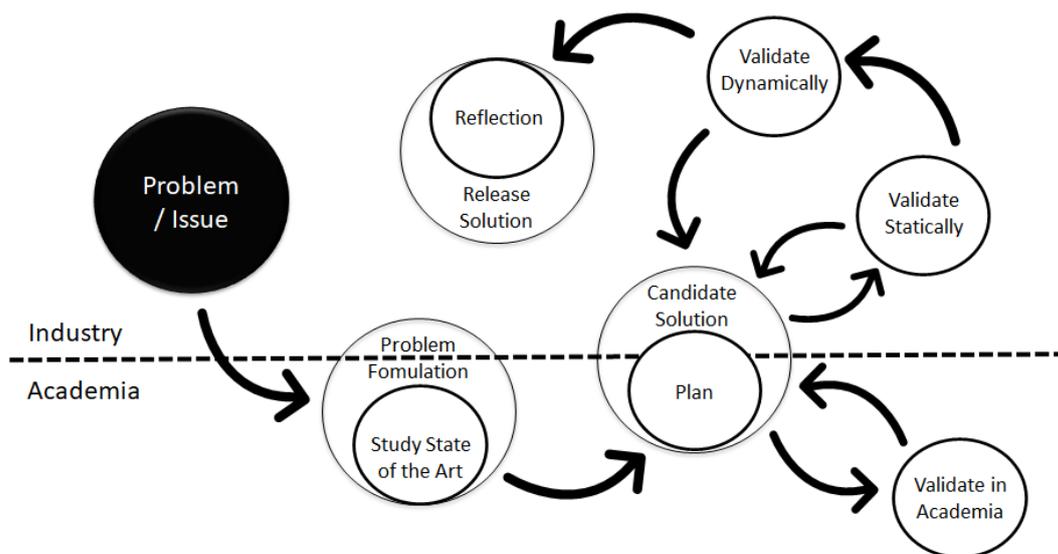
Figure 2.11 – Model for Technology Transfer in Practice – adapted from (GORSCHEK *et al.*, 2006) (GORSCHEK and WNUK, 2020)

With the focus on the knowledge translation cycle – which includes the synthesis of evidence – the work proposed by (BADAMPUDI, 2018) (BADAMPUDI, WOHLIN, and GORSCHEK, 2019) is a framework that provides a set of steps from the identification of knowledge and its assessment to the follow-up of its use after the initial adoption (steps 1-9 of knowledge translation in Figure 2.12). The framework combines contextualized expert opinions (especially those that can use the knowledge) with research evidence to guarantee an appropriate knowledge application in practice.
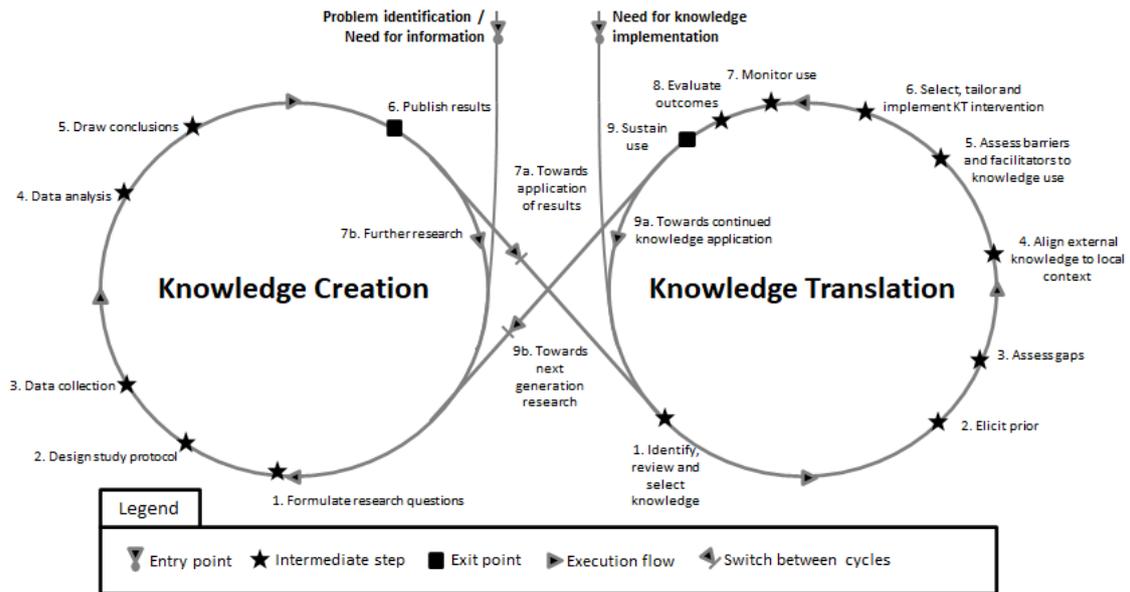
Figure 2.12 – Knowledge Creation-Translation Cycles – adapted from (BADAMPUDI, 2018)

With many proposals, Wohlin and Runeson presented a guideline to choose among those methodologies the most appropriate for an industry-academia collaboration (WOHLIN and RUNESON, 2021). Table 2.4 provides an overview of research-practice knowledge flow efforts in SE.

Table 2.4 – Research to Practice Knowledge Flow Efforts

Knowledge: scientific knowledge reports on software technologies and their associated empirical studies			
Transferors: Researchers		**Transferees:** practitioners	
Knowledge Transfer Goal (transferors' perspective): help practitioners with evidence to support decision-making in software development		Knowledge Transfer Goal (transferees' perspective): collect information for decision-making support	
Pushing Efforts by Transferors		**Pulling Efforts by Transferors**	
Knowledge Organization/Report:	**Knowledge Diffusion:** use of → models for	**Knowledge Searching/ Gathering:**	**Knowledge Organization/Use:**

use of → models for knowledge information description; mediums for knowledge presentation	knowledge diffusion; models for research-practice collaborations	EBSE (use of → literature review guidelines, especially rapid reviews)	EBSE (use of → models for knowledge translation; scientific methods for knowledge assessment, synthesis, and aggregation; context models for software project characterization)

2.2.3 The Ideal Knowledge-Sharing Scenario

The ideal knowledge-sharing scenario (Figure 2.13) is based on a standard terminology, type of knowledge, argumentation, medium, and repository for knowledge sharing. Unfortunately, very few approaches were created toward this ideal goal.

In some sense, many proposals presented in this chapter – especially those that combine information from research and practice – could be adapted to achieve part of this sharing scenario, focusing on standardized reporting and standardized collection of SE knowledge. To the best of our knowledge, the proposal by Santos was the first one with such intention (SANTOS, 2015). It provides the information required for scientific knowledge organization, the medium for representing it, a strategy to combine different types of knowledge (scientific and practical), and the repository for knowledge sharing. However, since most of the work focused on aggregating scientific evidence, the proposal leans towards a more scientific way of sharing knowledge.
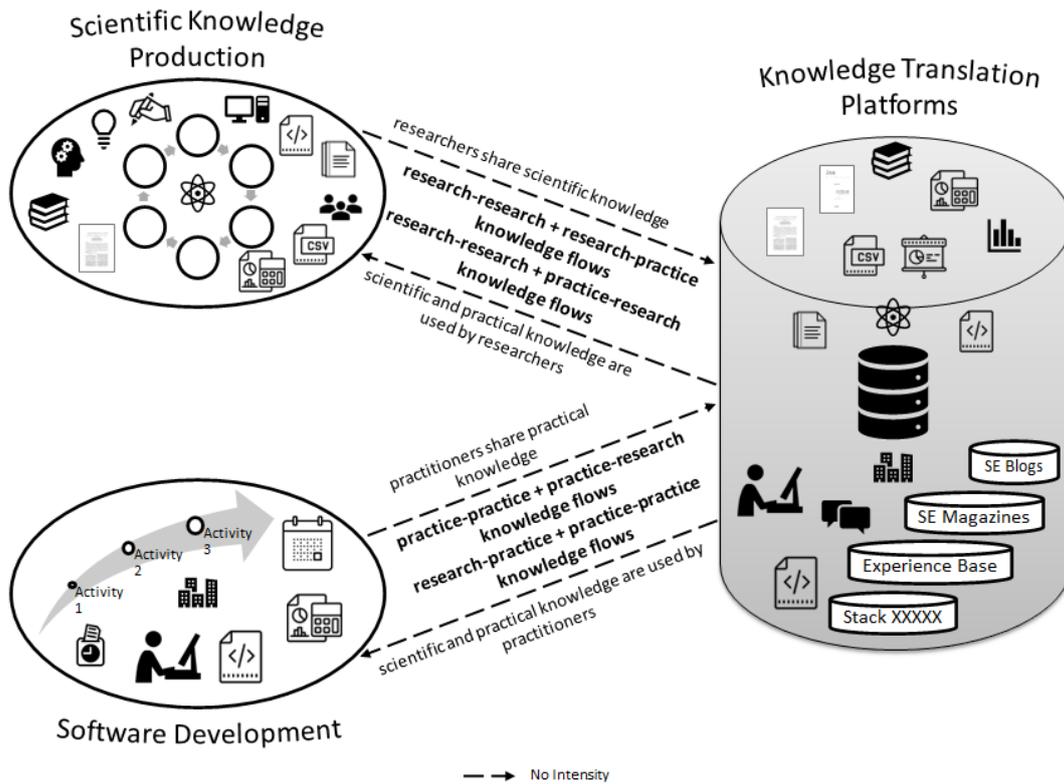
Figure 2.13 – Ideal Knowledge Sharing – Integrated Efforts and Common Repository

Although the main purpose of science is not to serve practice, any practical research should ideally provide means to reach practitioners properly. While the main way of sharing scientific knowledge is still through research articles, additional forms of communicating research evidence can pave the path for bridging the gap between research and practice, making scientific findings reach practice on a larger scale.

2.3 Discussions and Conclusions of this Chapter

Since its beginning, empirical software engineering (ESE) has been considered a rigorous discipline intended to support industrial decision-makers (BASILI, SELBY, and HUTCHENS, 1986). The EBSE has arisen in the context of ESE to support SE practitioners in gathering scientific knowledge from scientific sources and using it to assist software projects.

While the primary motivation of these approaches is to be a tool for practitioners, most of the searches for scientific knowledge are performed by academics, meaning software engineers either have not been using them or have not been reporting their usage (SANTOS and SILVA, 2013). It is possible to find some reports about significant achievements by using scientific knowledge in industrial software scenarios (KASOJU, PETERSEN, and MÄNTYLÄ, 2013) (RIBEIRO and TRAVASSOS, 2015) (LÓPEZ et al., 2015). However, most of them result from academia-industry collaborations, in which

academics search for scientific knowledge and translate it to practitioners, whereas practitioners deal with applying the translated findings to real projects.

Even in research-practice collaborations (in which researchers are the ones employing efforts in searching for scientific knowledge), the differences in research interests between research and practice and in perceptions of what solutions and outcomes are useful (GAROUSI, PETERSEN, and OZKAN, 2016) can overturn an excellent opportunity of bridging the research-practice gap.

In the next chapter, we discuss the issues involved in searching and understanding/using the scientific production that can make the Model A of Figure 2.6 the most appealing strategy for putting scientific knowledge from research into practice.

3 Challenges to Taking Scientific Knowledge to Practice

"Knowledge is of no value unless you put it into practice." – Anton
Chekhov

3.1 Introduction

According to Bennett and Jessani, there are four main reasons for the intended (BENNETT and JESSANI, 2011): i) they do not know the information exists or what action to take in its regard; ii) they do not understand the information; what it means and why it is essential; iii) they do not care and see the information as irrelevant, not beneficial to their agenda; iv) they do not agree, think the information is misguided or false.

This chapter overviews two works we performed to investigate whether (i) and (ii) hold in the SE field. The first study regards the investigation of challenges in surveying scientific evidence in the SE field. The second study is a family of studies investigating challenges in translating SE knowledge. The results presented in this chapter allow us to answer the following research question (see Table 3.1):

Table 3.1 – SRQ1 – Secondary Research Question 1

SRQ1: What are the problems faced by those who search for and use SE scientific evidence in practice?
The answer to this question allowed us to identify the: • issues possible to be dealt with to promote the search and use of scientific productions in SE practice   • people responsible for taking the scientific evidence to practice: researchers or practitioners.

3.2 Challenges and Pitfalls of Surveying Scientific Knowledge in SE

We performed an exploratory study to investigate the challenges and pitfalls of surveying scientific knowledge in SE. This study compared the planning and results of seven systematic literature reviews (SLR) dealing with the same research question and performed by similar teams of novice researchers (who have slightly better knowledge than practitioners regarding empirical methods). The study allowed us to identify the

main pitfalls that caused the seven SLR plans and reports to present unexpected differences. Additionally, the pitfalls provided hints for identifying the challenges existing in SE that might hamper the search for scientific knowledge by practitioners. The following sections provide an overview of the exploratory study planning and results. An overview of the demographics of this study can be found in Appendix A of this thesis. Details can be found in (RIBEIRO, MASSOLLAR, and TRAVASSOS, 2018).

3.2.1 Exploratory Study Planning Overview

The context of the exploratory study was during two years of the experimental software engineering course at COPPE/UFRJ (2010 and 2012). Being a course on empirical software engineering, the students received lectures and assignments on topics related to primary and secondary studies in SE.

Among the assignments, the students were invited to participate in an empirical study in which they would have to evolve an SLR research protocol, execute it and present the research results. Quality of use cases was the topic used for investigating the SLRs. The subject of use cases was selected because it is a basic topic in the SE field where the participants would have more knowledge and experience and feel more comfortable working.

The students (seven D.Sc. and 14 M.Sc.) were organized into seven groups of three, balancing the amount of D.Sc. and M.Sc. among them and assuring the existence of at least one practitioner in each team. The participants were graduate students in their first year, and none had previous knowledge of experimental methods before the course.

For the assignment, the teams received an initial SLR research protocol filled with the research question they should use to base their search for knowledge – "Which quality attributes (and measurements used to evaluate such attributes) have been empirically studied for use cases?" – and other information concerned with: i) initial terms to support the search string formulation; ii) the search engines to be used in the search; iii) initial inclusion and exclusion criteria, and iv) suggestions for information to be extracted from the scientific productions.

For two months, the teams should complete the protocol filling, perform the SLR according to their planning and present the identified results (quality attributes for use cases). They were advised to modify the research protocol according to their understanding of the research topic and question, though neither the research question nor the search engines should be changed since we understood any change in these

parts of the protocol would certainly lead to different results that would make unfeasible to perform any comparison on the works.

Search string terms; returned, excluded, and included papers; and quality attributes for use cases were among the analysis used for the comparison (using Jaccard and Kappa coefficients) of SLR research protocols and reports. Since the teams received the same training and initial research protocol, the results were expected to be similar. Whether this would not be the case, we wanted to identify the probable reasons for the differences.

3.2.2 Exploratory Study Results Overview

Figure 3.1 presents an overview of the quantitative data from the teams' SLR plans and reports. Looking at the raw numbers, we can observe a significant contrast among the groups. For example, the selected search terms ranged from eleven terms used by the Black team to 215 terms by the Purple, while the returned papers ranged from 157 papers that came back in the Pink search to 661 in the Black search. Interestingly, the number of quality attributes did not vary as much, although they do not necessarily represent the same attributes across the reports.
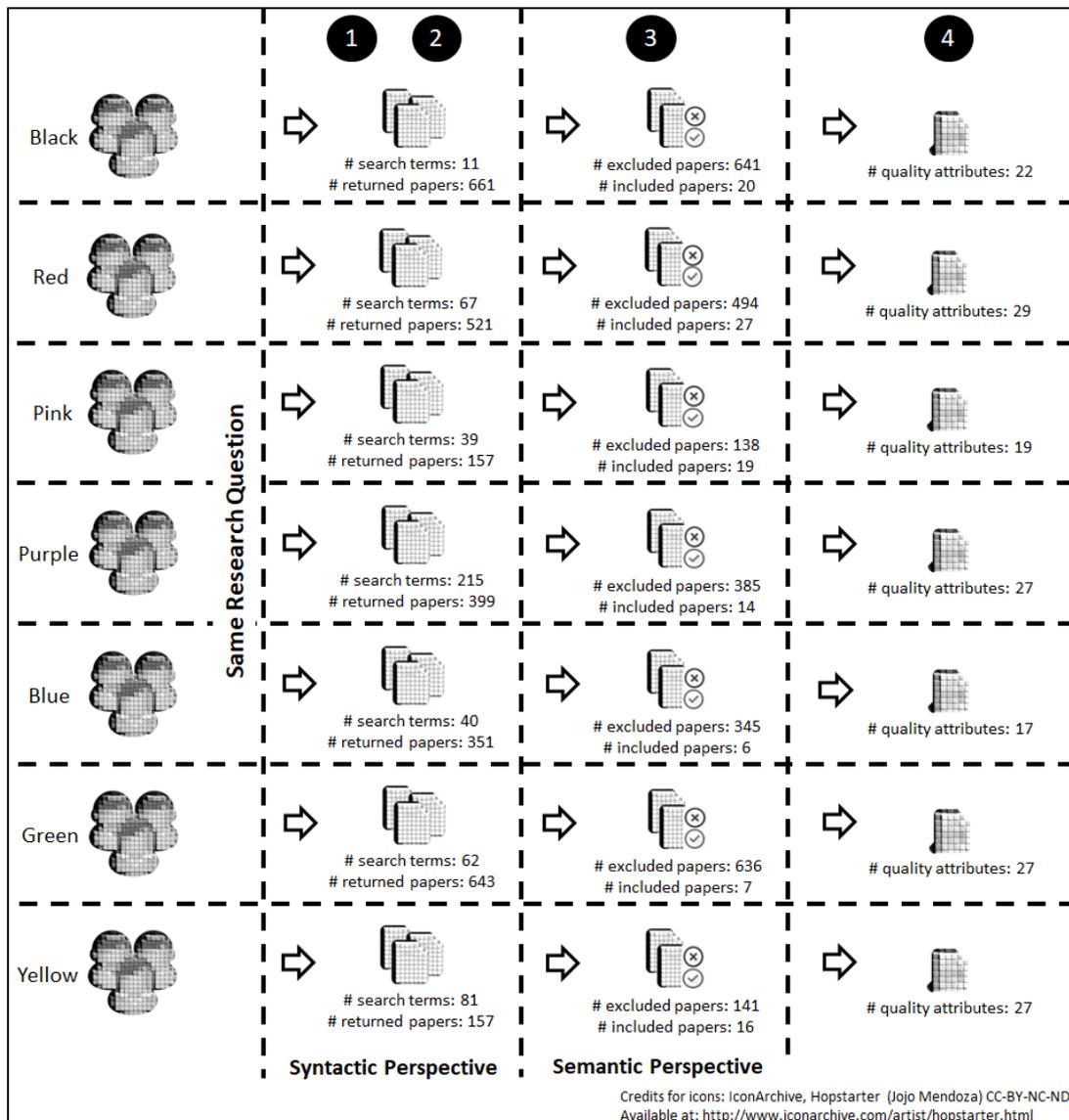
Figure 3.1 – Summary of the Seven SLRs Quantitative Data – adapted from (RIBEIRO, MASSOLLAR, and TRAVASSOS, 2018)

As the teams received the same training on experimental topics, the same research question to base their SLR research planning, and even the same initial protocol, we expected more similarities than the ones observed. However, rarely did a pair of teams reach more than 18% and 12% similarity concerning the search terms and returned papers, respectively. Concerning the papers excluded and included in common by a pair of teams (given the same set of returned papers), the results were better, even though the agreement between SLRs was more related to papers a pair of teams agreed to exclude than agreed to include. As for the similarities related to the reported quality attributes for use cases, no pair of teams reached 100% of similarity even when both teams in the comparison included the same set of papers.

Figure 3.2 presents the expected versus obtained results from the comparison of the seven: search strings (1); sets of returned (2), excluded and included papers (3); sets of quality attributes for use cases (4). Neither the same research questions nor similarities in research protocols and points of view were enough to induce the teams to report similar outcomes.
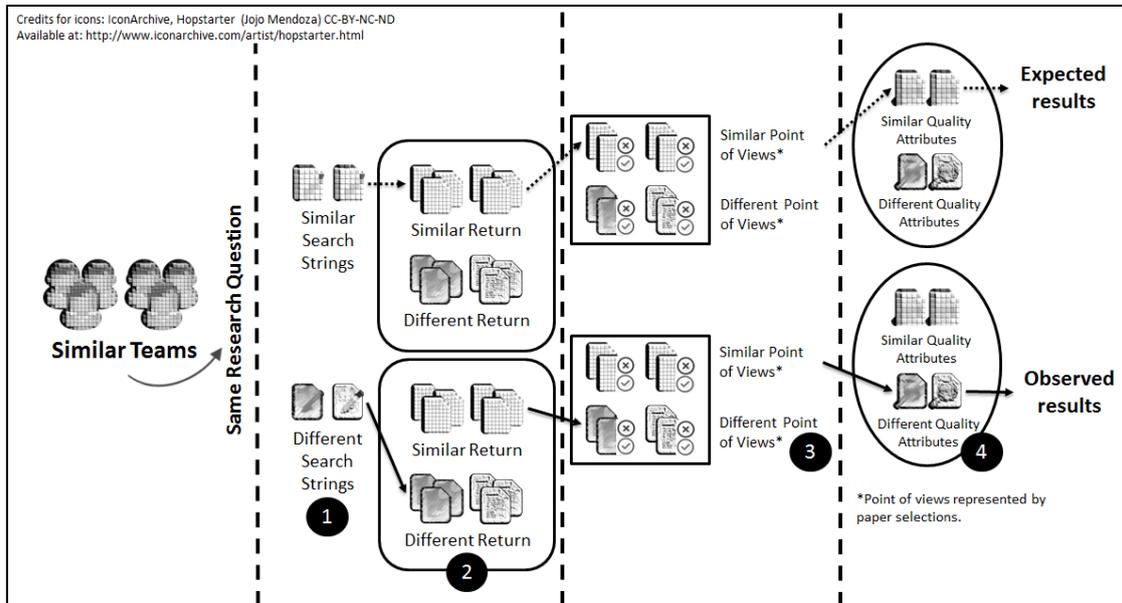
Figure 3.2 – Expected Results versus the Observed Results – adapted from (RIBEIRO, MASSOLLAR, and TRAVASSOS, 2018)

We know that the differences in the search strings induced a high quantity of different returns. For this reason, comparing the excluded and included papers was only performed considering the papers returned in common by the teams under analysis (see part 2 of Figure 3.2). A similar approach was done while comparing the quality attributes, meaning only the extraction of papers accepted in common were used for the comparison (see part 3 of Figure 3.2). Even when analyzing the same set of papers included by the teams, the quality attributes for use cases were not the same.

3.2.3 Problems in Knowing the Information Exists

Two months to perform the SLR was not enough to accomplish better results since it involves time-consuming tasks. Searching databases, choosing papers, and extracting data represent some of the leading barriers to SLRs (CARVER *et al.*, 2013). While this is true, if we consider the EBSE should be used in software development settings, two months can be the maximum limit for performing a software technology adoption decision.

While searching for the leading causes for the differences in the seven SLR executions, we identified the following pitfalls related to the main points of comparison:

(1) & (2) Different Search Strings and Different Returns:

    i.   <u>unnecessary search terms</u> – some teams used various words with the same root form while others used only the terms' root forms to perform the search;

    ii.   <u>keywords too high-level</u> – some teams preferred to use generic over specific terms;

    iii.   <u>many different combinations of terms producing noise return</u> – some teams tried to maximize the number of terms, disregarding whether they were valid search terms;

    iv.   <u>inappropriate selection of search terms</u> – some teams used search terms not related to the topic under investigation;

    v.   <u>overuse of 'and' operators and distributive properties</u> and <u>misuse of the guidelines</u> – some teams did not organize the search terms using the PICO strategy (PAI *et al.*, 2004), as recommended in the provided initial protocol and guidelines. This fact induced the search strings logic to differ among the SLR plans, resulting in separate returns even in cases similar to the search terms.

(3) & (4) Different Points of View and Different Quality Attributes:

    i.   <u>misunderstandings concerning empirical/experimental study strategies</u> and <u>no explanation concerning the empirical focus used</u> to assess the papers – some teams classified the same papers using divergent experimental study type labels, and this influenced the quality assessment and/or selection of the papers;

    ii.   <u>tacit knowledge regarding the study selection strategy</u> – some teams did not provide any explanation on the inclusion or exclusion of papers;

    iii.   <u>controversial understanding of the research topic</u>, <u>misinterpretation of the research question</u> – some teams included papers not related to the specific topic under investigation (quality of use cases), but rather to a more general topic (e.g., software quality);

    iv.   <u>the subjectivity of the research synthesis strategy</u> – overall, the teams did not thoroughly describe the procedure to combine the results from the

included papers. It influenced the differences in the teams' reports concerning quality attributes identified in the same set of included papers.

Six main reasons – challenges for conducting SLRs in SE – can support explaining the mentioned pitfalls. They are the lack of:

- experience in the investigated topic that caused novice researchers to misuse terms in the search string, include studies, and give answers not related to the research question;

- experience of novice researchers in systematic reviews promoting inconsistencies in their review protocol and execution and making them do unnecessary work and not report relevant information;

- a standard terminology regarding use cases, requirements, and quality attributes that made novice researchers search for studies using a variety of different terms and report the results inconsistently;

- clearness and completeness of the papers that might have caused their inconsistent inclusion/exclusion and data collection among the reviews;

- verification procedures to support the identification of inconsistencies throughout the SLR process;

- commitment or interest in the research topic caused novice researchers to overlook essential features of SLRs, not report significant decisions made during its process nor report details on the results.

Even considering that practitioners could search for scientific knowledge less systematically, from the results of this study, we can conclude that the pitfalls encountered by the students (with slightly more knowledge in research methods than practitioners) would be identified with practitioners trying to search for scientific knowledge in the SE field as well. It means that it would not be easy for practitioners to retrieve the information they were looking for. This scenario worsens because most research articles are not free/available to everyone, and the movement for open science is still an initial one (MENDEZ *et al.*, 2020).

Most of the challenges identified – especially those related to the lack of standard terminology and clearness and completeness of research articles, are not overcome easily in the sense that they would require a joint effort from a different group of researchers all around the world. Still, we believed these challenges were worth investing further since they are related to understanding the information provided and

being aware of its existence. Therefore, the next section presents the studies identifying specific barriers and challenges to understanding/using scientific knowledge in SE.

3.3 Challenges and Barriers to Understanding and Using Scientific Knowledge in SE

In 2014, during an academia-industry collaboration in an embedded software development company, we applied research strategies under the action-research methodology to support developers in solving a problem related to recurrent reconstructions of source code (RIBEIRO and TRAVASSOS, 2015). After identifying that the source code reconstructions were caused by differences in the developers' source code quality perspectives, we decided to provide an evidence-based solution for supporting the practitioners in the alignment of their quality perspectives and the quality improvement of the source code produced in the company. Furthermore, this collaboration resulted in the construction of coding guidelines focused on source code reading and comprehension (results from the masters of this doctoral student).

To accomplish it, we performed a structured literature review to collect source code attributes, their measurement procedures, and their impact on source code reading and/or comprehension. We intended to use the attributes as input to formulate source coding guidelines, focusing on source code readability and comprehensibility.

At that time, we encountered several difficulties in gathering knowledge from the scientific literature to build a solution for the company, which made us make assumptions concerning the findings and carry out additional local observational studies to provide a solution for their problem in a short period. The difficulties were related to the extraction of information from the scientific reports, and the aggregation of their results since many evidence contradictions (the same source code attribute influenced both positively and negatively the source code reading and/or comprehension) were identified during the technical literature review.

Outside the company's context, we identified an opportunity to conduct a more in-depth investigation of the difficulties encountered while trying to combine evidence with building a solution for a practical problem, which led us to identify several barriers and challenges concerning the understanding and the use of scientific knowledge in SE. The following sections provide an overview of the studies' planning and results. An overview of the demographics of this family of studies can be found in Appendix A of this thesis. Details can be found in the works (RIBEIRO and TRAVASSOS, 2018) and (RIBEIRO, SANTOS, and TRAVASSOS, Submitted).

3.3.1 Structured Literature Review Re-Execution

3.3.1.1 Structured Review Planning Overview

The main difficulty identified while building a software technology from scientific knowledge was revealing the important information from the papers and aggregating their results since many of them were contradictory. Using the same objective from (RIBEIRO and TRAVASSOS, 2015) for collecting source code attributes, their measurement procedures, and their impact on source code reading and/or comprehension, we re-executed the structured literature review search string using the Scopus engine, re-evaluated the returned papers, and re-extracted the information from the accepted ones. Our focus was not only on extracting the previous information required in the literature review protocol but also on extracting any additional information that could support identifying the reasons for the contradictory findings in the initial returned works.

According to our understanding of empirical investigations, we conjectured that two main reasons could explain the encountered differences in effects caused by the same source code attribute:

1. Use of different concepts regarding source code reading[3] and comprehension[4].

2. Existence of unknown context variables to moderate or mediate the cause-effect relationship.

Thus, any additional information that would support explaining the evidence contradictions, for instance, the ones stated previously, should be collected. In some sense, moderators and mediators in a cause-effect observation support describing a phenomenon of interest. We hypothesize that this description can support the identification of reasons for evidence contradictions in the literature.

> **Note:**
>
> Before proceeding with our investigation report, it is essential to differentiate the mediator and moderator concepts. Both mediators and moderators go in between a cause-effect relationship. Mediators are

---

[3] The readability of source code can be defined as a quality characteristic based on the human judgment of how easy it is to read the written code. This characteristic relates to program elements and their presentation to programmers (RIBEIRO and TRAVASSOS, 2017).

[4] The comprehensibility of source code is a quality characteristic based on the human judgment of how easy it is to understand the source code. This characteristic relates to the source code interpretation by its reader (RIBEIRO and TRAVASSOS, 2017).

responsible for linking the cause to its effect, supporting and explaining how external variables are combined to produce a particular effect. On the other hand, moderators are responsible for affecting the cause-effect relationship`s direction and/or strength (WU and ZUMBO, 2008). Mediators and moderators can represent qualitative (knowledge, experience, gender, and others) or quantitative (source code size, source code structural complexity, source code semantic complexity, and others) variables inside the relation between an independent or predictor variable and a dependent or response variable (BARON and KENNY, 1986).

For example, we can consider the relationship between the source code comprehensibility level and source code maintainability effort (the higher the comprehensibility level, the lower the maintainability effort). The existence of up-to-date code documentation can be a mediator variable in this relationship since it can support explaining why there is a relationship between the source code comprehensibility level and source code maintainability effort, while developers' domain knowledge level can be a moderator variable of this same relationship since it can affect the strength of this cause-effect relation. See below:

- Mediator: Existence of up-to-date code documentation. Source code with a high level of comprehensibility usually is well documented → code documentation can support reducing the human effort in understanding the source code → maintenance activities involve source code understanding → reducing effort in source code understanding can lead to a reduced source code maintainability effort.

- Moderator: Developers' domain knowledge level. The relation between the source code comprehensibility level and source code maintainability effort can be stronger for developers with a high level of domain knowledge and less intense for developers with a low domain knowledge level since, in this latter case, the developer can have difficulties not only to understand the source code but also the documentation it might present.

The explanation of cause-effect relationships through mediators and moderators is based on reasoning and formal logic. It is part of a (non-existent, to our knowledge) software engineering theory[5].

3.3.1.2 Structured Review Results Overview

Figure 3.3 and Figure 3.4 present an overview of the investigated source code entities, attributes, measurement procedures, and their impact on source code readability/comprehensibility. Each positive, negative, and neutral icon represents a cause-effect evaluation effect. A positive icon means that the attribute and the quality characteristic are directly proportional. A negative icon indicates that the attribute and

---

[5] In software engineering research, we passed the notion of just proposing software technologies and we started applying research methods to help us in constructing software technologies and evaluate them. However, whereas in other fields of research the evaluations of scientific phenomena are based on a strong theoretical background (not only through theories, but also through postulates, premises and arguments that can be verified), in SE, the research is still based on the freely reasoning of the researchers.

the quality characteristic are inversely proportional. As one can see, five attributes out of 13 presented contradictory evidence, not necessarily concerning the same measurement procedure, as we shortly explain after.
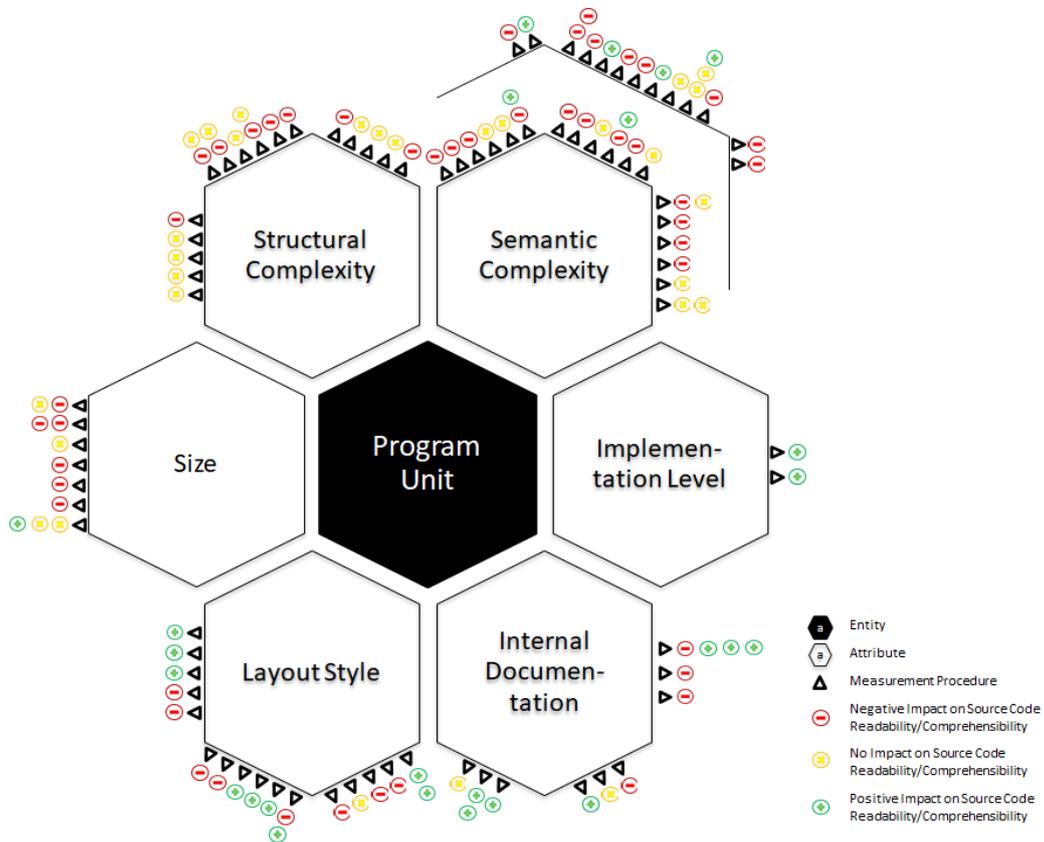
Figure 3.3 – Program Unit Entity and its Attributes
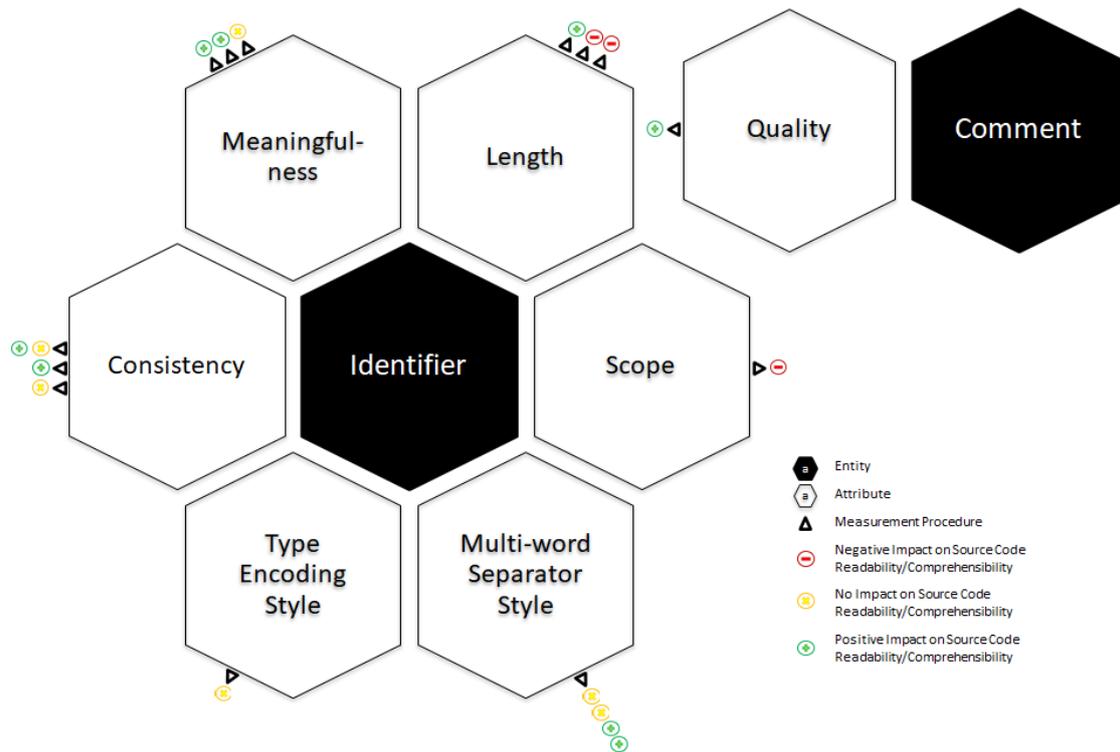
Figure 3.4 – Identifier and Comment Entities and their Attributes

The organization of the entities, attributes, and measurement procedures depicted in Figure 3.3 and Figure 3.4 is part of the synthesis we performed and explained (RIBEIRO and TRAVASSOS, 2018). The strategy used to organize the findings was based on a bottom-up continuous comparison, in which we: i) collected every variable that was mentioned to be the cause of a cause-effect relationship regarding source code reading or comprehension; ii) compared their measurement procedures; iii) clustered the ones that were measuring similar source code features; and then iv) abstracted from the clusters, the attributes of source code the measurement procedures were intending to measure, which could be related to the whole program unit or specific program elements. In some studies, we could not identify the differences among the investigated effects (readability and comprehensibility), which made us put them together in pretty much any cause-effect explanation.

Table 3.2 presents the works involved in the contradictory results illustrated in Figure 3.3 and Figure 3.4, along with their definition for the involved quality characteristics. Six works did not define the quality characteristics they were assessing, and almost all works mentioned only one source code quality characteristic – readability or comprehensibility – and did not necessarily provide a straightforward definition for the one used. Nevertheless, one can notice that the definition of readability of one work matches the definition of comprehensibility of another work and vice-versa, which

provides evidence that the works in the field use these two concepts interchangeably. Even when their definitions do not provide enough information concerning this fact, their measurement procedures can be used as a proxy for it. It is important to point out that the information concerning what was under assessment was not easy to be found in the papers since many did not follow guidelines for reporting studies.

The next sections provide an overview of the contradictory findings. It is impossible to directly identify the reasons for the results being different among the different works. Some works discuss the contextual information controlled during their experiment as a way of trying to guarantee the validity of the obtained results. Other works conjecture about the possible impacts of some contextual information on the relations (causality and correlation) between source code reading/comprehension, and the source code attributes under assessment.

Table 3.2 – Contradictory Works and their Definition of Readability and Comprehensibility

Work	Readability		Comprehensibility	
	Definition	Measurement Procedure	Definition	Measurement Procedure
(DEYOUNG and KAMPEN, 1979)	No definition	Human grading (1-5) on the readability of source codes	No mention	No mention
(JøRGENSEN, 1980)	"The readability of a text is the sum of stylistic factors that make the text more or less understandable to the reader."	Human grading (1-7) on the readability of source codes	No mention	No mention
(WOODFIELD, DUNSMORE, and SHEN, 1981)	No mention	No mention	No definition	Human grading (1-100) of answers about the comprehension of source codes
(MIARA *et al.*, 1983)	No mention	No mention	No definition	Human grading of answers about the comprehension of source codes and subjective rating (1-7) of source code difficulties
(TENNY, 1988)	"The more readable the coding is, the more quickly and accurately a programmer can obtain critical information about a program by reading the program text."	Human speed in answering comprehension questions and accuracy of human answers about the comprehension of source codes	No mention	No mention
(LAWRIE *et al.*, 2006) & H (LAWRIE *et al.*, 2007)	No mention	No mention	No definition	Human speed in describing source codes and recalling

			identifiers; accuracy of human descriptions of source codes, and human confidence in the descriptions provided
(BUSE and WEIMER, 2008) & (BUSE and WEIMER, 2010)	"We define readability as a human judgment of how easy a text is to understand."	Human grading (1-5) on the readability of source codes	No mention	No mention
(BUTLER *et al.*, 2011)	No definition	Readability metric	No mention	No mention
(FEIGENSPAN *et al.*, 2011)	No mention	No mention	"Program comprehension is an internal cognitive process that we cannot observe directly."	Human speed in correcting bugs in source codes and accuracy of human tasks related to solving defects in source codes
(POSNETT, HINDLE, and DEVANBU, 2011)	"Software readability is a property that influences how easily a given piece of code can be read and understood […] The relationship between readability and understanding is analogous to syntactic and semantic analysis. Readability is the syntactic aspect while understandability is the semantic aspect."	Human grading (1-5) on the readability of source codes	No mention	No mention
(TASHTOUSH *et al.*, 2013)	"Code Readability can be defined as a human judgment on how	Survey on source code attributes related to readability	No mention	No mention

	easy it is to understand a program source code."			
(LEE, LEE, and IN, 2015)	No definition	Readability metric	No mention	No mention
(BÖRSTLER and PAECH, 2016)	No definition	Human speed in reading source codes and human grading (1-5) on the readability of source codes	No definition	Human speed in completing comprehension tasks, the accuracy of human tasks related to filling in the blank spaces of source codes, and the accuracy of human descriptions of source codes

3.3.1.2.1 Program Unit Internal Documentation Contradictions

Table 3.3 presents the contradictory findings for comments. Even though the divergent results (positive and negative) are for different measurement procedures, both deal with the presence of comments for the source code. Overall, commenting on the source code is considered good practice for improving its comprehension. Among the works listed in Table 3.3, the one by Lee, Lee, and In (LEE, LEE, and IN, 2015) was the only one suggesting the contrary.

Table 3.3 – Measurement Procedures and Contradictory Findings for Program Unit Internal Documentation

Measurement Procedure	Impact on Source Code Readability/ Comprehensibility	Mediators	Moderators
Program Unit → Internal Documentation → Identify whether the program unit has comments	⊕ (WOODFIELD, DUNSMORE, and SHEN, 1981)	Meaningfulness of identifiers. Indentation presence.	Experience in programming.
	⊕ (TENNY, 1988)	Control structures complexity. Data structure complexity. Lexical structure complexity. Source code size.	Experience in the programming language used.
	⊕ (BÖRSTLER and PAECH, 2016)	No mention	Gender. Experience in programming. Naming style preferences. Domain knowledge.
	⊖ (LEE, LEE, and IN, 2015)	No mention	Experience in programming. Team size. Project size. Project maturity.

Program Unit → Internal Documentation → Identify whether the program unit has comments for methods	⊖ (LEE, LEE, and IN, 2015)	No mention	Experience in programming. Team size. Project size. Project maturity.
Program Unit → Internal Documentation → Identify whether the program unit has comments for variables	⊖ (LEE, LEE, and IN, 2015)	No mention	Experience in programming. Team size. Project size. Project maturity.

3.3.1.2.2 Program Unit Layout Style Contradictions

This group of contradictions regards the number of spaces used for indentation. The idea of analyzing the influence of indentation spacing on source code readability was presented in the empirical studies presented in Table 3.4. According to these works, indentation does aid program readability, but the number of spaces used for indentation differs among the studies. For example, the works described by Jørgensen (JøRGENSEN, 1980) and by Lee, Lee, and In (LEE, LEE, and IN, 2015) presented a positive relationship between the indentation spacing and the readability of source code, meaning the more spaces used for indentation, the better. Conversely, all other works described the negative relationship between the number of spaces used for indentation and readability, meaning that although significant, the indentation should not contain too many spaces to prevent the hampering of readability/comprehensibility.

Table 3.4 – Measurement Procedures and Contradictory Findings for Program Unit Layout Style

Measurement Procedure	Impact on Source Code Readability/ Comprehensibility	Mediators	Moderators
Program Unit → Layout Style → Identify the indentation variant among the four options: <u>zero space</u>, <u>two spaces</u>, <u>four spaces,</u> and <u>six spaces</u>	⊖ (MIARA *et al.*, 1983)	Control structures blocking.	Experience in programming.
	⊕ (LEE, LEE, and IN, 2015)	No mention	Experience in programming. Team size. Project size. Project maturity.

Program Unit → Layout Style → Divide the number of blank characters in the left margin by the number of characters in the program unit	⊕ (JøRGENSEN, 1980)	Programming style features variation	Experience in programming.
Program Unit → Layout Style → Divide the number of blank characters in the left margin by the number of lines in the program unit	⊖ (BUSE and WEIMER, 2008) & (BUSE and WEIMER, 2010)	Programming language feature.	Experience in programming.
Program Unit → Layout Style → Count the number of blank characters in the left margin of the deepest code block (maximum indentation)	⊖ (BUSE and WEIMER, 2008) & (BUSE and WEIMER, 2010)	Programming language feature.	Experience in programming.

3.3.1.2.3 Program Unit Semantic Complexity and Identifier Length Contradictions

Similarly, the studies in this group of contradictions did not measure program unit semantic complexity or the identifier length using the same measurement procedures. However, in some sense, their results can be compared. Table 3.5 presents the works and the real contradictions for these attributes. We can identify three central contradictions in Table 3.5: i) concerning arithmetic operators; ii) concerning commas; iii) concerning the length of identifiers. Most works identified that programming experience plays an important role while assessing source code readability/comprehensibility. They commonly state that programmers might identify more difficulties/facilities depending on their years of programming experience, which might make them be (less) used to source code writing features.

Table 3.5 – Measurement Procedures and Contradictory Findings for Program Unit Semantic Complexity and Identifier Length

Measurement Procedure	Impact on Source Code Readability/ Comprehensibility	Mediators	Moderators

Program Unit → Semantic Complexity → Divide the number of arithmetic operators by the number of lines	⊖ (JøRGENSEN, 1980)	Programming language feature.	Experience in programming.
	⊕ (BUSE and WEIMER, 2008) & (BUSE and WEIMER, 2010)	Programming language feature.	Experience in programming.
Program Unit → Semantic Complexity → $e^{-(\frac{NF}{LoC})}$, where NF is the number of arithmetic formulas and LoC is the number of lines of code	⊖ (TASHTOUSH et al., 2013)	No mention	Experience in programming.
Program Unit → Semantic Complexity → Divide the number of commas (parameters or variables in the declaration) by the number of lines	⊖ (BUSE and WEIMER, 2008) & (BUSE and WEIMER, 2010)	Programming language feature.	Experience in programming.
	⊕ (TASHTOUSH et al., 2013)	No mention	Experience in programming.
Program Unit → Semantic Complexity → Count the number of characters in the longest identifier (maximum length of identifier)	⊖ (BUSE and WEIMER, 2008) & (BUSE and WEIMER, 2010)	Programming language feature.	Experience in programming.
Program Unit → Semantic Complexity → Sum the length (in character) of each variable/identifier and divide it by the number of variables/identifiers (average length of variables/identifiers)	⊛ (DEYOUNG and KAMPEN, 1979)	Variable scope.	No mention
	⊕ (JøRGENSEN, 1980)	Programming language feature.	Experience in programming.
	⊛ (BUSE and WEIMER, 2008) & (BUSE and WEIMER, 2010)	Programming language feature.	Experience in programming.
Identifier → Length → Identify the length variant among three options: meaningful full-word; meaningful single-letter, and meaningful abbreviation	⊕ (LAWRIE et al., 2006) & H (LAWRIE et al., 2007)	No mention	Gender. Comfort with the programming language used. Experience with computer science.
Identifier → Length → Count the number of words in the identifier	⊖ (BUTLER et al., 2011)	No mention	Familiarity with typographical conventions. Knowledge of the application used.

| Identifier → Length → Count the number of characters in the identifier | ⊖ (BUTLER *et al.*, 2011) | No mention | Familiarity with typographical conventions.

Knowledge of the application used. |

3.3.1.2.4 Program Unit Size Contradictions

Table 3.6 presents three different works with divergent results on the impact of source code size on its readability/comprehensibility. Even though the divergent results (positive and negative) are for different measurement procedures, both deal with lines of code, allowing us to analyze their results together.

Table 3.6 – Measurement Procedures and Contradictory Findings for Program Unit Size

Measurement Procedure	Impact on Source Code Readability/ Comprehensibility	Mediators	Moderators
Program Unit → Size → Count the number of lines	⊛ (DEYOUNG and KAMPEN, 1979)	No mention	No mention
	⊛ (FEIGENSPAN *et al.*, 2011)	No mention	Experience in the programming language used.
	⊕ (POSNETT, HINDLE, and DEVANBU, 2011)	The number of characters in the source code. The number of words in the source code.	Experience in programming.
Program Unit → Size → Count the number of lines containing statements	⊖ (DEYOUNG and KAMPEN, 1979)	No mention	No mention

3.3.2 Local Studies Aggregation

3.3.2.1 Local Studies Aggregation Planning Overview

The information retrieved from the scientific articles was difficult to find inside the papers and even more difficult to combine. Readability and comprehensibility were used interchangeably in most works, which was reflected especially in the different measurement procedures used to perceive these quality characteristics in the source code (see Table 3.2). The source code attributes assessed also presented different

ways of measurement through the articles, and their impact on source code quality was, in some cases, contradictory. While very few works tried to explain the observed phenomenon through contextual information (especially concerning mediators), most of them conjectured that the experience in programming plays an important role (as moderator) while perceiving the impact of source code attributes on source code readability/comprehensibility.

These contradictions motivated us to plan, execute, analyze, and aggregate three local empirical studies on the impact of the presence of comments, indentation spacing, identifiers length, and code size on source code readability and comprehensibility, considering stratified results according to programming experience (novices and experienced programmers). The separate analysis concerning experience was an attempt to identify possible reasons for the contradictions identified in the technical literature and, at the same time, some hints on the important information to be reported by researchers to support understanding and using research findings.

The empirical studies were set up to be conducted in person in the classroom. In the three studies – each with a different group of people from computer courses – the participants received general explanations of source code quality from the perspective of readability and comprehensibility, discriminating against these two concepts (Figure 3.5 – part 1). No detail concerning the source code attributes was mentioned during the concept explanation. Next, after consenting to participate in the study (Figure 3.5 – part 2), the participants received the task package (Figure 3.5 – part 3) containing the characterization form and eight different source code snippets. The distribution of the snippets was randomized among the participants, meaning other participants received different packages of work (see options in Figure 3.5 – part 3). The final part of the study (Figure 3.5 – part 4) assessed the participants' source code snippets, in which they reported their perceptions/explanations/answers according to the requests.
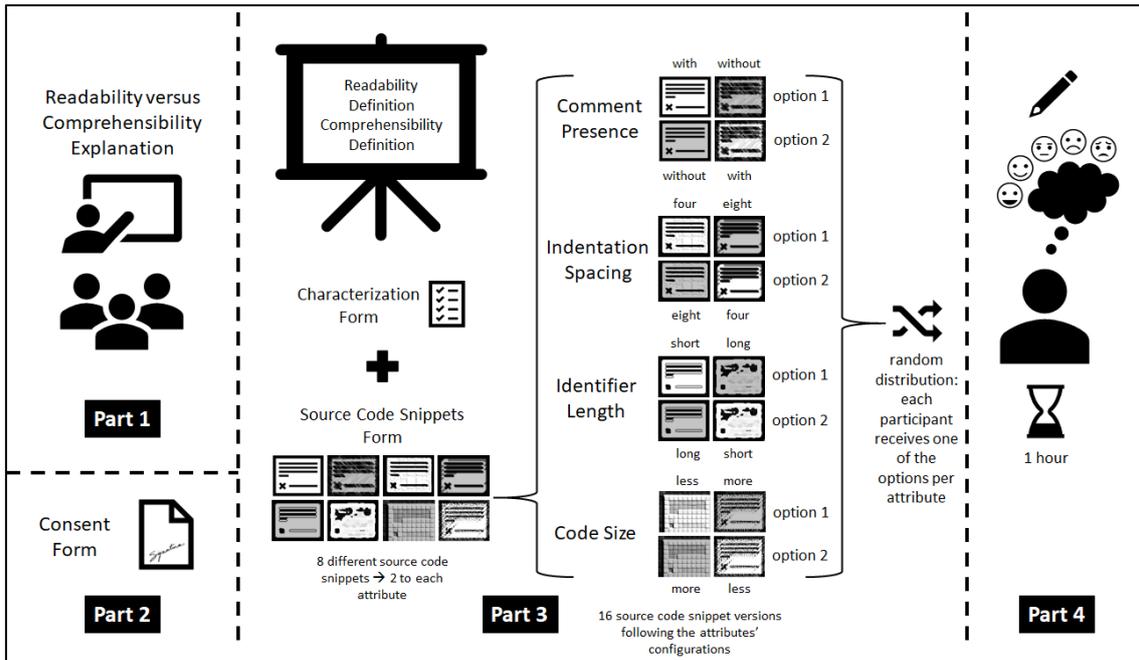
Figure 3.5 – Illustration of Local Studies Execution

The characterization form used to distinguish novices and experienced programmers asked about software development experience, including code guidelines, writing, review, debugging, correction, and maintenance. Also, questions on Python language knowledge (the programming language used for the experiments) were asked. For the stratification by novices and experienced, we decided to use the students' self-reported experiences (as recommended by Siegmund et al. (SIEGMUND *et al.*, 2014)) in the different experience dimensions captured in the Likert scale (ranging from 0 – no experience to 5 – vast experience).

The two quality characteristics (readability and comprehensibility) were evaluated independently, and different versions of the selected source code snippets were created to avoid confounding results. Eight pieces of paper portraying eight Python snippets (two per attribute assessed) printed one snippet per paper in landscape orientation represent the source code snippet form. Below each source code snippet (Figure 3.6a/ Figure 3.7a), smileys (five points Likert scale: very satisfied, satisfied, neutral, dissatisfied, very dissatisfied) were presented as a replacement for written Likert levels codes to capture the perceptions of readability (Figure 3.6b/ Figure 3.7b) and comprehensibility (Figure 3.6d/ Figure 3.7d) related to the source code under evaluation. It was also requested explanations concerning the reasons for the readability perception choice – we added the source code attribute as one of the options in the explanation box – (Figure 3.6c/ Figure 3.7c), and an answer concerning a question related to the code comprehension – we made the comprehension question be related to the attribute

of our interest – (Figure 3.6e/ Figure 3.7e). While the perceptions would provide a subjective opinion on the source code quality, the explanation mentioning the attribute under evaluation and the right answer to a comprehension question would corroborate that the perception provided concerned the attribute under assessment.

Figure 3.6 – Version 1 of One of the Two Source Code Snippets Used for Analyzing the Impact of Identifier Length on Readability and Comprehensibility

Figure 3.7 – Version 2 of One of the Two Source Code Snippets Used for Analyzing the Impact of Identifier Length on Readability and Comprehensibility

The results of all three empirical studies were analyzed independently and in their aggregation form, considering results from novices separately from experienced programmers. The results were aggregated using an odds ratio for effect size statistics (LIPSEY and WILSON, 2001). The full explanation of the planning of this local studies aggregation can be found in (RIBEIRO, SANTOS, and TRAVASSOS, Submitted).

3.3.2.2 Local Studies Aggregation Results Overview

The three studies collected quantitative and qualitative data from 66 software programmers, represented by undergraduate and graduate students, regarding their opinions on reading and understanding Python snippets. Table 3.7 presents the summary of the findings with statistically significant differences concerning the configurations of the attributes on either source code readability or comprehensibility. We identified an impact as positive or negative whenever a significant difference was encountered for either the perception or the explanation/answer concerning the attribute under evaluation on the source code readability/comprehensibility.

Although the quantitative analysis presents statistical significance related to the comprehensibility of commented source codes and the readability of long identifiers, it does not reveal any information that can contribute to solving the contradictions in the technical literature. Moreover, the data analysis did not support our initial assumption that experience could explain the contradictions in the technical literature since the results (especially the aggregated ones) are similar for studies with novices and experienced ones. Nevertheless, the isolated studies could provide some hints about the contradictory findings.

Unexpectedly we could not reject the null hypotheses regarding perceptions and explanations of the comments' presence for one of the studies (S2). To understand the reasons for this finding, we conducted a post-study session with the participants of S2 in which we presented the study design and results and asked them if they could explain the unexpected outcomes. In general terms, it seemed that they deliberately did not read the comments in the Python snippets. For instance, one of the participants mentioned that "*nobody trusts in comments*" – because, in his industrial experience, comments are usually outdated. Another participant called attention to the artificiality of the comments: "*it did not look real; it was too detailed.*" Indeed, the impact of their negligence could be observed in the answers to the comprehension questions. This finding corroborates with Salviulo and Scanniello (SALVIULO and SCANNIELLO, 2014), which mentioned in their ethnography analysis that practitioners often ignore comments. Another interesting result was presented in (NIELEBOCK *et al.*, 2019), in which practitioners were more prone to solve tasks correctly when dealing with source code without comments. The participants said that comments do not always tell the truth about the code.

In the case of the length of identifiers, a thorough analysis of the readability explanations given in S1, S2, and S3 shows that, even though the readability and comprehensibility concepts were differentiated beforehand, some participants used

them interchangeably, especially for novices. For instance, participants mentioned that "the identifiers are not intuitive" and "the identifiers are ambiguous," showing some misinterpretations as to what was expected as readability explanations. We cannot assure that these concepts were not misinterpreted in the existing studies in the technical literature, which can also help explain the contradictory findings. Still, despite these potential misinterpretations, the aggregated analysis regarding the explanations showed that both novices and experienced programmers preferred long identifiers. Hence, independently of the adopted perspective – i.e., readability or comprehensibility – long identifiers seem to provide a higher 'sense of understanding,' especially for novices. We hypothesize that experienced programmers are more used to the abbreviations used in the source code snippets since they follow common patterns (e.g., abbreviating the first character of words) usually found in real settings. Previous works have shown that some attributes are so widely used as identifiers for specific purposes that there is little difference between them and long descriptive identifiers, especially for more experienced programmers (BENIAMINI *et al.*, 2017). It might be possible that this also happens to common abbreviations, as in the study we performed.

Table 3.7 – Summary of the Results per Study and Aggregation – Noves x Experiences and Readability (R) x Comprehensibility (C)

ID	Participants		Source Code			Comment Presence (C)	Indentation Spacing (R)	Identifier Length (R)	Identifier Length (C)	Code Size (R)	Code Size (C)
	Number	Experience	Source	Size	Language						
S1	19	Novices	real code	2-16 loc	Python	⊕ (green +)	⊖ (red –)	⊕ (green +)	⊗ (yellow ×)	⊗ (yellow ×)	⊗ (yellow ×)
S3	18	Novices	toy and real code	2-16 loc	Python	⊕ (green +)	⊗ (yellow ×)	⊕ (green +)	⊕ (green +)	⊗ (yellow ×)	⊗ (yellow ×)
S1	19	Experienced	real code	2-16 loc	Python	⊕ (green +)	⊗ (yellow ×)	⊕ (green +)	⊗ (yellow ×)	⊗ (yellow ×)	⊗ (yellow ×)
S2	10	Experienced	real code	2-16 loc	Python	⊗ (yellow ×)	⊗ (yellow ×)	⊗ (yellow ×)	⊗ (yellow ×)	⊗ (yellow ×)	⊕ (green +)
Agg. S1+S3	37	Novices	toy and real code	2-16 loc	Python	⊕ (green +)	⊗ (yellow ×)	⊕ (green +)	⊗ (yellow ×)	⊗ (yellow ×)	⊗ (yellow ×)
Agg. S1+S2	29	Experienced	real code	2-16 loc	Python	⊕ (green +)	⊗ (yellow ×)	⊕ (green +)	⊗ (yellow ×)	⊗ (yellow ×)	⊗ (yellow ×)

In summary, even the most cited moderator among the studies (programming experience) did not support explaining the encountered contradictions. Performing a study for each of the other mentioned ones (and mediators), apart from being time-consuming, could lead to similar conclusions. It is certainly an underwhelming result since it looks like we are far from identifying what is relevant to report in our studies to support their understanding and use. While reporting every piece of information possible is an alternative, too much extraneous information, besides not helping make decisions, can make it difficult to understand the studies' conclusions.

3.3.3 Problems in Understanding and Using the Information

The combination of the findings was not an easy task to perform, and it did not provide a full understanding of the topic, given that we could not identify the roots for the contradictory findings, even after conducting a family of studies and aggregating their results on the conjectured ideas from the initial works. It is challenging to build an evidence-based software technology to support improving source code quality concerning its readability and comprehensibility. We believe this challenge exists in many other SE topics.

While using a single result to answer some practitioner questions in the industry might be more straightforward, this action can be misleading not only because a single study might lack important information to support decision-making but also because there might be other results on the topic that lead to different directions. Aggregated studies can enlighten the contradictions to some extent, but as presented, the studies found in the technical literature might be based on concepts that can make them hard to combine. Moreover, performing complete planning, execution, and aggregation of studies can be time-consuming and might not provide enlightening findings for the initial purpose.

During the analysis of the scientific works, we could identify some barriers to understanding and using scientific information related to two different moments: 1) extracting and grouping information; 2) identifying the reasons for the contradictions:

1. Extracting and Grouping Information:

    i. <u>distribution of important information throughout the report</u> – we had to employ a significant amount of time to identify the information we were looking for in each report, basically because they were scattered in many different parts of the paper;

ii. differences in understanding of source code reading and comprehension – some studies did not provide the perspective used concerning the quality characteristic under evaluation, and some others that provided definitions for the quality characteristics under assessment used the concepts of readability and comprehensibility interchangeably;

iii. inconsistency of measurement procedures for a same-name measure – some studies described a same-name measure using different measurement procedures, which prevented us from aggregating their findings;

iv. low amount of details on the measurement procedures used – many studies did not present enough information on the measurement procedures used for the cause-effect relationship investigation, which prevented us from aggregating results with similar measurement procedures.

2. Identifying the Reasons for the Contradictions:

i. little to no explanation about the phenomenon under investigation – very few works tried to explain the phenomenon they were investigating, and this fact led to some controversial observations, such as the analysis of the effect that the number of periods (BUSE and WEIMER, 2010) and commas (BUSE and WEIMER, 2010) (TASHTOUSH *et al.*, 2013) in a program unit have on source code reading and comprehension.

ii. low amount of information on contextual information affecting the results – most of the works presented the results by merely reporting the obtained statistical data; and while this presentation can be enough when several study reports suggest the same result, it is not sufficient when comparing two different works results with the expectation of identifying which of the two apply to a specific scenario. Furthermore, though some works presented contextual information along with the statistics, most did not explain how the contextual information affected the results.

Four main challenges can support explaining the mentioned barriers to using scientific knowledge to build or support the selection of software technologies in SE. They are the *lack of*:

- standard presentation of the information especially concerning definitions and the phenomenon under investigation that prevented us from quickly extracting what we were looking for;

- standard terminology for SE that prevented us from identifying what exactly was being assessed in the works;

- theoretical studies in SE to explain the SE phenomena and ground the empirical investigations in the field; and

- guidance on reporting measurement procedures, mediators, and moderators of cause-effect and correlation relationships that would enable their use by others;

- a repository for knowledge sharing since we know it is complicated to report all the necessary information supporting the understanding of scientific knowledge in a limited number of pages.

In addition to our findings, Wöhlin and Rainer report several issues with producing, consuming, and disseminating evidence in the SE field (WOHLIN and RAINER, 2021). Some of them are worth mentioning, such as the producer's risks for misinterpretations, the inconsistency of presentation of evidence across multiple works from the same authors, and the misunderstanding of evidence by consumers. All these issues highlight the problems with understanding evidence that any consumer (researcher or practitioner) would have while in contact with scientific works.

3.4 Discussions and Conclusions of this Chapter

Performing rigorous scientific procedures can be tricky, even for those experienced in research methods. The exploratory study conducted with novice researchers provided evidence of pitfalls while searching for scientific knowledge related to, among other factors, the inexperience of the participants in research methods. It is important to stress that SE practitioners are not used to performing such procedures, which makes them in a worse position when compared to the study participants – at least concerning the research method knowledge/expertise.

The amount of reasoning required to extract, interpret, and synthesize useful scientific knowledge from scientific productions in a problem-solving situation is significant, which should not happen if scientific productions were intended to provide information to practitioners. The barriers encountered during using scientific knowledge to produce evidence-based software technology were identified by a researcher, who is more used to following scientific procedures than practitioners. It means that not only

might the use of scientific productions be unattractive to practitioners, but it can also be challenging to researchers in collaboration with the software industry.

Although it is not hard to imagine researchers gathering, selecting, extracting, analyzing, and aggregating evidence scattered in hundreds of scientific articles, these activities are not natural for practitioners. The challenges related to the lack of standard terminology in SE, clearness, and completeness of research reports, standard presentation of the information, and repository for knowledge sharing impose additional difficulties in searching, understanding, and using scientific knowledge by practitioners and require an investment of significant time. After all, they can hamper the understanding of scientific knowledge by those that not only are far from the research context but also communicate using a different language from research (GAROUSI, PETERSEN, and OZKAN, 2016), and through different channels of communication.

The challenges identified cannot be easily overcome in the SE field. While we can try to provide some guidance to overcome some of them in the long term, this would require combining efforts from the whole SE research community. Thus, practitioners should not be looking for scientific productions in scientific repositories – not until a set of challenges presented in the SE research field is overcome – but rather, researchers should be the ones to take science to practice, not necessarily the same way is done in research, that is, not using scientific language to present information that is strictly relevant to researchers in articles that are hosted in scientific repositories.

Concerning the information that should be provided to practitioners, its presentation, and placement, we believe that a more straightforward approach to deal with these issues would be looking for what SE practice does in this respect. "From a research point of view, successful technology transfer requires knowing which information decision-makers in the industry need, and where they look for it" (JEDLITSCHKA *et al.*, 2007). When searching for information, practitioners primarily use internal colleagues, textbooks, industry workshops, articles in practitioners' journals, and the internet (JEDLITSCHKA *et al.*, 2007). In the next chapter, we discuss what should be transferred to practice and how this should be done.

4 Understanding the Information Needs of Software Engineering Practitioners

"Everyone hears only what he understands." – Johann Wolfgang von Goethe

4.1 Introduction

Continuing the investigation on the main reasons for the intended users not to go after and use information by Bennett and Jessani (BENNETT and JESSANI, 2011), two reasons remain to be investigated: iii) they do not care and see the information as irrelevant, not beneficial to their agenda; and iv) they do not agree, think the information is misguided or false.

In this chapter, we introduce works that present the opinions of SE practitioners towards SE scientific knowledge; and investigate what practitioners consider relevant and credible concerning SE knowledge in practice. In addition, we present two investigations on SE practical questions and answers from an important Q&A forum of SE that are intended to identify the information that seems important to SE practitioners. Finally, the results presented in this chapter allow us to answer the following research question (see Table 4.1):

Table 4.1 – SRQ2 – Secondary Research Question 2

SRQ2: What are the practitioners' information needs that can be used to guide empirical studies on SE?
The answer to this question allowed us to identify the: - relevant SE practical issues and common types of issues raised by practitioners   - key contextual information used by practitioners to share practical knowledge and common types of information important to practitioners assess shared knowledge

4.2 The Scientific Knowledge Relevance and Credibility to Practice

Several works have investigated the relevance of SE scientific knowledge to practitioners. For example, Rainer, Hall, and Baddoo reported an exploratory work in which they identified a misalignment between the importance developers say evidence has and their consideration of using evidence when said so. They mention that even in

the face of strong arguments from researchers on a certain SE phenomenon, the practitioners would not be convinced that the phenomenon would happen in their context (RAINER, HALL, and BADDOO, 2003), despite the fact they believed scientific evidence is an important asset to practice. It might be related to a phenomenon observed in industrial settings by Passos et al., in which the authors identified that "people act mostly based on their strong beliefs and that those are obtained from personal experience" (PASSOS *et al.*, 2011). It means that even in the face of strong evidence, there might be chance practitioners can lean towards what they have experienced before.

In another work about the relevance of scientific knowledge, the participants acknowledged the academic literature. However, they never go after them when looking for answers because "although researchers address relevant problems, that relevance is often buried in the details" (BEECHAM *et al.*, 2014). The authors argue that this practice-research paradox exists because of scientific knowledge's accessibility (the reports are not comprehensible to practitioners), credibility (practitioners do not write the reports), and relevance (the reports do not focus on the main results). The authors also discovered that practitioners praise expertise in the topic they want to support the most (BEECHAM *et al.*, 2014). However, when they cannot directly reach a peer who performed what they want or used a software technology of interest, they search for opinions and experiences in blogs, wikis, and their organization intranets.

Lo, Nagappan and Zimmermann conducted a large-scale survey with 512 practitioners to capture their perception of the relevance of SE research. The SE research was assessed through ideas from research articles published in ICSE, ESEC/FSE, and FSE. The practitioners should label scientific ideas as essential, worthwhile, unimportant, or unwise. Overall, the results favor SE research, with more than 70% of the ratings mentioning that the research ideas are essential or worthwhile (LO, NAGAPPAN, and ZIMMERMANN, 2015). However, this survey highlights why some research ideas are considered unwise. Among them, we can point the opinions regarding i) the irrelevance of empirical studies and/or their findings; ii) the mild benefit of a proposed software technology opposing its usage and maintenance costs; iii) the disagreement towards the assumptions about inputs or conditions used in some work; iv) the disbelief in particular software technology, and v) the side effects of a proposed solution.

Carver et al. replicated the survey on ESEM research articles with 437 practitioners. This time, the positive rating reached 67%. In addition, the authors collected information on the problems practitioners thought the research community should focus. Among specific problems mentioned by the practitioners in their

responses, many provided generic replies such as "Focus on solving real problems the industry has" (CARVER *et al.*, 2016). While these generic responses can highlight practitioners' misunderstandings of the question, it can also be the case that practitioners might perceive there are major problems to be solved in practice that are not being dealt with in research.

When analyzing scientific knowledge's relevance to practice, knowing whether practitioners cite research articles can be an interesting way of gathering this information. A survey study performed by Williams (WILLIAMS, 2018) showed quantitative and qualitative results on the matter, which led to the conclusion that practitioners cite very little research while reporting knowledge to practice.

On the lines of looking for what the grey literature has to say about the relevance of SE research to industrial practice, a recent multivocal literature review was performed. The authors provided a holistic view of the topic, mentioning some of the roots causes of low relevance being (GAROUSI, BORG, and OIVO, 2020): i) the simplistic views/wrong assumptions about SE in practice; ii) the lack of connection with industry; iii) the wrong identification of research problems; iv) the disregard of the cost-benefit of applying SE technologies; v) the "advocacy" research ("research-then-transfer" approach), and; vi) the research focus being too generic and not considering context; among others.

The most recent work on the topic was a survey with 153 practitioners on the relevance of scientific papers related to requirements engineering (FRANCH *et al.*, 2022). Concerning the requirements research ideas, almost 70% reached more positive than negative ratings. An interesting finding was along the lines of results from previous works in which respondents perceived papers as being more relevant whenever a practitioner was involved in them and less relevant whenever an academic professional was involved in them. When asked about the reasons for the negative perception of some research works, the practitioners mentioned, among other reasons, the lack of necessity (e.g., not needed generally/in their context, old-fashioned, and others) and the difficulty (e.g., unrealistic, too specific, not efficient, among others) in applying the knowledge in their settings.

Overall, practitioners believe research produces interesting ideas/proposals, but scientific knowledge is not among their main needs. The fact that they measure a scientific production's relevance (or even credibility) based on whether a practitioner took part in it says a lot about their perception of research produced in labs. The negative comments on research works are usually related to the lack of necessity of it to practice,

the lack of fitness to practice, the difficulty in applying it in practice, the real impact of it to practice, and the differences in the context of its application to what exists in practice, among others.

In summary, the problem relevance, the solution utility, the solution impact, and the source of the data/results (whether they are from practitioners or not) are important to practitioners when assessing scientific knowledge as relevant and credible. (FRANCH *et al.*, 2022).

4.3 What is Relevance to SE Practitioners?

In some sense, when practitioners report the reasons for not thinking a scientific production is relevant, they state what is important for them in a more generic sense. Some of the previous works also collected practitioners' opinions on what research should deal with, and other new ones tried to collect this information, meaning they wanted to specifically know what is important for practitioners in terms of SE topics.

A total of 12 categories of questions SE practitioners are curious about were created by Begel and Zimmermann after surveying 203 responses (BEGEL and ZIMMERMANN, 2014). Although the questions are related to interests in data science investigation inside the practitioners' projects, they can provide insights into what software engineers think is important to know. They are:

- Bug Measurement: related to bug location (code and design), most common bugs, phases in which bugs are commonly identified, and cost of fixing bugs, among others.

- Development Practices: related to code-oriented practices, including debugging, refactoring, code reviewing/ commenting/ documenting.

- Development Best Practices: related specifically to the best and worst ways of performing a software practice.

- Testing Practices: related to software testing, including testing automation, testing strategies, unit testing, test-driven development, test coverage, test case, and testing process.

- Evaluating Quality: related to code optimization tradeoffs, complexity metrics, code duplication, test coverage vs. feature usage, legacy software, and aging codebases.

- Services: related to cloud development, operations, and performance measurement.

- Customers and Requirements: related to customers' interests, such as important features, backward compatibility, the impact of testing in production, and the cost of specifications and design.

- Software Development Lifecycle: related to time allocation in all phases of a software lifecycle.

- Software Development Process: related to software methodology, such as agile and pair programming.

- Productivity: related to metrics for measuring the productivity of software developers.

- Teams and Collaboration: related to team size required to deliver a certain feature size, the number of people playing each expected role, and practices to improve collaboration and knowledge sharing, among others.

- Reuse and Shared Components: related to source code reconstruction or reusability time.

More specifically to what practitioners think research should focus on, Carver et al. (CARVER *et al.*, 2016) list four categories:

- Software: related to code quality, sustainability, architecture, software design, and usability.

- Process: related to testing, estimation, process improvement, success/failure of projects, and cost/benefit analysis of technologies.

- Developers: related to productivity, communication and coordination, people, and knowledge management.

- Users: related to requirements and customers.

In another survey conducted with practitioners (67) by Ivanov et al. on the importance of SE topics, participants mentioned they primarily care about estimation, development productivity, and techniques related to code review, coding standards, testing, and DevOps (IVANOV *et al.*, 2017).

There are several ways of collecting what is relevant to practitioners. Most of the works available in the SE technical literature use surveys with practitioners to achieve such purposes, either on general topics or specific ones. Previous works have shown that the grey literature can provide good insight into practical fields, such as SE (GAROUSI, FELDERER, and MÄNTYLÄ, 2016). Given the issues encountered by

practitioners concerning the relevance and credibility of scientific knowledge, we wanted to investigate the subject further by looking at what practitioners share among themselves concerning SE topics in the grey literature. We believe that by looking at their knowledge sharing on the internet, we can identify what is important to them and, thus, provide means to guide research in SE to focus on more practical issues.

To perform a broader analysis of the relevance and credibility of SE topics/reports to practice that is not based on practitioners' claims but their behavior, we decided to analyze posts (questions) from the Stack Exchange community regarding SE topics. Although we could have used practical blog posts, online articles, and video talks, we decided to use Stack Exchange as a source for SE practical knowledge (and issues), believing it hosts a representative amount of SE practical knowledge given its success amount SE practice, especially when it comes to its main forum Stack Overflow.

4.3.1 Empirical Study Planning

4.3.1.1 Research Goal

The goal of this study is twofold. First, we want to identify the topics and related questions relevant to SE practitioners. Second, we want to identify information in the questions that can be used to guide research investigations on them. Using the GQM approach (BASILI, SELBY, and HUTCHENS, 1986) (BASILI, CALDIERA, and ROMBACH, 1994), the objective of this study is as follows:

Analyze the content of practitioners' questions

For the purpose of characterization

With respect to the SE topics and question patterns

From the point of view of SE researchers

In the context of Stack Exchange questions raised by practitioners while dealing with their problems in software development and maintenance activities.

4.3.1.2 Research Questions

RQ1: What are the SE topics and related questions relevant to practitioners?

RQ2: What information can be identified in the practical questions that can be used to support research investigations?

4.3.1.3 Dataset Overview

Stack Exchange is a platform that offers a collection of 176[6] forums for Q&A about a variety of topics (see Figure 4.1). Its main forum, Stack Overflow, has more than 21,286,479 questions and 31,692,495 answers on several topics related to software development and maintenance, especially programming. In addition, Stack Exchange provides data dumps of data generated by its users in all its forums, including all questions (see Figure 4.2) and their information: tags, comments, answers, creator, score, view count, answer count, and favorite count, among others. For this study, we downloaded a data dump containing data from June 2008 (ever since its launch) to June 2021 (our last update for the study) (STACK EXCHANGE COMMUNITY, 2014).
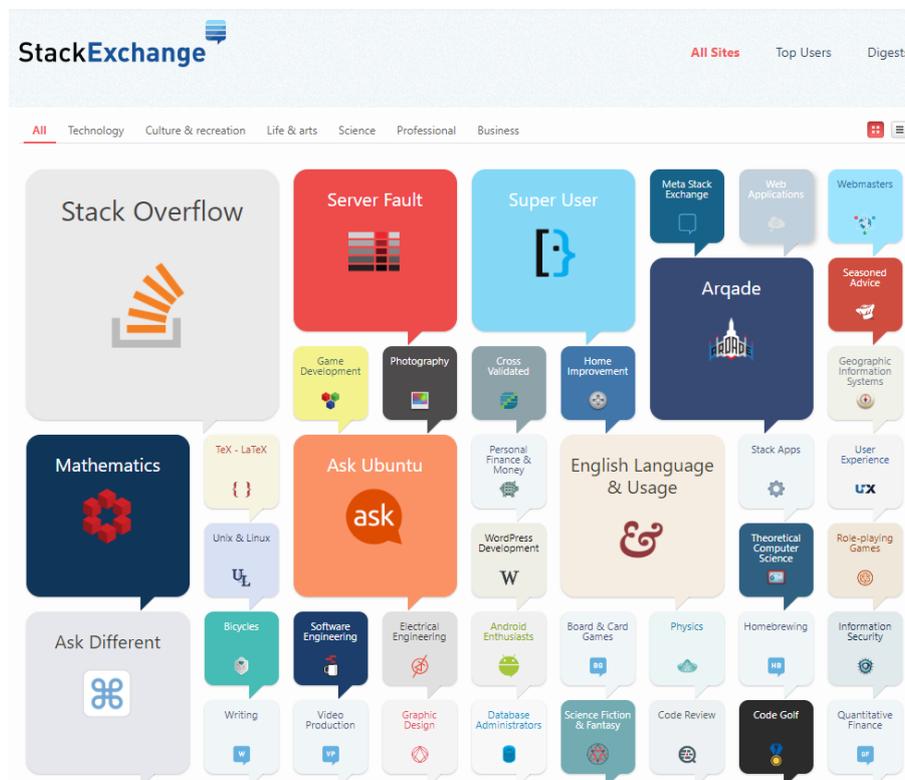
Figure 4.1 – Ranked Forums from Stack Exchanged (Source: (STACK EXCHANGE COMMUNITY, 2008))

---

[6] Numbers provided are related to the data dump from June 2021.
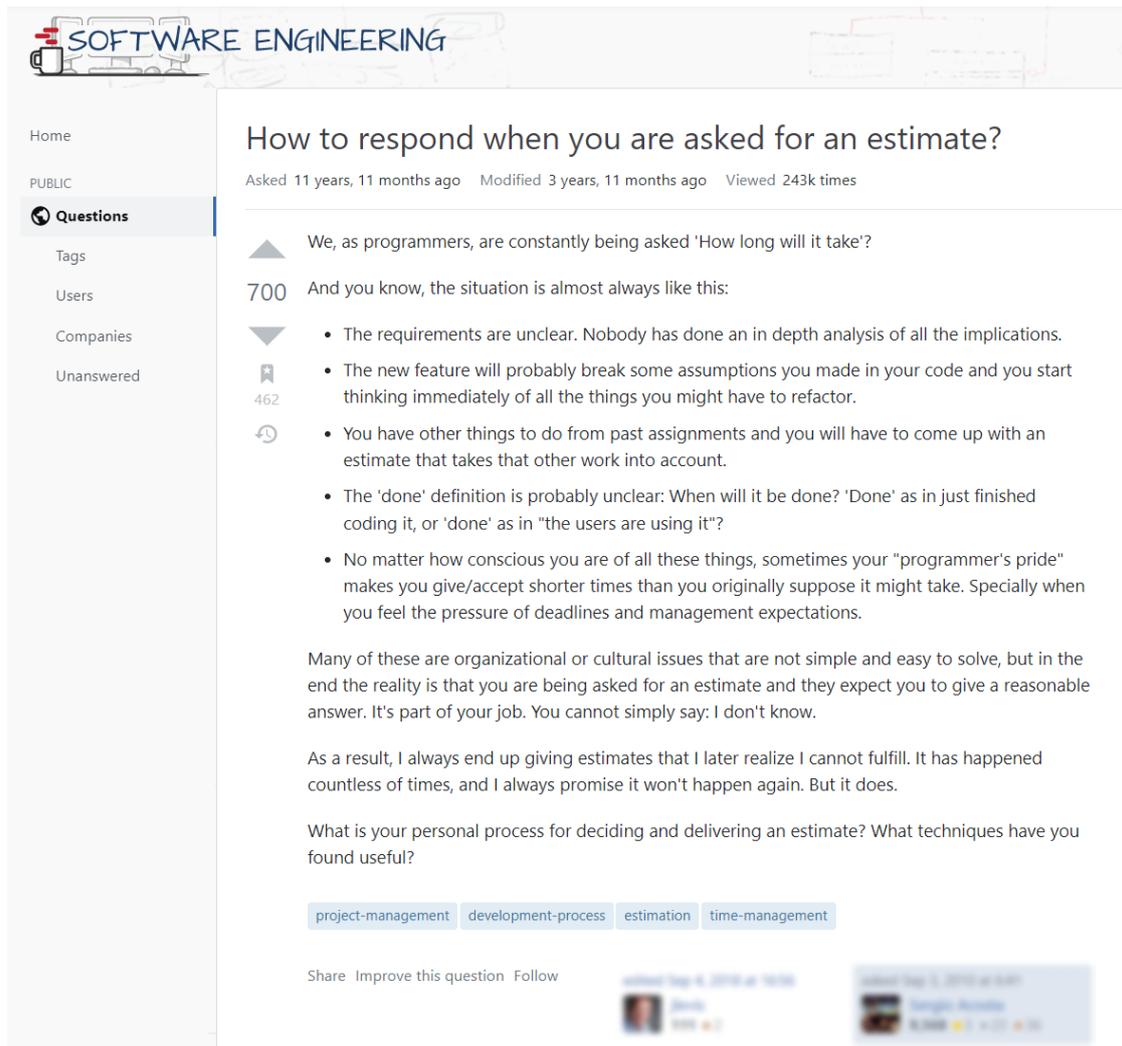
Figure 4.2 – Example of Question Posted on the Software Engineering Forum (Source: (STACK EXCHANGE COMMUNITY, 2008))

The example of the question from Figure 4.2 was extracted from the forum of Software Engineering, in which practitioners usually share problems and experiences about various topics related to software lifecycles. The questions raised in the Software Engineering forum are more related to state-of-the-art ones than those made in Stack Overflow. However, in terms of size, Stack Overflow holds more than 87GB of data from posts and users (not counting data from badges, comments, history, links, tags, and votes), while the Software Engineering forum is smaller, holding less than 500MB of data from posts and users.

To guarantee a certain quality of posts (questions and answers) in Stack Exchange forums, experienced users with a high reputation in the forums (moderators) review low-quality posts automatically filtered by the system and provide feedback on the acceptance, closure, and deletion of them.

4.3.1.4 Data Sampling

Certainly, it is unfeasible to analyze all the questions from any available forum manually. Therefore, we had to plan a strategy for narrowing the number of questions that should be analyzed without hampering the study's main goal. Figure 4.3 presents an overview of the steps taken and their outputs which are all described in the next sections.
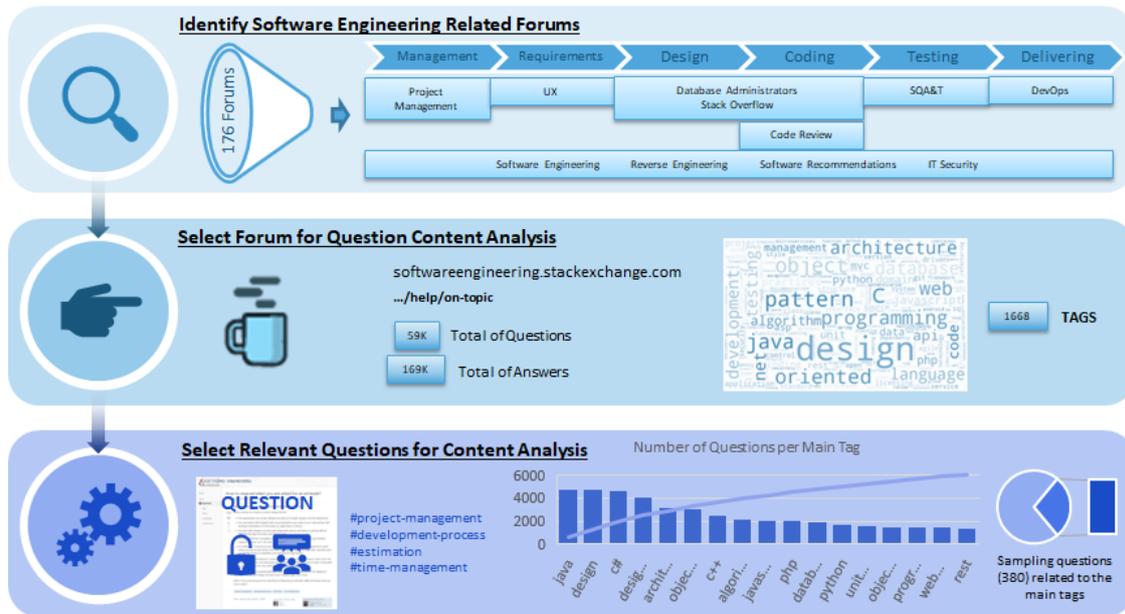
Figure 4.3 – Steps Followed for Question Sampling

4.3.1.4.1 Identify Software Engineering Related Forums

Not all forums in Stack Exchange are related to SE topics. Therefore, to properly analyze the questions made by software engineers, we first had to select the forums that comprised SE topics concerning discussions on activities related to software development and maintenance.

Each forum from Stack Exchange has its guidelines – inside its help center – for supporting users on the topics accepted for discussion inside the forum and on the questions that are not considered valid. Therefore, we analyzed the descriptions of the forums, their target audience, and, in some cases, their post guidelines. In total, 176 forums from Stack Exchange were analyzed, and whenever there was a doubt on whether the forum was related to SE, the "/help/on-topic" and "help/don't-ask" of the forum were also analyzed. Forums concerning specific domains and technologies were not considered for selection since their purposes align with discussions on using specific software technologies.

The following forums were selected as they relate to different phases of the software lifecycle (see Figure 4.3 and Table 4.2):

Table 4.2 – Forums Related to SE Topics

Forum	Total of Questions	Total of Answers	Total of Tags
Project Management	5,868	17,724	287
UX	30,258	79,130	1,028
Database Administrators	90,485	121,698	1156
Stack Overflow	21,286,479	31,692,495	61,059
Code Review	71,933	110,887	1,102
SQA&T	10,822	22,977	460
DevOps	4,285	5,612	418
Software Engineering	59,351	169,049	1,668
Reverse Engineering	8,042	9,694	339
Software Recommendations	20,790	20,491	964
IT Security	62,495	109,979	1,219

4.3.1.4.2 Select Forum for Question Content Analysis

Even upon the reduction of forums for analysis, the number of questions in each was too high for qualitative analysis (our intention), so a more feasible approach would be selecting one forum for the question content analysis. One option would be using the most popular of them – "Stack Overflow." However, Stack Overflow has a considerably high number of questions that expect "state-of-the-practice" answers, such as source code solutions, which, although accepted as questions/answers inside the forum[7], are not the type of questions target of this research.

We decided to use the Software Engineering forum because of three main reasons: i) it has an important amount of questions after Stack Overflow; ii) it holds questions on a variety of SE topics, and iii) it is possible to find questions that are usually made or to which answers can be found in "state-of-the-art" of SE (see example in Figure 4.2).

---

[7]https://stackoverflow.com/help/on-topic

4.3.1.4.3 Select Relevant Questions for Content Analysis

Since the goal is to identify relevant SE topics and related questions for practitioners, not all forum questions represent this relevance. From our point of view, a question is relevant whenever: i) it matches the forum's expectations; ii) it is within the most talked topics of the forum; iii) practitioners like it, and iv) practitioners try to provide answers for it. We understand that there are other perspectives of relevance, and a single question might represent an important issue to be addressed by research. Still, we had to handle numerous questions that had to be relevant and assessed to prioritize them.

Among the existing information for a question, it is possible to identify whether it was closed by moderation. Moderation closes a question whenever it is duplicated or does not follow the forum's guidelines. Thus, we could only gather questions that matched the forum's expectations by looking at open questions.

Relevant questions would be about relevant topics of the forum. Each question can be tagged with terms (usually more than one) created by users of the forum to summarize its content. We ranked the 1,668 tags from the Software Engineering forum, considering the number of questions that mentioned each tag, and identified the 17th most used tags. The tags presented in Figure 4.4 represents the 99th percentile of all tags, meaning the number of questions tagged by any of the presented tags is bigger than the number of questions tagged by 99% of the existing tags.
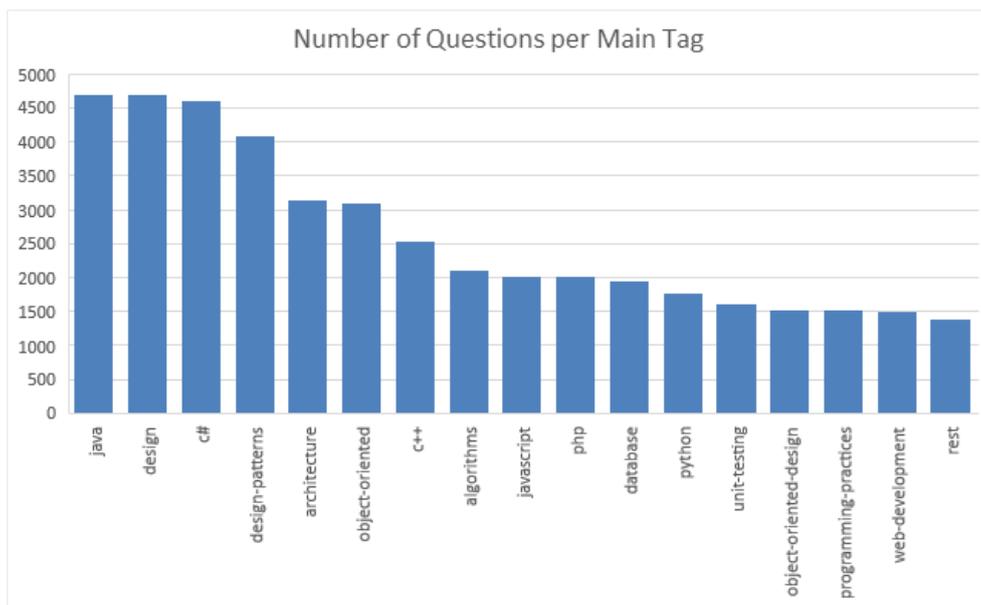
Figure 4.4 – Main Tags from Software Engineering Forum

The data dump from Stack Exchange provides measures for each question that can be used to identify those more relevant to practitioners. Such measures are:

- View count: number of times someone viewed the question;

- Score: calculated by the result of the number of upvotes minus the number of downvotes given to the question by users;

- Favorite count: number of times users followed the question so they could receive updates about it;

- Answer count: number of answers provided for the question, among others.

We believe that score and answer count can provide interesting indications that a question might be relevant to practitioners. Whenever the score and answer count were above zero, that would indicate the question draws some interest from the users (an assumption previously done in (PONZANELLI *et al.*, 2014) too). Out of the 59,351 questions on the Software Engineering forum, a total of 30,710 matched our specification of relevance for practitioners, meaning they were not closed, they were tagged by at least one of the main tags of the forum, they had more than zero as score, and they received at least one answer. On this last number of questions (30,710), we decided to apply a sample size calculation (KASUNIC, 2005) to identify the right number of questions that should be analyzed to provide 95% confidence in the results in describing the whole target population of relevant questions.

The sample size is calculated by the following formula $SS = \frac{Z^2 \times p \, x \, (1-p)}{c^2}$. *SS* represents the sample size, *Z* represents the Z-value established according to the confidence level, *p* represents the percentage of selecting a choice, and *c* represents the desired confidence interval. Considering the population is finite and has a size (*pop*), the following formula is the one to be used $SS_f = \frac{SS}{1+\frac{SS-1}{pop}}$. For a *pop* of 30,710 questions, using a confidence level of 95% (*Z*), 0.5 percentage selecting a choice in the worst case (*p*), and a confidence interval of 0.05 (*c*), the sample size of the question to be analyzed is of **379.42**. Since there is no 0.42 of a question, the sample size can be rounded to **380**.

The 380 questions are selected from the set of questions tagged by only one of the 17 main tags. In addition, we removed from the selection questions tagged by two or more tags in the set not to analyze duplicates and so that the analysis could focus on the particularities of each tag. Each tag is represented by a certain number of questions proportional to its popularity in the set (based on the number of questions marked with

the tag and not marked with other tags from the set). This removal slightly changed the tags' ranking compared to the general ranking that considers all questions (including duplicates). See Figure 4.4 and Figure 4.5.
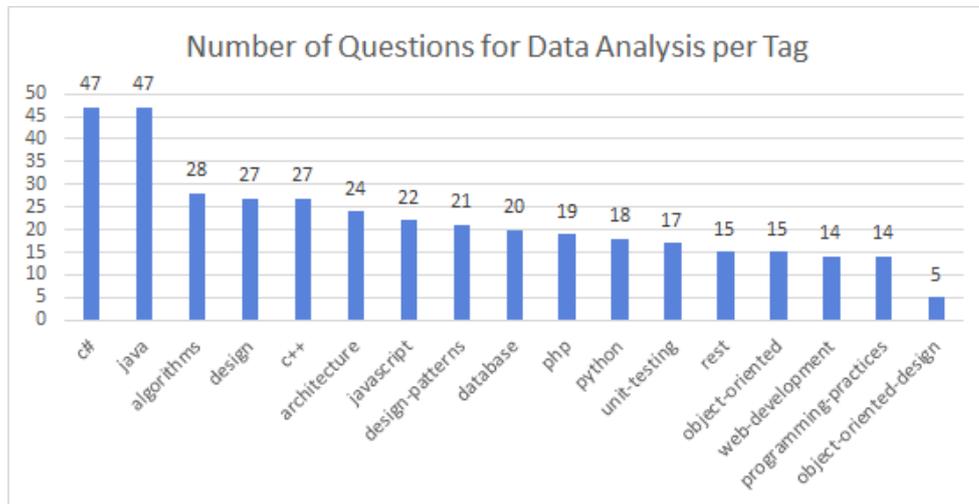
Figure 4.5 – Number of Questions Selected for Data Analysis per Main Tag

It is important to state that the questions selected in each tag are the ones that draw more attention from the community. Thus, the top questions concerning view counts can be selected for each tag.

4.3.1.5 Analysis Procedure

For extracting the SE topics and patterns from the questions, thematic analysis (BRAUN and CLARKE, 2008) is the qualitative method used. The choice for thematic analysis is because it is among the most used synthesis methods for qualitative data in SE (CRUZES and DYBÅ, 2011) and is useful for pattern (themes) identification from textual data.

Thematic analysis/synthesis is a method for identifying, analyzing, and reporting themes related to textual data (BRAUN and CLARKE, 2008). According to the method, a theme captures and summarizes something important about the data that supports answering the research question. Themes can capture responses from different levels of abstraction, and they usually appear in different parts of the data set. Before creating the themes, familiarization with the textual data and a coding process is required.

There are three ways of performing a thematic analysis: through the inductive approach (bottom-up), deductive approach (top-down), or integrated approach (partway between inductive and deductive approach) (CRUZES and DYBA, 2011). While performing an inductive approach, the themes are more attached to the data as no data is forced to fit a particular code/theme. In the deductive approach, a provisional "start

list" of codes is created to integrate concepts that exist in the data that are well-known by the researchers.

In this study, the integrated approach was selected to be executed, being the first approach to start the inductive one to support the identification of the initial codes and themes used in the deductive approach. Figure 4.6 provides an overview of the steps involved in applying thematic analysis (CRUZES and DYBÅ, 2011).
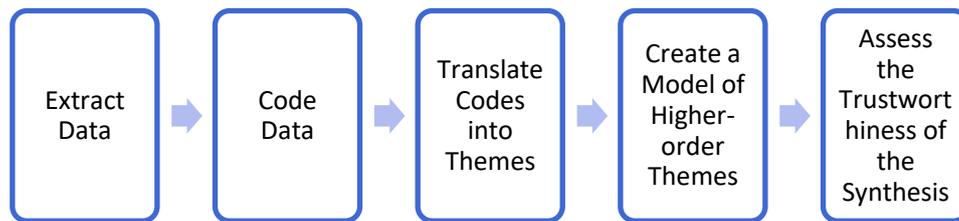
Figure 4.6 – Thematic Synthesis Steps – Inspired by (CRUZES and DYBA, 2011)

4.3.2 Empirical Study Results: Thematic Analysis of Questions Pattern from Stack Exchange

4.3.2.1 Formulate Research Questions to Part of the Practical Questions and Generate Initial Codes and Themes

To become familiar with the data and start an initial list of codes/themes, we applied the inductive thematic analysis approach to 25% of the 380 questions (98[8] practical questions). Figure 4.7 depicts the distribution of questions to be familiarized with versus the questions to be analyzed per tag.

---

[8] The actual number is 95. However, after the identification of questions to be familiarized per tag, we had to round the numbers.
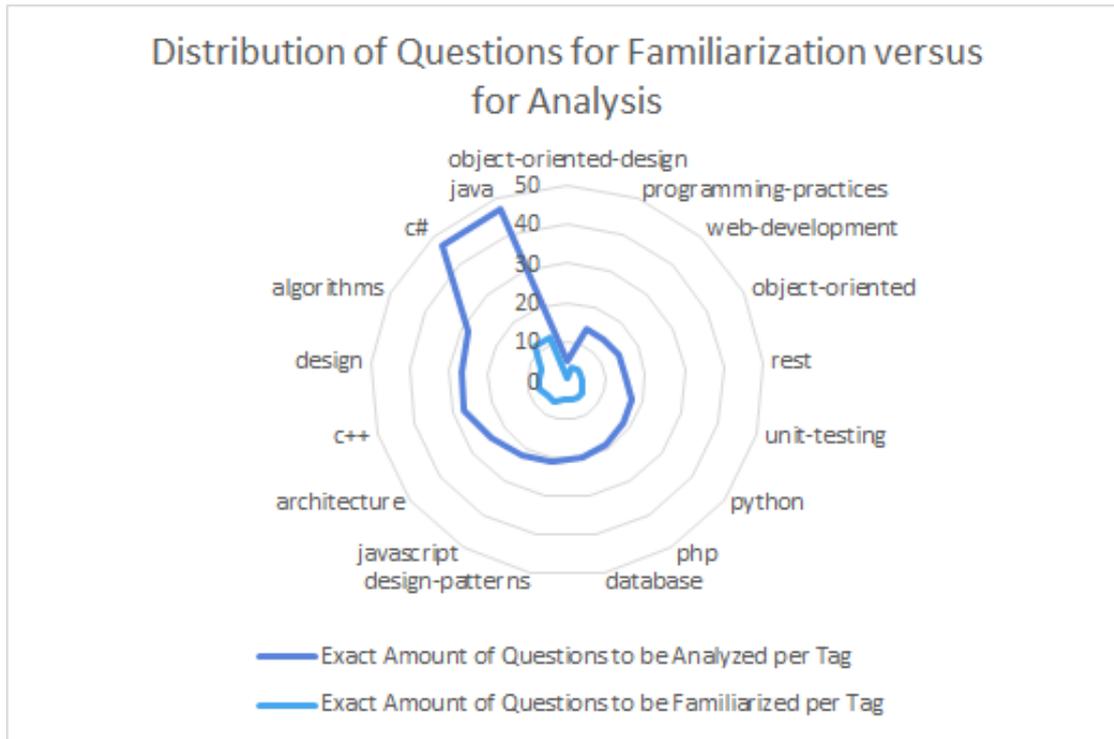
Figure 4.7 – Distribution of Questions to be Familiarized versus to be Analyzed

Just by looking at the tags, we can notice some interesting clusters. The clusters/concepts we identified from the tags are mostly related to the phases of design and coding of the software life cycle, even though the questions chosen for analysis are from the Software Engineering forum instead of Stack Overflow. See the following:

- Programming Languages: related to the tags C#, Java, C++, JavaScript, PHP, and Python.

- Software Architecture and Design: related to tags design, architecture, design patterns, rest, and object-oriented design.

- Programming Practices: related to the tags algorithms and programming practices.

- Database: related to the tag database.

- Software Testing: related to the tag unit-testing.

- Object Oriented Development: related to the tag object-oriented.

- Web Software Development: related to the tag web development.

Concerning the familiarization task, we decided to create possible research questions by looking at each practical question's title, body, and tags from the familiarization set. Our intention with this approach was to identify any information that

would allow us to determine patterns on them. To identify patterns easily, we grouped questions related to the same concept to be analyzed together.

While producing the research questions, we noticed many practical questions not meant to be answered by research. Most of these practical questions were related to some particularities of software technology that could be found in the technical documentation – case of many questions from programming languages, software architecture and design, programming practices, and object-oriented development concepts. Other questions in this set regarded expectations for solutions for specific tasks that someone was performing, usually related to architecture or coding (certainly, these last questions were out of the scope of the Software Engineering forum). This fact led us to categorize the possible source of answers to each practical question between "state-of-the-art" and "state-of-the-practice." Certainly, the ones related to "state-of-the-art" would interest us the most. Still, we decided to formulate research questions for all of them without distinctions.

To produce research questions, we mostly relied on the body of the practical questions. It happened because, in most cases, the title was insufficient to understand the question, and other questions were also asked while describing a single instance. Figure 4.8 presents the research questions produced to the practical questions related to web software development and software testing. The id of each practice question is presented in front of each research question, and strikethrough questions represent "state-of-the-practice" ones. Notice that the same practical question can produce more than one research question.
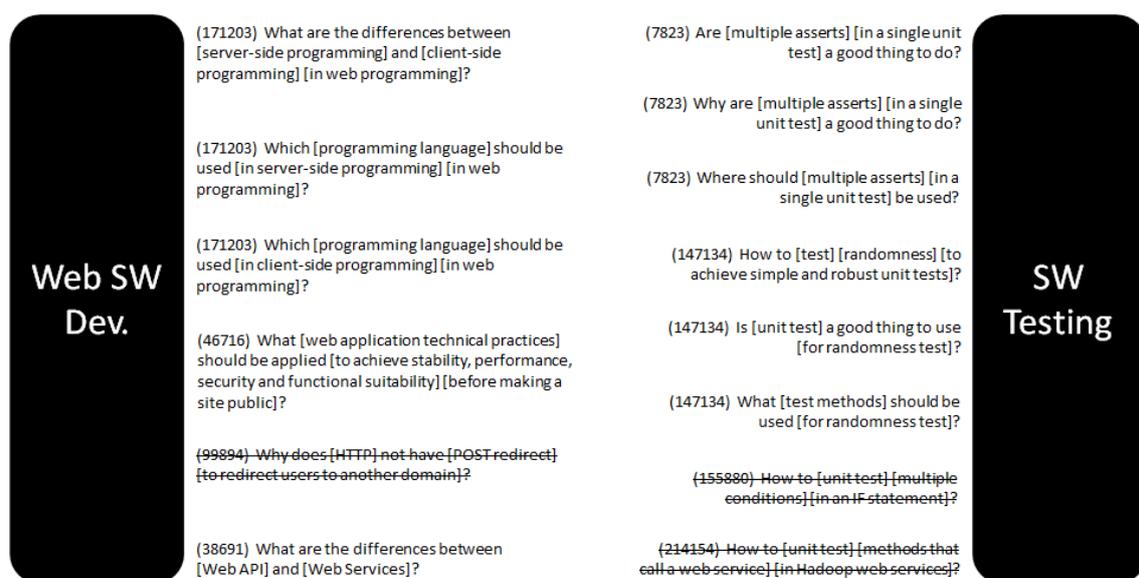
Figure 4.8 – Questions Created to the Concepts of Web Software Development and Software Testing During Familiarization

Similar questions were produced for all questions in the familiarization set. After finishing the process, we could identify interesting patterns in the parts of the questions that differentiated from one another. For example, their subjects, verbs, and complements could be mapped to population, intervention, comparison, outcome, and context from PICO/PICOC approach that is usually used to produce search strings in literature reviews (PAI *et al.*, 2004). Next, Table 4.3 presents the questions formulated for the concept of a database and the separation of information using the PICOC structure.

Table 4.3 – Questions Created to the Concept of Database During Familiarization

ID	Question	Intervention	Comparison	Outcome	Population/Context	Possible Answer Source
150669	Is [storing large files] [in the database] not a good thing to do?	storing large files	NULL	NULL	in database	state-of-the-art
150669	Why is [storing large files] [in the database] not a good thing to do?	storing large files	NULL	NULL	in database	state-of-the-art
150669	Is [storing large files] [in a file system] a good thing to do?	storing large files	NULL	NULL	in file system	state-of-the-art
150669	Why is [storing large files] [in a file system] a good thing to do?	storing large files	NULL	NULL	in file system	state-of-the-art
150669	Which [storing file method] should be used [to avoid slowing down data access] [in cases of large files]?	storing file method	NULL	to avoid slowing down data access	large files	state-of-the-art
190482	Why should [storing data in the database] be done over [saving data into disk]?	storing data in the database	saving data into disk	NULL	NULL	state-of-the-art
190482	Why is [database] used?	database	NULL	NULL	NULL	state-of-the-art
54373	Where should [non-relational databases] be used over [relational databases]?	non-relational databases	relational databases	NULL	NULL	state-of-the-art
54373	What are the differences between [non-relational databases] and [relational databases]?	non-relational databases	relational databases	NULL	NULL	state-of-the-art

54373	What is [non-relational database] for?	non-relational databases	NULL	NULL	NULL	state-of-the-art
~~120178~~	~~What are the differences between [MariaDB] and [MySQL]?~~	~~MariaDB~~	~~MySQL~~	~~NULL~~	~~NULL~~	~~state-of-the-practice~~
~~120178~~	~~What are the similarities between [MariaDB] and [MySQL]?~~	~~MariaDB~~	~~MySQL~~	~~NULL~~	~~NULL~~	~~state-of-the-practice~~
~~120178~~	~~What are the differences between [MariaDB query language] and [MySQL query language]?~~	~~MariaDB query language~~	~~MySQL query language~~	~~NULL~~	~~NULL~~	~~state-of-the-practice~~
~~288370~~	~~How to [synchronize] [databases] [to guarantee performance] [in two different SGBDs (from MySQL to DB2) with different and big schemas]?~~	~~synchronize databases~~	~~NULL~~	~~to guarantee performance~~	~~in two different SGBDs (from MySQL to DB2) with different and big schemas~~	~~state-of-the-practice~~

Appendix B of this thesis presents the list of the main questions created during this phase. After producing the questions and analyzing their structures, we identified that usually:

- subjects of questions are related to interventions practitioners want to know more about in a particular context; in most cases, they concern software technologies or specific characteristics of software technologies.

    - e.g., programming languages, web application technical practices, and test methods.

- compound subjects are also interventions, but they require comparison since they are related.

    - e.g., non-relational database and relational database; server-side programming and client-side programming.

- verbs of questions are related either to some task that needs to be performed (in this case, they can be the interventions themselves) or to the use/execution/application of interventions by practitioners.

    - e.g., storing large files; testing randomness.

- complements of questions are related to several things, such as the qualification of the interventions (population), the qualification of the environments in which the intervention is placed (context), or also to something practitioners want to achieve with the intervention (outcome).

    - e.g., large files; different SGBDs with different schemas; avoid slowing down data access.

Although not a rule, usually questions starting with "how" are "state-of-the-practice" questions, and those that present some information related to expectation for achieving something with the intervention (e.g., product quality characteristics) are "state-of-the-art" questions. Questions that compare two or more interventions are also strong candidates for "state-of-the-art" questions.

Based on the previous reasoning, we created a "start list" of codes to be used during the thematic analysis of the complete data. The "start list" comprises 16 meta-questions (see Table 4.4), five options for intervention/comparison (see Table 4.5), 13 options for the outcome (see Table 4.6), and 33 options for context (see Figure 4.9). The population is usually something that is embedded in the other structures, and for this reason, we do not provide any option for it.

While the meta-questions were created based on the initial research questions formulated during this phase, the remaining lists are from other works that somehow describe the information we expected for each identified structure. Therefore, at the end of this analysis, we want to produce up-to-date lists with only information that could be collected from the practical questions. It means that the initial lists will either reduce to remove information that was not presented in the practical questions or be enlarged with other specific information.

Table 4.4 – Meta-Questions Identified After Familiarization Phase

Meta-Questions
{Is | Are} <intervention> [not] a good thing to {do | use | apply | produce} [<to achieve this outcome>] [<in this context>]?
How should <intervention> be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?
How to <intervention> [<to achieve this outcome>] [<in this context>]?
What {is | are} <intervention> [<in this context>] for?
What {is | are} <intervention> [<in this context>]?
What <intervention> should [not] be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?
What are the {differences | similarities} among <intervention>, <comparison>, [... ,] and <comparison> [<to achieve this outcome>] [<in this context>]?
What are the {differences | similarities} between <intervention> and <comparison> [<to achieve this outcome>] [<in this context>]?
Where should <intervention> be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?
Where should <intervention> be {done | used | applied | produced} over <comparison> [<to achieve this outcome>] [<in this context>]?
Which <intervention> should be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?
Why {does | do} <intervention> [not] have <intervention> [<to achieve this outcome>] [<in this context>]?
Why {is | are} <intervention> [not] {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?
Why {is | are} <intervention> [not] a good thing to {do | use | apply | produce} [<to achieve this outcome>] [<in this context>]?

Why can [not] \<intervention\> be {done | used | applied | produced} [\<to achieve this outcome\>] [\<in this context\>]?	
Why should \<intervention\> be {done | used | applied | produced} over \<comparison\> [\<to achieve this outcome\>] [\<in this context\>]?	
Legend: | - operator to express the need for a decision among the options {} - mandatory structure; requires decision among the options [] - optional structure \<intervention\> - refers to a software technology/practice or a particular characteristic of it \<comparison\> - refers to a software technology/practice or a particular characteristic of it \<to achieve this outcome\> - refers to a particular goal that is intended to be achieved by the questioner \<in this context\> - refers to contextual information particularly important to the question asked	

Table 4.5 – Software Technologies Listed in (PFLEEGER, 1999) to be used as Intervention/Comparison Codes

Intervention	Description
Method	"It is a formal procedure for producing some result."
Paradigm	"A paradigm represents a particular approach or philosophy for building software."
Procedure	"A procedure is like a recipe: a combination of tools and techniques that, in concert, produce a particular product."
Technique	"It is a formal procedure for producing some result."
Tool	"It is an instrument, language, or automated system for accomplishing something in a better way."

Table 4.6 – Quality in Use Characteristics and Product Quality Characteristics in (ISO/IEC, 2011) to be used as Outcome Codes

Outcome	Description
Quality in Use	
Context Coverage	"Degree to which a product or system can be used with effectiveness, efficiency, freedom from risk and satisfaction in both specified contexts of use and contexts beyond those initially explicitly identified."
Effectiveness	"Accuracy and completeness with which users achieve specified goals."
Efficiency	"Resources expended concerning the accuracy and completeness with which users achieve goals.
Freedom from Risk	"Degree to which a product or system mitigates the potential risk to economic status, human life, health, or the environment."

Satisfaction	"Degree to which user needs are satisfied when a product or system is used in a specified context."
Product Quality	
Compatibility	"Degree to which a product, system or component can exchange information with other products, systems or components, and perform its required functions while sharing the same hardware or software environment."
Functional Suitability	"This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions."
Maintainability	"This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it, or adapt it to changes in the environment and requirements."
Performance Efficiency	"This characteristic represents the performance relative to the number of resources used under stated conditions.
Portability	"The degree of effectiveness and efficiency of a system, product or component can be transferred from one hardware, software or other operations or usage environment to another."
Reliability	"Degree to which a system, product or component performs specified functions under specified conditions for a specified time."
Security	"Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization."
Usability	"Degree to which specified users can use a product or system to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."

Figure 4.9 – Context Information Adapted from (PETERSEN *et al.*, 2021) to be used as Context Codes

4.3.2.2 Review Codes and Themes Against the Whole Practical Questions Sample and Evolve Them

During this phase, we categorized all remaining practical questions into "state-of-the-practice" (206) / "state-of-the-art" (174) before performing the coding process. It was the most challenging task because it is difficult to identify what can be researched or not when the topics are so diverse and when they can be related to other fields of computer science (e.g., algorithms, artificial intelligence, graphic computing, among other software technologies). While there are clear examples of "state-of-the-practice" questions, such as "(206609) How do I design a card game?" and (92174) "What does 'S' stands for in OOPS?", others are not easy to identify.

We noticed that the "state-of-the-practice" questions would not help us identify topics and question structures/information that would be interesting to research. Thus, we decided to use just the "state-of-the-art" questions to support answering the research questions of this study.

When comparing the distribution of all 380 practical questions per concept (see Figure 4.10) with the distribution of practical questions marked as "state-of-the-art" (see

Figure 4.11) per concept, one can see that there is a rearrangement of importance among the concepts.
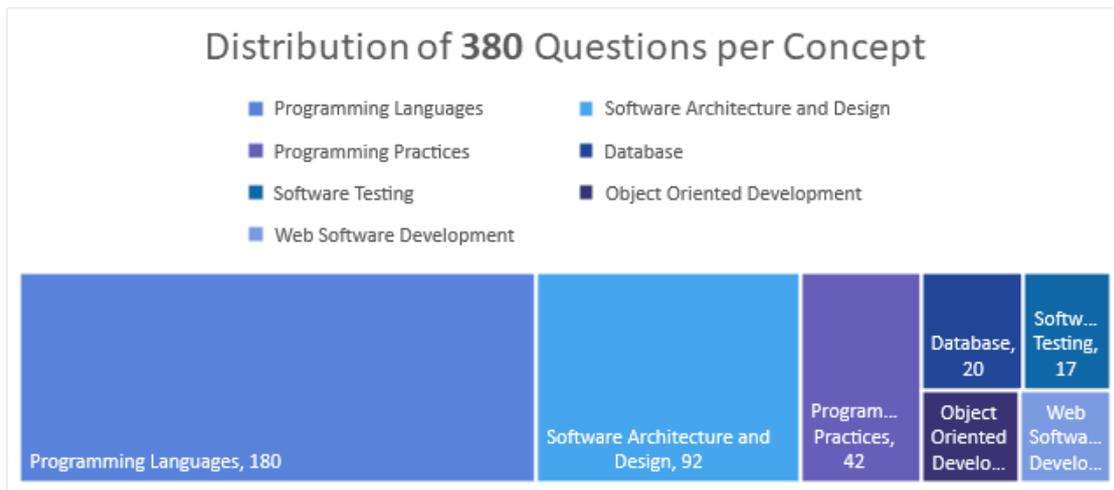
Figure 4.10 – Distribution of All Questions per Concepts
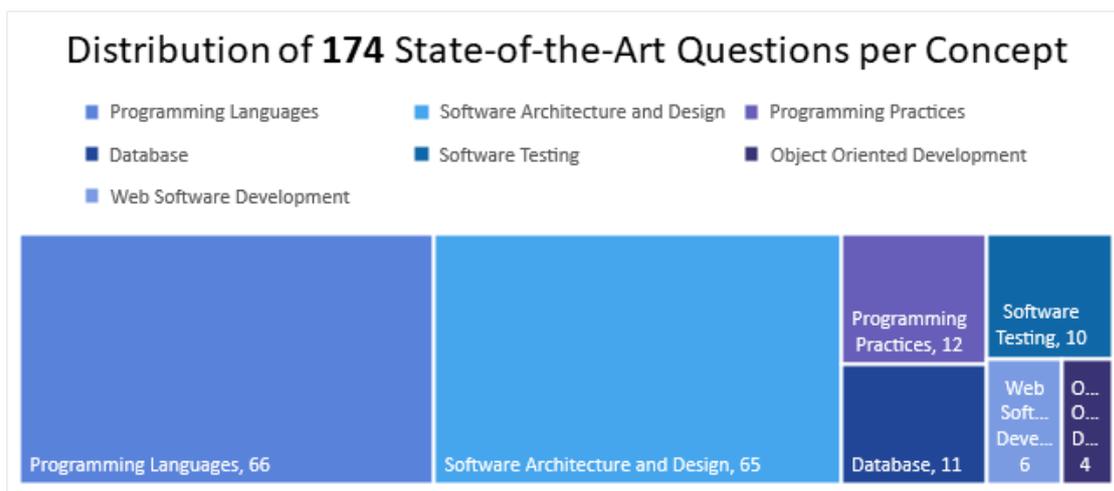
Figure 4.11 – Distribution of State-of-the-Art Questions per Concepts

On the 174 practical questions, we performed the coding process using the QDA Miner Lite tool[9] (see Figure 4.12) to track the information used by practitioners to refer to a particular code from the categories created: intervention, comparison, outcome, and context. In addition, we identified the meta-questions related to each practice question so we could have an idea of the types of questions that are usually made.

---

[9] It is a software for supporting qualitative analysis tasks, including coding.
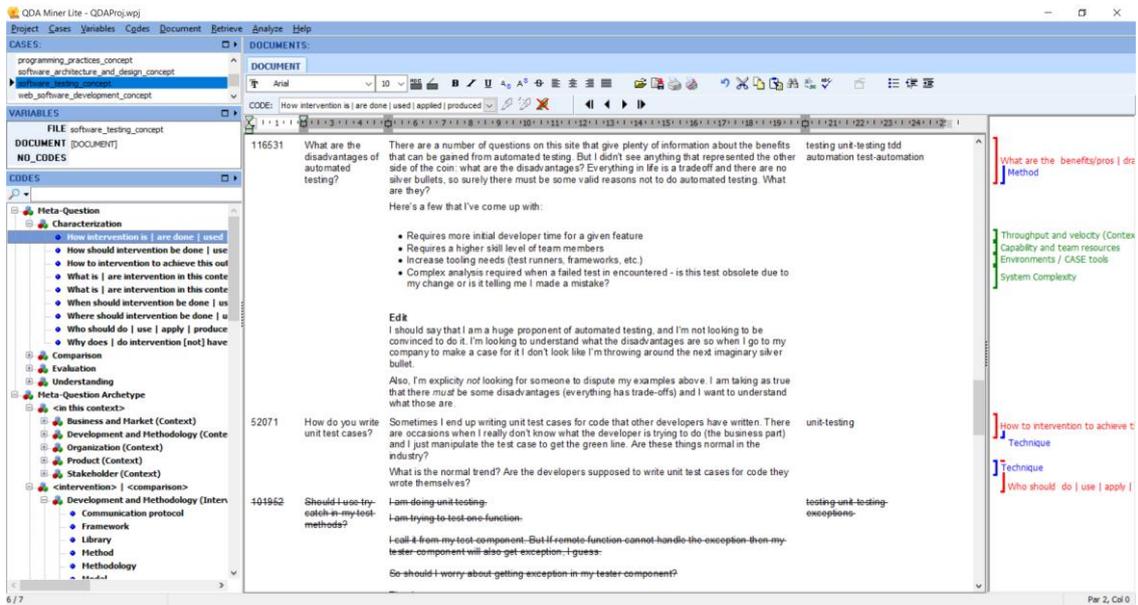
Figure 4.12 – Example of Codes and Themes in QDA Miner Lite – Questions

This process allowed us to update the initial lists of codes. As a result, 25 meta-questions were created. We identified that some questions were expecting answers that would describe an intervention. Others were expecting some comparison between the two interventions. We concluded that each meta-question could be mapped to research purposes presented in works like (BASILI, SELBY, and HUTCHENS, 1986) and (WOHLIN *et al.*, 2012). Characterization, comparison, evaluation, and understanding were the purposes we identified from the final list of meta-questions presented in Table 4.7. Table 4.8 presents the description of each purpose.

Table 4.7 – Final Meta-Questions Identified

Meta-Questions	Purpose
How <intervention> {is | are} {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?	Characterization
How should <intervention> be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?	Characterization
How to <intervention> [<to achieve this outcome>] [<in this context>]?	Characterization
What {is | are} <intervention> [<in this context>] for?	Characterization
What {is | are} <intervention> [<in this context>]?	Characterization
When should <intervention> be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?	Characterization
Where should <intervention> be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?	Characterization

Who should {do | use | apply | produce} <intervention> [<to achieve this outcome>] [<in this context>]?	Characterization
Why {does | do} <intervention> [not] have <intervention> [<to achieve this outcome>] [<in this context>]?	Characterization
What are the {benefits/pros | drawbacks/cons} of <intervention> over <comparison> [<to achieve this goal>] [<in this context>]?	Comparison
What are the {differences | similarities} among <intervention>, <comparison>, [... ,] and <comparison> [<to achieve this outcome>] [<in this context>]?	Comparison
What are the {differences | similarities} between <intervention> and <comparison>?	Comparison
When should <intervention> be {done | used | applied | produced} over <comparison> [<to achieve this outcome>] [<in this context>]?	Comparison
Where should <intervention> be {done | used | applied | produced} over <comparison> [<to achieve this outcome>] [<in this context>]?	Comparison
Which should be {done | used | applied | produced} [<to achieve this goal>] [<in this context>]: <intervention> or <comparison>?	Comparison
Why should <intervention> be {done | used | applied | produced} over <comparison> [<to achieve this outcome>] [<in this context>]?	Comparison
{Is | Are} <intervention> [not] a good thing to {do | use | apply | produce} [<to achieve this outcome>] [<in this context>]?	Evaluation
What are the {benefits/pros | drawbacks/cons} of <intervention> [<to achieve this goal>] [<in this context>]?	Evaluation
What <intervention> should [not] be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?	Evaluation
Which <intervention> should be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?	Evaluation
Why {is | are} <intervention> [not] a good thing to {do | use | apply | produce} [<to achieve this outcome>] [<in this context>]?	Evaluation
Why {is | are} <intervention> [not] {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?	Evaluation
Can <intervention> {done | used | applied | produced} [<to achieve this goal>] [<in this context>]?	Understanding
What contextual information should be considered for selecting <intervention> [<to achieve this goal>] [<in this context>]?	Understanding
Why can [not] <intervention> be {done | used | applied | produced} [<to achieve this outcome>] [<in this context>]?	Understanding
Legend: | - operator to express the need for a decision among the options {} - mandatory structure; requires decision among the options	

[] - optional structure
<intervention> - refers to a software technology/practice or a particular characteristic of it
<comparison> - refers to a software technology/practice or a particular characteristic of it
<to achieve this outcome> - refers to a particular goal that is intended to be achieved by the questioner
<in this context> - refers to a piece of contextual information particularly important to the question asked

Table 4.8 – Research Purposes / Answers Purposes

Answer Purpose	Description
Characterization	An answer of this type should provide descriptions of the basic features and the feasibility of an intervention.
Understanding	An answer of this type should provide explanations about the use of an intervention and the conditions surrounding its use.
Comparison	An answer of this type should compare the features and/or performance of the two or more interventions at stake.
Evaluation	An answer of this type should report the intervention's quality, importance, or value in a new application context.

Concerning the options for interventions/comparisons, the final list had an important increase (not necessarily all related to software technology as before): from five to 20 codes. See the complete list in Table 4.9.

Table 4.9 – Final List of Interventions/Comparisons Identified

Intervention / Comparison	Description
Development and Methodology	
Communication protocol	It is a set of formal rules describing how to transmit or exchange data, especially across a network.
Framework	It is a supporting structure that other things are built on top.
Library	It is a collection of pre-compiled and non-volatile routines used by programs.
Method	"It is a formal procedure for producing some result."
Methodology	It is a system of methods used in a particular study area or activity.
Model	It is a simplified representation of a real object.
Paradigm	It is "a particular approach or philosophy for building software."
Practice	It is the customary, habitual, or expected procedure or way of doing something.

Procedure	It is "like a recipe: a combination of tools and techniques that, in concert, produce a particular product."
Process	It is a series of actions or steps to achieve a particular end.
Standard / Pattern / Style	It is an idea or thing used as a measure, norm, or model in comparative evaluations.
Task	It is a piece of work, a small job.
Technique	"It is a formal procedure for producing some result."
Tool	"It is an instrument, language, or automated system for accomplishing something in a better way."
Product	
Algorithm	It is a finite sequence of well-defined instructions typically used to solve problems.
Computer program	It is a sequence of instructions written in a programming language that a computer can execute or interpret.
Data access	It involves authorization to access data.
Data type	It is the form in which data is stored, processed, and transmitted.
Instruction	It is detailed information telling how something should be done, operated, or assembled.
Programming language	It is a communication system used by a particular group of people or things.
System	It is a set of computer programs working together to achieve a purpose.

Concerning the outcome, all characteristics of Product Quality appeared in the questions. Most practitioners seek support in achieving functional suitability and performance efficiency. Regarding Quality in Use, the questions revealed effectiveness, efficiency, and satisfaction. We added a new theme related to Market Quality (see Table 4.10) that comprises the codes concerning competitiveness: cost/price ("Degree to which a cost and price of a product or system meet the market expectation.") and throughput and velocity ("Degree to which a product or system meet the throughput/velocity expected.").

Table 4.10 – Final List of Outcomes Identified

Outcome	Description
Quality in Use	
Effectiveness	"Accuracy and completeness with which users achieve specified goals."

Efficiency	"Resources expended concerning the accuracy and completeness with which users achieve goals.
Satisfaction	"Degree to which user needs are satisfied when a product or system is used in a specified context."
Product Quality	
Compatibility	"Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment."
Functional Suitability	"This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions."
Maintainability	"This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it, or adapt it to changes in the environment and requirements."
Performance Efficiency	"This characteristic represents the performance relative to the number of resources used under stated conditions.
Portability	"The degree of effectiveness and efficiency of a system, product or component can be transferred from one hardware, software or other operations or usage environment to another."
Reliability	"Degree to which a system, product or component performs specified functions under specified conditions for a specified time."
Security	"Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization."
Usability	"Degree to which specified users can use a product or system to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use."
Market Quality	
Competitiveness	"The degree to which a product or system meets the cost/price and throughput/velocity of delivery expected from the market."

As for the context, most of the mentioned context information was related to product and organization, whereas no context information was mentioned from business and market (see Figure 4.13). It is important to note that we did not consider the goal practitioners expected to achieve with intervention as contextual information since we had a particular list for matching them (outcomes).

The issue with context checklists is that they try to be as broad as possible, and while something can be a context in one setting, it might be an intervention/comparison or outcome in another setting. So, not to be confused in our coding process, we tried first to identify the information that was the primary goal of the question and then identify the information that was related to side effects or that set boundaries for something to happen/work.
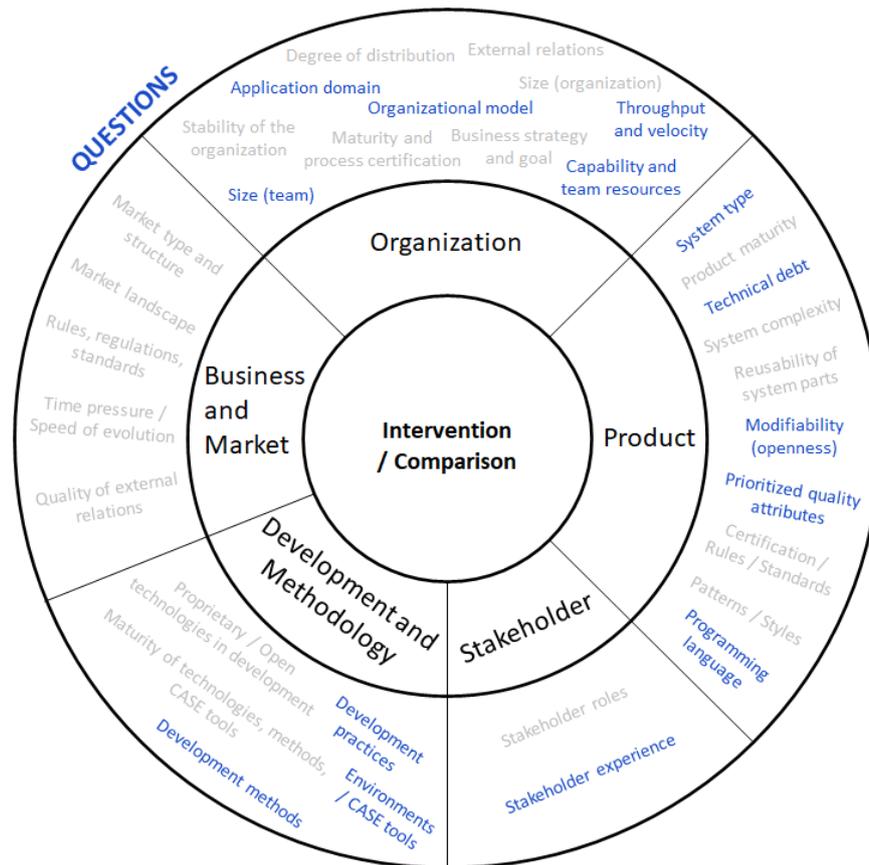
Figure 4.13 – Mentioned Contextual Information in Questions

4.3.2.3 Discussion of the Findings

The identified clusters/concepts overlap the topics mentioned by practitioners in the presented surveys about the relevance of SE knowledge. Therefore, practitioners are too into technical issues, at least regarding the popularity of these questions compared to others related to management and requirements.

Concerning the number of questions that expect state-of-the-practice answers, most regard using a software technology that could have been gathered from the technical documentation. Therefore, it might represent an indication that there is a problem related to their documentation.

We noticed that the meta-questions are related to research purposes, especially characterization, comparison, evaluation, and understanding of interventions. This fact can support researchers in identifying research goals based on the meta-questions that can be mapped to practical questions. Furthermore, because the meta-questions refer to information such as intervention, comparison, outcome, and context, once assigned to a practical question, it is easier to assemble a search string that can be used to search for answers to the question in the technical literature.

We believe that the set of lists identified from the thematic analysis can be good assets while planning research questions based on practical questions from Q&A forums.

4.4 What is Credibility to SE Practitioners?

Most works related to the relevance of scientific knowledge to practitioners are concerned with the presence of practitioners either as participants or as authors of research works. While this cannot be easily done/controlled in research studies, other mentioned concerns can. For example, many survey respondents based their disbelief in the scientific evidence on the disbelief that an observed phenomenon would apply to their industrial scenario. It is something that needs further investigation. Challenges involving fitting scientific findings to the industry context are often reported in the literature, as presented by Garousi, Petersen, and Ozkan (GAROUSI, PETERSEN, and OZKAN, 2016).

Despite widespread interest in its explicitness in SE, there is no well-defined standard for determining what information should be described as context (KITCHENHAM *et al.*, 2002). Hence, contextual information is reported differently in each report, and relevant information is often missing from them (JEDLITSCHKA, CIOLKOWSKI, and PFAHL, 2008), making it difficult to identify whether the reported software technology or evidence applies to some specific setting and how it can be interpreted in these settings by researchers and practitioners.

Jedlitschka et al. provided interesting results from a survey on the state of practice (at that time) regarding the application of reviews and inspections. The topic chosen for the survey was a proxy to support identifying the information that was important to practitioners while applying software technology in practice (JEDLITSCHKA *et al.*, 2007). For example, information regarding the impact of technologies on product quality, cost, development time, and technology cost-benefit ratio are the ones mentioned as the most important among decision-makers.

An interview with 191 practitioners concerning their perception of SE success supported the identification of the dimensions for SE success, including the impact on stakeholders, project efficiency, artifact quality, and market performance (RALPH and KELLY, 2014). In some sense, SE success can provide hints about what is important to practitioners when looking for software technologies to be employed in their projects.

All these studies raise the importance of understanding practitioners' information needs so that the communication of scientific knowledge to practitioners can be successfully reached, especially when it comes to making it understood and credible.

Several works have been proposed concerning identifying contextual information in SE empirical studies and software development projects (KIRK and MACDONELL, 2014) (CARTAXO *et al.*, 2015). The identified works can be divided into two main groups:

1. Abstract models for characterizing context by providing some guidance on how to do so, such as (BASILI, SHULL, and LANUBILE, 1999), (KARAHASANOVIC *et al.*, 2005), (DYBÅ, SJøBERG, and CRUZES, 2012); and

2. Lists of context variables involved in empirical studies, such as (GALLIS, ARISHOLM, and DYBÅ, 2003), (HÖST, WOHLIN, and THELIN, 2005), in software development projects, such as (PETERSEN and WOHLIN, 2009), (CLARKE and O'CONNOR, 2012), (JEDLITSCHKA, JURISTO, and ROMBACH, 2014), and both environments (PETERSEN *et al.*, 2021).

Overall, either the proposals for context collection and reporting are too abstract while leading researchers/practitioners to identify the important information to be collected/reported, or they are too literal and extensive, as the one proposed recently by Petersen et al. and depicted in Figure 4.14. We want to identify what practitioners use to make their SE knowledge credible.
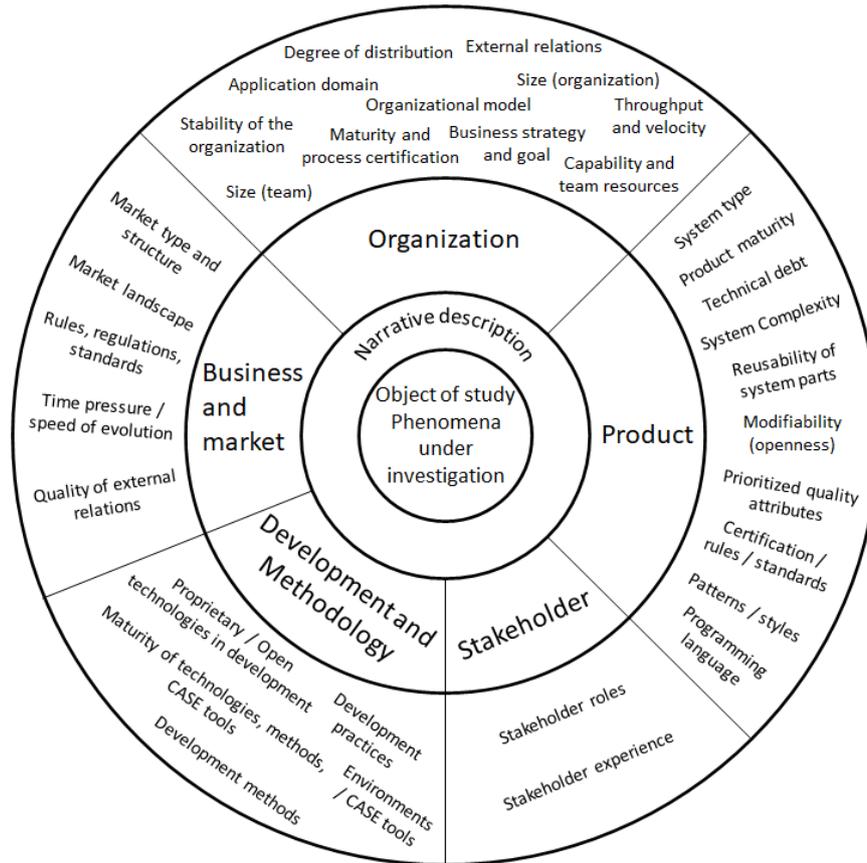
Figure 4.14 – Overview of the Context Checklist Proposed by (PETERSEN *et al.*, 2021)

4.4.1 Empirical Study Planning

4.4.1.1 Research Goal

The goal of this study is twofold. First, we want to identify the contextual information included in answers to SE practical questions that make them be considered good. Second, we want to identify information in the answers that can be used to guide researchers to report information to practitioners. Using the GQM approach (BASILI, SELBY, and HUTCHENS, 1986) (BASILI, CALDIERA, and ROMBACH, 1994), the objective of this study is as follows:

Analyze the content of practitioners' answers

For the purpose of characterization

With respect to the relevant contextual information and answer patterns

From the point of view of SE researchers

In the context of Stack Exchange answers given by practitioners while supporting answering questions provided by other practitioners dealing with problems in software development and maintenance activities.

4.4.1.2 Research Questions

RQ1: What contextual information do practitioners use while answering practical questions?

RQ2: What information can be identified in the practical answers used to support research investigations?

4.4.1.3 Dataset Overview

The dataset used for this study is from the same data dump used in the previous study about the questions from Stack Exchange. Similarly, answers come with information attached: comments, creator, and score, among others. A piece of important information for an answer is whether it was accepted as the most relevant answer provided for a question. Figure 4.15 depicts the accepted answer to the question presented in Figure 4.2.

Figure 4.15 – Example of an Accepted Answer to a Question (Figure 4.2) (Source: (STACK EXCHANGE COMMUNITY, 2008))

4.4.1.4 Data Sampling

The answers to be analyzed in this study concern the ones provided to the questions previously selected. This way, the answers for analysis are related to relevant questions. Still, it is not because an answer is provided that the community welcomes it. A study performed by Wu et al. showed that most of the answers used by users, especially those related to source code, come from accepted answers by moderation (48%) (WU *et al.*, 2019).

This way, we decided to select only accepted answers for credibility analysis, focusing on answers for questions categorized as "state-of-the-art" questions in the previous study.

4.4.1.5 Analysis Procedure

For answering RQ1 of this study, the answers are analyzed using the deductive thematic analysis approach. As for the provisional "start list" of codes to start the thematic analysis of the answers, the same list of context information used in the previous study is used in this one (see Figure 4.9).

Concerning RQ2, we believe the inductive approach is more appropriate than others. However, we decided to look for insights mainly related to argumentation schemes (WALTON, REED, and MACAGNO, 2008) (RAINER, 2017) used in the answers and their presentation concerning their styling. In addition, we are analyzing the information from the creators of the answers to identify whether they influence the acceptance of an answer.

4.4.2 Empirical Study Results: Thematic Analysis of Answers Pattern from Stack Exchange

4.4.2.1 Formulate Codes and Themes for All State-of-the-Art Answers

Like what was done with the questions, we performed the coding using QDA Miner Lite (see Figure 4.16). We read all accepted answers to the "state-of-the-art" questions previously identified. A total of 141 answers were analyzed from the 174 state-of-the-art questions. Not all questions had an accepted answer attached to them.
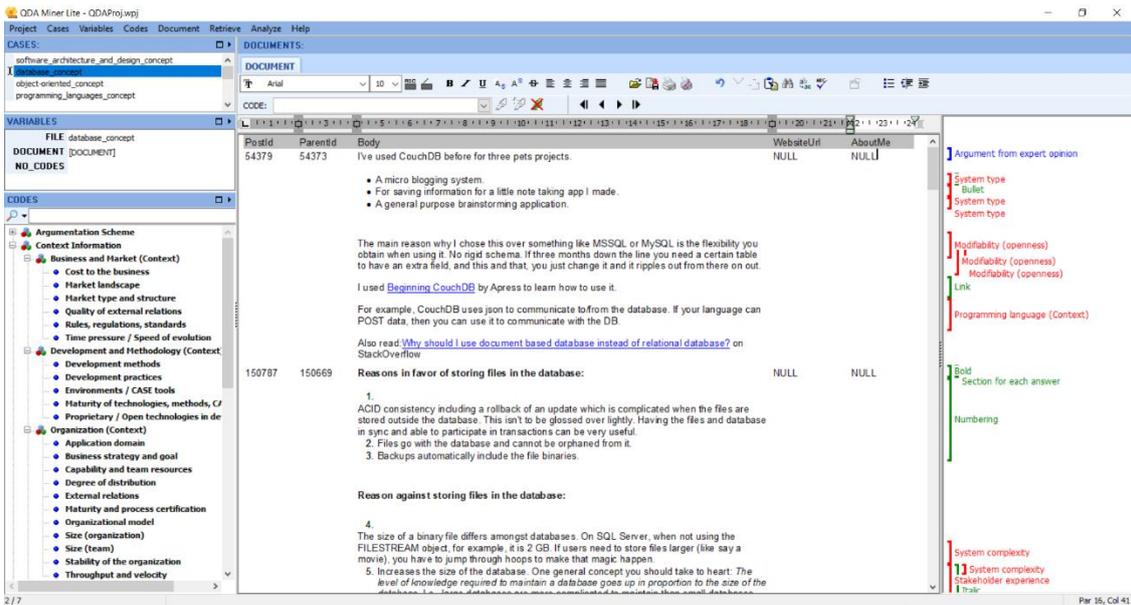
Figure 4.16 – Example of Codes and Themes in QDA Miner Lite – Answers

The set of contextual information identified in the answers (see Figure 4.17) is like the ones identified in the questions (see Figure 4.13). The product facet seems to be more important to answers than to questions.
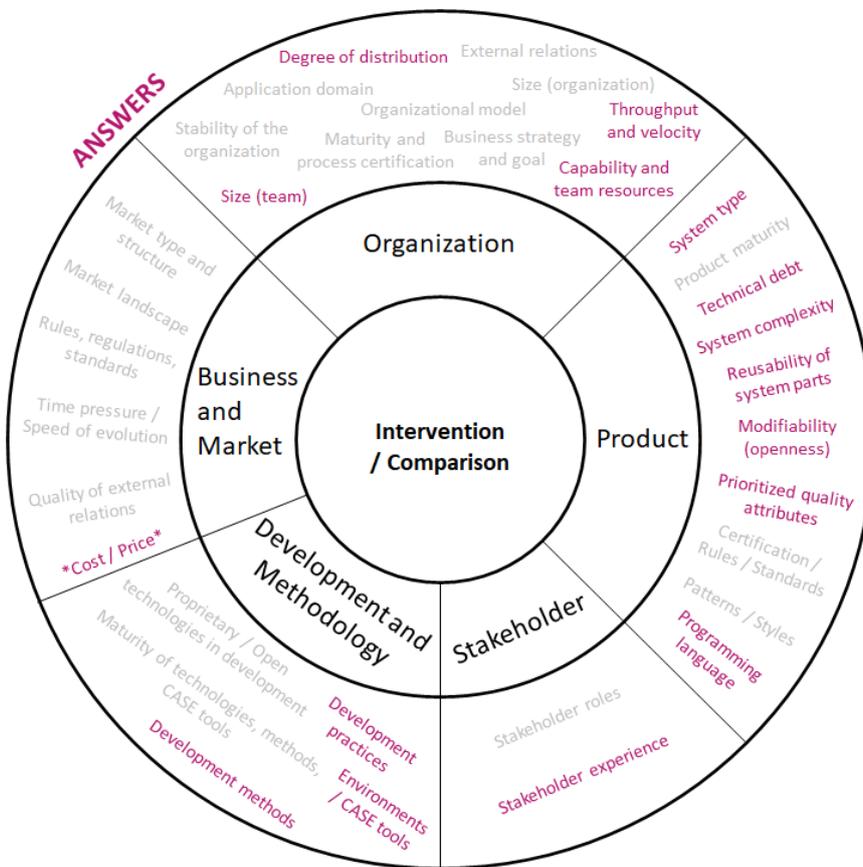
Figure 4.17 – Mentioned Contextual Information in Answers

We added the "Cost / Price" information in the Business and Market facet, as practitioners pointed out this as a side effect of many different interventions. However, the practical answers do not mention much contextual information from the original work. It might indicate that these are the least important information to practitioners or that they are more related to other topics of SE that were not discussed in the analyzed answers.

An important thing to state is that the contextual information does not appear in practical answers in a way it is easy to find or follows a standard way of reporting. Instead, the information appears in the answers among the words explaining the answer. See the example below in which we underline the contextual information we tagged as "Programming language":

> "For example, CouchDB uses JSON to communicate to/from the database. <u>If your language can POST data, then you can use it to communicate with the DB.</u>"

Notice that there is no programming language mentioned in the excerpt, but the particularities of a language set the boundaries for the use of an intervention (CouchDB).

Apart from the information, we analyzed the types of arguments used in the answers that might make them be accepted by moderation and well-scored by the community. We noticed that the arguments that usually the answers creators provide are usually based on personal experience/opinions. Thus, the argumentation schemes presented along with the answers are basically: an argument based on cases, an argument from analogy, an argument from alternative, an argument from example, an argument from expert opinion, an argument from popular opinion, an argument from popular practice, and practical reasoning.

Concerning the style of the answers, all are presented in written format, and very few provide images or drawings along with it. However, many users provide external information (links) that complement their answers. Also, the answers are usually organized in sections and/or in items/numbers, and the creators of the answers take advantage of format styling (e.g., bold, italic, underline, and font size) to emphasize important information.

The accepted answers usually try to cover all points raised by the questioners and make this explicit by splitting the answers into blocks and repeating the questions right before providing an answer for them. The presence of examples in the data is high, and the answers are usually filled with imperative phrases.

When looking at the information concerning the practitioners that provide the accepted answers, we noticed that all of them have a medium-high reputation in the

Software Engineering forum (see the comparison between statistics from the overall forum versus each concept – Table 4.11). Moreover, at least 1/3 of them provide information about themselves in their profile. All this information provides hints that they are engaged in the forum.

Table 4.11 – Average of Reputation/UpVotes/DownVotes of Answers' Creators in the Software Engineering Forum versus in each Concept

	AVG (Reputation)	AVG (UpVotes)	AVG (DownVotes)
SW Eng. Forum	97.99	6.05	1.01
Web SW Dev. Concept	6,276.00	140.00	11.20
OO Dev. Concept	58,124.00	1,359.25	887,25
SW Testing Concept	30,566.00	511.75	1,870.62
Database Concept	47,731.66	2,314.00	2,740.83
Programming Practices Concept	27,549.09	1,051.18	313.90
SW Architecture and Design Concept	23,857.85	1,389.03	2489.38
Programming Languages Concept	18,589.04	1,059.32	357.19

4.4.2.2 Discussion of the Findings

The main difference between practitioners' and researchers' answers relates to being direct with the information provided. In research presentation, there is a concern with presenting the full methodology followed to make a study repeatable and to provide full disclosure of the findings. On the other hand, practitioners do not need to present how they acquired the knowledge they are presenting. Also, practitioners are a lot more assertive with the guidance, probably because there is no worry about strong statements.

The answers are particularized but do not necessarily concern the context stated in the question. Accepted answers usually describe the use of a particular intervention in different settings, explaining different outcomes and side effects that can be produced in each setting. To base the statements, practitioners usually use personal practical examples. Certainly, the community vouches for its experience in the topic of discussion, which is usually evidenced by the cases described in the answers.

One thing is not completely clear to us: whether the practitioners that provide answers are highly ranked because they provide highly ranked answers; or the answers are highly ranked because highly ranked practitioners provide them.

We are not to produce the same answers that practitioners provide, but we can learn from them especially concerning the presentation of the results. Also, performing different context-driven research focused on slightly different settings is an interesting strategy for providing broader and complete answers. It opposes the belief that all contextual information concerning a single study should be reported to make it complete.

4.5 Related Work on Stack Exchange Questions and Answers

Most (if not all) works published related to data from Stack Exchange were based on the Stack Overflow dataset, meaning their findings are closely related to more technical knowledge. Therefore, we can divide works related to the ones presented in this chapter into three categories that can be briefly presented: i) works that provide an overview of the topics/discussions from Stack Overflow, and that identify types of questions that users usually ask; ii) works related to assessing/improving the quality of questions asked; and iii) works related to assessing/improving the quality of answers provided.

4.5.1 Discussed Topics and Types of Questions

While some works investigate what the crowd has to say in a broader sense, others focus their findings on specific topics such as refactoring tools, web development, and technical debt.

Programming languages, frameworks, and environments are among the main categories of tags identified by Treude, Barzilay, and Storey while trying to identify the most asked topics of Stack Overflow (TREUDE, BARZILAY, and STOREY, 2011). Particularly concerning the coding performed on a sample of questions by the authors, they identified the most frequently asked questions related to "how-to," discrepancy, environment, error, and decision help.

While providing an overview of the questions made in Stack Overflow, Allamanis and Sutton (ALLAMANIS and SUTTON, 2013) concluded that the questions are usually about code and not application domain, and usually, their creators ask questions seeking some support to achieve a task and not exactly to solve a problem, which matches the previous findings that identified "how-to" questions as the most popular

ones. The authors used topic modeling analysis to find the questions' concepts, topics, and types. Among the findings, the authors identified that questions are usually related to something done by the questioners but did not work, to something the questioners do not know how/why something works, to something the questioners do not know how to do or how to use, etc.

Another work that used topic modeling was performed by Barua, Thomas, and Hassan (BARUA, THOMAS, and HASSAN, 2014), which identified 40 main topics discussed in Stack Overflow. The topics are web development, data management, platform-specific discussions, security, quality assurance and collaboration, and knowledge/experience.

Differently from the works researching the topics of question on Stack Overflow are the ones concerning the reasons for using Stack Overflow (ABDALKAREEM, SHIHAB, and RILLING, 2017). Among the findings are the reasons for using knowledge (programming languages, API usage, configuration management, web frameworks, web browsers, development tools, implementation issues, database technologies, operating systems), documenting bugs, promoting Stack Overflow, feature/systems improvements, code reuse, among others.

Concerning specific topics, Pinto and Kamei (PINTO and KAMEI, 2013) performed both quantitative and qualitative research to categorize questions about refactoring tools. Users' interest in Stack Overflow is related to refactoring tools for dynamic languages, multi-languages, and even SQL. Also, the authors identified that there is interest from the community in how to perform refactorings, such as refactoring recommendations and catalogs. An interesting finding from work was that among the barriers to adopting refactoring tools is related to a lack of trust in them, in the sense that the tools might change something in the code that was not expected to be done, affecting the feature.

Bajaj, Pattabiraman, and Mesbah used mining techniques to identify common challenges and misconceptions among web developers (BAJAJ, PATTABIRAMAN, and MESBAH, 2014). The authors present findings on the concepts that seem to be related to technologies used in web development, such as JavaScript (e.g., document structure, file handling, etc.), HTML (e.g., media, elements, the offline web, etc.), and CSS (e.g., fonts). The authors also present temporal trends in the discussions about the topics, trying to identify the (lack of) popularity of certain topics over time, such as cross-browser compatibility issues that have been declining in the recent past.

More recently technical dept was the topic of attention from an investigation on Stack Overflow questions (KOZANIDIS, VERDECCHIA, and GUZMÁN, 2022). After using a combination of automated and manual processes, the authors identified that architecture, code, and design debts are the most recurring debt type presented in questions from Stack Overflow, and among the themes that are usually presented along with the types of debts are technical dept resolution, technical dept management, databases, dependencies, and coding standards.

4.5.2 Quality of Questions Asked

While assessing the quality of the questions, different works use data mining techniques to identify which characteristics of a question make them be:

- closed by moderation, such as being duplicate, off-topic, subjective, etc. (CORREA and SUREKA, 2013);

- unanswered by users, such as being unclear, vague, program-specific without a program snippet etc. (ASADUZZAMAN *et al.*, 2013);

- accepted by moderation, such as having good readability and popularity measures (title and body length, tags count, upvotes, creators' numbers of badges, etc.) (PONZANELLI *et al.*, 2014); or

- answered by users, such as having short titles, short descriptions, a few tags attached (CHUA and BANERJEE, 2015), and tags with high popularity (YAZDANINIA, LO, and SAMI, 2021).

Some works support asking questions, such as the guidelines proposed by (CALEFATO, LANUBILE, and NOVIELLI, 2018) and by (MONDAL *et al.*, 2021).

On the lines of providing support to questioners are works that automatically identify tags for questions such as (SAHA, SAHA, and SCHNEIDER, 2013), (ZHOU *et al.*, 2017), (WANG *et al.*, 2017) , and (SAINI and TRIPATHI, 2018). These works create models to suggest questioners more appropriate tags to summarize a question asked based on different information from posts and users. It is particularly important because many users from Stack Exchange follow certain tags and are informed about topics on them; as seen in other works, the use of appropriate tags for questions makes them more answerable.

4.5.3 Quality of Answers Provided

Similarly, the quality is one of the main aspects assessed for answers provided on Stack Exchange forums (particularly Stack Overflow). The work presented by

(PONZANELLI *et al.*, 2014), while assessing the quality of questions, also extends its assessment for answers. It means that readability and popularity measures are also important to measure answer quality.

While mining answers from Stack Overflow, Calefato et al. (CALEFATO *et al.*, 2015) tried to identify factors influencing certain answers' success (acceptance vote). The reputation score and badge of the answers' creators are closely related to the success of the answers. In some sense, this follows our results and the survey results discussed in section 4.2, which consider the credibility of knowledge based on the credibility of the person providing the knowledge.

The obsolescence of answers in Stack Exchange forums is also a topic of scientific investigation. It is particularly important because an outdated answer can mislead answers seekers. The authors from this last study identify that comments made by other users on answers play a major role in identifying obsolete answers. Obsolete answers are particularly problematic when related to technical issues such as programming and framework usage, as shown in (WU *et al.*, 2019) and (ZHANG *et al.*, 2021).

4.6 Discussions and Conclusions of this Chapter

There is certainly a correlation between the relevance and credibility of works to practice and the success of a technology transfer. Brings et al. discussed the approaches, success factors, and barriers to technology transfer in SE, and the results are closely related to the practitioners' perception of the relevance of scientific work (BRINGS *et al.*, 2018). Technology should be mature, sufficiently evaluated, and well documented, and practitioners should be involved during this process to foster the software technology acceptance by other practitioners.

The related works presented in the chapter showed that while SE scientific knowledge is relevant to practice, it might not be among the main concerns of practitioners. Also, in many cases, the evidence provided by research is not considered credible to practitioners. This last fact is grounded on two important things for practitioners: i) the participation of people from SE practice in the research; ii) the presentation of information useful to make decisions while applying certain knowledge in the industry.

As stated by Pfleeger, "the audience for our evidence should help us decide what kinds of studies to perform and in what context" (PFLEEGER, 1999). From the study performed on the questions from Stack Exchange, it was possible to identify SE topics

and questions that are truly relevant to practitioners. While some of the questions seek "state-of-the-practice" answers, many others are related to existing challenges researched by the SE community. Therefore, a movement from research must be done to produce scientific knowledge targeting practitioners' needs and to implement it.

From the study on the answers from Stack Exchange, we identified that it is not the presentation of all contextual information about a single observed setting that is important but a variety of observations in different settings. Certainly, some contextual variables are more important than others to practitioners, and they are usually related to what intervenes to produce a different outcome: the mediators and moderators of an observed phenomenon.

The presentation is something that we need to learn from practitioners. We need to be more straightforward with the answers and use styling to highlight important things. However, presenting a long answer or providing additional and external information is no problem if it presents various settings based on real cases.

Thus, we assume that to accomplish a successful knowledge transfer, we should focus not only on the relevance and credibility of the knowledge but also on the ways of communicating the information and the communication channels used. These last concerns are among the challenges in academia-industry collaborations (GAROUSI, PETERSEN, and OZKAN, 2016).

We believe that using practical platforms is a good alternative for presenting scientific knowledge that is of interest to the practice community.

5 Hermes[10] – A Support for Researching Real Software Engineering Issues

> "When you transfer knowledge to someone, you are not giving them
> everything but rather a foundation to build upon and to make greater." –
> Terry Mark

5.1 Introduction

Chapters 3 and 4 presented different empirical studies and related works trying to identify why practitioners do not use SE scientific in practice. We explored the four assumptions of Bennett and Jessani (BENNETT and JESSANI, 2011) for users not to use the information and concluded that the only reason that does not apply completely to the SE field is the irrelevance of the information. However, even in this case, although practitioners see the relevance of research works, they seem more concerned with other issues than the ones presented in the research.

Through two studies in the Stack Exchange community, we could identify the relevance of SE topics and issues to practitioners. Also, we could identify key information, types of arguments, and ways of presentation practitioners use in accepted answers to practical questions. Our main assumption is that to have a successful knowledge transfer and even translation to practice, researchers should include as part of their research activities a way of gathering practical issues from practice, research them and take the results back to practitioners' channel of communication. Figure 5.1 depicts an overview of the findings and conclusions produced in this thesis based on Hermes's development.

---

[10] Hermes is one of the ancient Greek gods. He is the god of trade, luck, fertility, language, travel, among others. Above all, he is well-known for being the messenger of the Olympus, being the symbolism of crossing of boundaries and being responsible for guiding the two realms of gods and humans. To achieve his responsibilities, Hermes wears winged sandals.
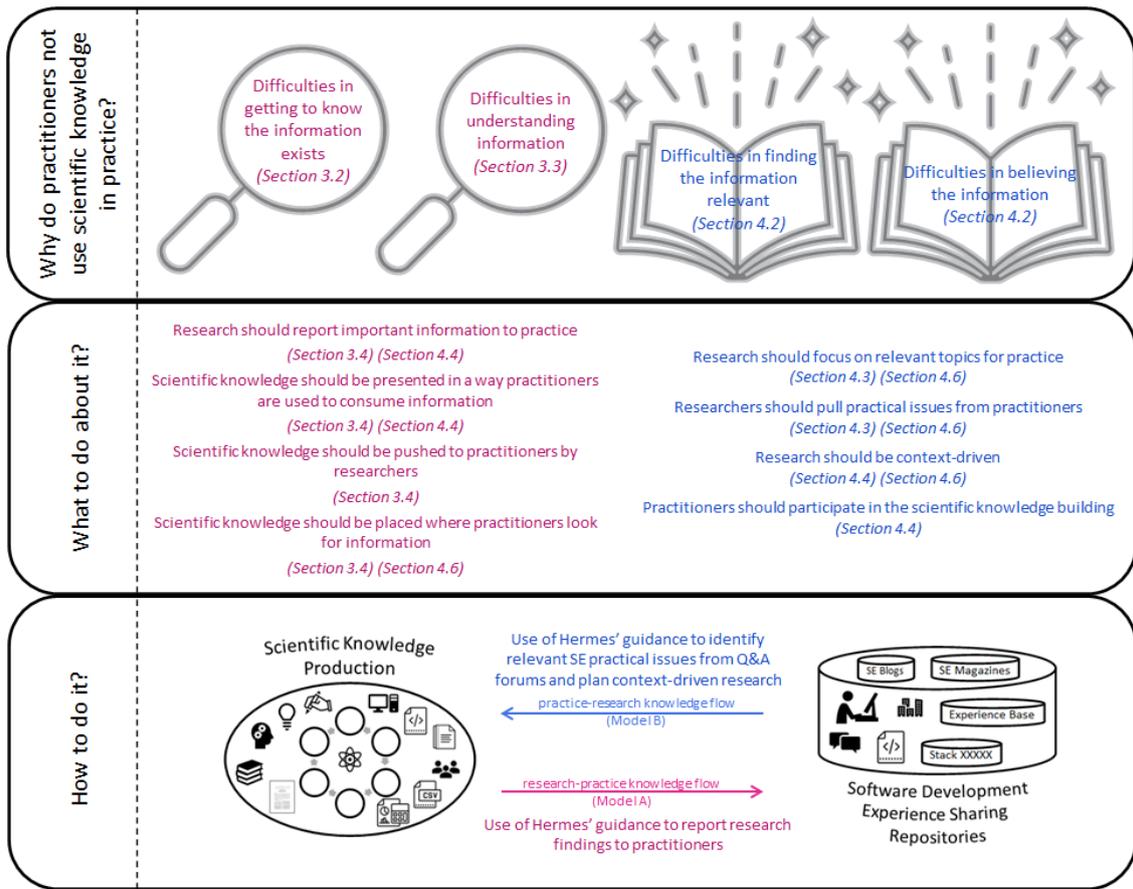
Figure 5.1 – Overview of this Thesis Research

The next sections provide more details on the guidance Hermes provides through heuristics for more cognitive tasks and computational infrastructure for repetitive and time-consuming tasks. The proposal presented in this chapter allows us to answer the main research question of this doctoral research (see Table 5.1):

Table 5.1 – Primary Research Question

> ***What to consider while planning, executing, and reporting empirical studies in software engineering to reach practitioners?***

5.2 Heuristics[11] for Researching Practical Issues in Software Engineering

Both knowledge creation and translation are cyclic, especially knowledge creation in science. Researchers are always identifying problems to be solved, producing scientific knowledge, aggregating them, and finding a way of communicating the findings. When a knowledge translation is effective, there are these parts in which the knowledge passes through one cycle to the other. Figure 5.2 provides an overview of the two cycles involved in knowledge flow.
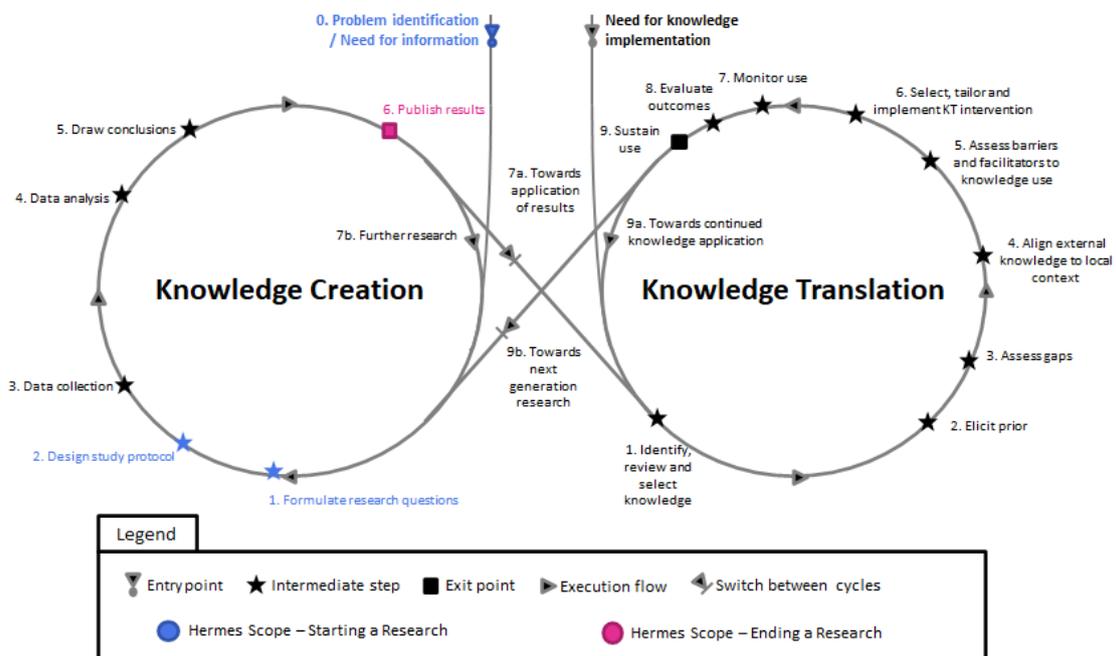
Figure 5.2 – Knowledge Creation-Translation Activities Target of Hermes – adapted from (BADAMPUDI, 2018)

During knowledge creation, the problem that needs to be addressed through scientific studies is identified, and practical research is usually aligned with practical information needs. After that, the process of planning and executing research starts until its results are ready to be published. It is possible that during this path, new research questions appear, and new studies are needed to conclude a research investigation. It can postpone the dissemination of scientific knowledge to practice since it might be identified that the knowledge is not mature enough for translation.

---

[11] We decided for the term heuristics to express a set of rules and their experience-based explanation for application.

During knowledge translation, scientific knowledge is assessed according to its applicability to a real industrial setting, and a process of tailoring it to apply it in real scenarios is performed. This cyclic process of creating and translating scientific knowledge is a symbolic representation of how scientific knowledge is produced to reach the practice. However, it might not represent all context in which knowledge translation happen.

Four main points were highlighted in the knowledge creation cycle (related to the entrance and exit steps) to mark which scientific tasks Hermes can be used for guidance. A total of 8 heuristics were proposed to guide researchers in researching practical issues and reporting scientific results using repositories of SE practical knowledge. The heuristics were created based on the studies' execution and conclusions throughout this doctoral research, as seen in Figure 5.1. The next sections detail the heuristics provided for each highlighted knowledge creation step.

5.2.1 (0.) Problem Identification / Need for Information

Heuristic 1: Focus on relevant topics for practice

Why: To make practitioners interested in the research results.

Description: Problem identification is the main part of any research. When performing practical research, the voice of practitioners needs to be heard. Different approaches can be used to collect relevant topics for practice, especially when researchers are in research-practice collaborations. Interviews, ethnographies, and surveys are examples of research strategies that can successfully collect practical relevance. We advocate that the grey literature can also be a good source for this information. Many different SE practical blogs are available to discuss various aspects of engineering software, and they hold publications and discussions that can help gather the SE hot topics in the industry. Another source of hot topics in SE is the Q&A forums in the field, where practitioners place difficulties they encounter while developing and maintaining software.

Using the Grey Literature: The SE practical community continuously updates a list of practical blogs on the field (CHOI, 2015). Today, more than 335 people contributed to its update, and almost 30K people liked the contribution. In addition to practical blogs, there are many interesting Q&A forums on SE, like Stack Overflow, but on various topics. For example, see the list in Figure 5.3 presenting other forums of Stack Exchange that can be helpful for SE topic identification. In addition, other domain-specific forums (e.g., IoT and Robotics) can be of interest when dealing with specific domains.
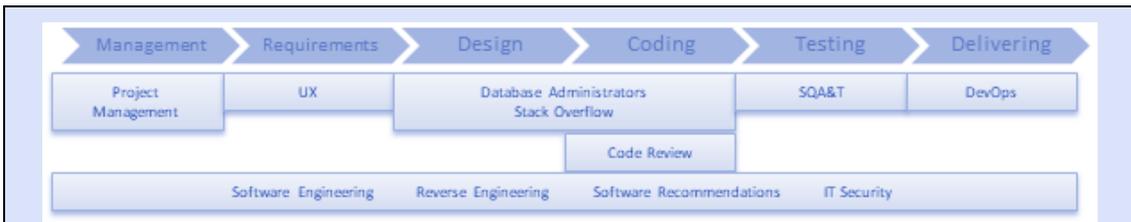
Figure 5.3 – Stack Exchange Forums Related to SE Topics

With so much data, identifying what is relevant can be overwhelming. Mining techniques can be used whenever a data repository is available (e.g., Stack Exchange data dump). Use surrogates such as numbers of posts, visualizations, comments, likes, sharing, and so on to measure what might be more relevant to practice. Quantitative data analysis can provide interesting insights about relevance, whereas qualitative data analysis can support the identification of research gaps based on practical findings. It is also possible to use qualitative analysis to support the identification of topics for research investigation in a broader sense. In any of these cases of use, a coding process might be required to synthesize the knowledge presented in different levels of abstraction and using different terminologies. While performing qualitative analysis, a sampling strategy on the amount of data to be analyzed might be needed. See below the formula for this last case—details in (KASUNIC, 2005).

$$SS = \frac{Z^2 \times p \, x \, (1-p)}{c^2}$$

SS represents the sample size, Z represents the Z-value established according to the confidence level, p represents the percentage of selecting a choice (normally, it is set at 0.5), and c represents the desired confidence interval. Considering the population is finite and has a size (pop), the following formula is the one to be used

$$SS_f = \frac{SS}{1 + \frac{SS-1}{pop}} \, .$$

Heuristic 2: Pull practical issues from practitioners

Why: To gather practitioners' problems and information needs to be researched.

Description: Along with identifying topics, it is possible to identify the practical issues from practitioners. While interviews, ethnographies, and surveys are good strategies to achieve such a goal, they all require participation from practitioners, which can limit the number of observations. The grey literature can be an alternative source for practical problems when feedback from practice to more established strategies is low.

Using the Grey Literature: The main problem in looking into the grey literature to identify issues can be the amount of data. This step might be easier if a strategy for identifying relevant topics was used previously. Furthermore, while a single question on an important topic can be sufficient to start research, data analysis on several questions on the same topic can provide a better overview of the community's issues. In this case, although mining techniques can be applied, we advise using qualitative data analysis to identify the practical issues precisely.

5.2.2 (1.) Formulate Research Questions

Heuristic 3: Focus on context-driven research

Why: To make practitioners trust that the research results can be adaptable to their industrial context.

Description: Formulating research questions in specific settings can produce more particularized research results with specific outcomes for an intervention that can depend on the context. Although this can pass the idea of a lack of consideration concerning the big picture of the research, many organizations share similar contexts, and the practitioners tend to relate more to the results once they see similar settings in the research. To achieve this, use the practical problems/issues descriptions to capture the contextual information on which the research should be based. Then, they can guide in building research questions and purposes with context embedded in them.

> **Using the Grey Literature:** To formulate context-driven research questions from the Q&A forum, try to match practical questions to at least one of the meta-questions from Table 4.7 to support the production of research questions and the identification of research purposes. For example, to fulfill the meta-questions, try to match words that comprise the options from Table 4.9 for intervention/comparison, the options from Table 4.10 for the outcome, and the highlighted information from Figure 5.4 for context.
>
> There are some tips on producing research questions from practical questions based on the meta-research questions:
>
> - Some questions come with other questions embedded in them. One must identify all those questions that have been asked along with the main one.
>
> - Usually, yes-no questions have a hidden expectation of an explanation about the provided answer, so a "why" question or even a "what"/"which" question is usually hidden in it.
>
> - Some creators answer their questions as a way of obtaining confirmatory opinions. However, even in these cases, it is important to identify the questions hidden in the answers they provide along with the questioning.
>
> - Not all practical questions can lead to research questions. Try to identify whether the question can be answered by looking at the documentation of the software technology mentioned or by providing a source code/design solution. These cases rarely lead to a research question.
>
> - Identify the keywords of the topic. Keywords are usually related to intervention, comparison, and, in some cases, outcomes of an investigation.
>
> - Contextual information is not easily identified because it is usually presented scattered in the question and mixed with intervention, comparison, and especially with expected outcomes. Therefore, do not mistake the expected outcome for contextual information. Qualitative skills are helpful in this step.
>
> - Use the information gathered to formulate a research question based on a meta-question. After that, you can create PICOC search strings that can be used to search for existing scientific knowledge

that supports answering the practical question. Alternatively, you can conduct a primary study based on the information if you cannot find existing scientific knowledge on the topic.
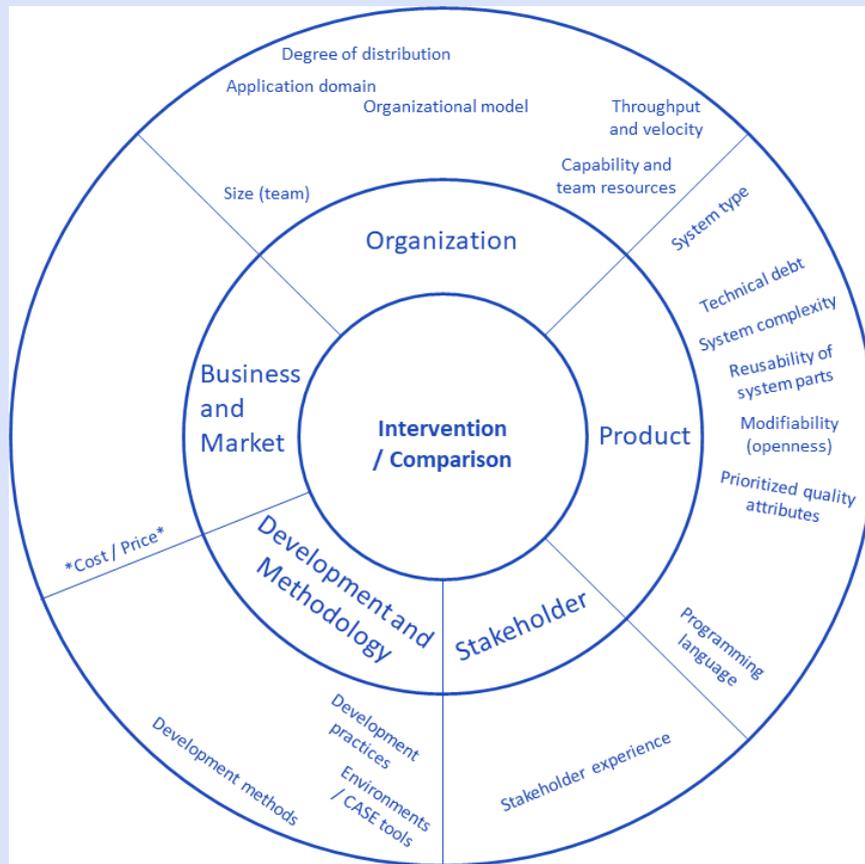
Figure 5.4 – Combined Context Information Identified in Questions and Answers

5.2.3 (2.) Design Study Protocol

Heuristic 3: Conduct context-driven research

Why: To make practitioners relate to the findings and trust that the research results can be applicable/adaptable to their industrial context.

Description: There is a misalignment of researchers' and practitioners' preferences for types of knowledge and explanation. While the former is more prone to general contextual information and propositional explanations, the latter leans towards specific information and practical explanations (RAINER, 2017). The information to be reported to practitioners to make them relate to the findings is closely related to the information planned for collection during the study's design. No matter the study strategy used, the contextual information guides retrieving information from articles in tertiary/secondary studies or even planning a primary study. The context information can be used in search

strings for articles, in extraction form to collect information from articles, and as variables to be controlled/observed in primary studies. Use the identified problems/issues/questions descriptions to plan and collect the information that matters for research and practice. Additional contextual information can also be collected, but make sure the information collected can support explaining the results by mediating our moderating the intervention-outcome relation. For example, a family of studies using the same intervention/comparison in slightly different settings might enrich discussions on the outcomes of an intervention/comparison, especially in the light of contradictory findings. It is also seen as important to practice. The contextual information can be identified using the model in Figure 5.4. Since practitioners are concerned with new knowledge's cost-benefit and side effects, this information is important in research planning.

Heuristic 4: Call practitioners to take part in the scientific knowledge building

Why: To make practitioners trust that the research results were produced with the help of people with experience in industrial settings concerning the topic.

Description: Credibility to practitioners is related to the amount of practical knowledge the messenger has. Having a practitioner as an author or participant in research can provide confidence to practitioners in the results produced. Apart from the collaboration academia-industry that naturally leads to research results with the participation of practitioners, other strategies can be used to invite practitioners to participate in the research. When considering people to participate in certain research, we usually use our network of practitioners to pass out invitations, which can be restricted. People who produce posts related to the research topic on practical blogs and Q&A forums can also be good alternatives for receiving an invitation for collaboration.

> **Using the Grey Literature:** Searching for users' contacts in blogs and Q&A forums can be a trick since the practitioners' contact information is not always available directly, and a manual search for them on the sites can be very time-consuming. After an observational study on using the Stack Exchange dataset to sample software engineers for a survey (see Chapter 6), we identified some interesting information on the dataset from the Software Engineering forum:
>
> - 43.24% of the users provide URLs for an external personal website (not necessarily contact info) → information built based on a sample of 75 users from the Software Engineering forum by looking at the data dump from Stack Exchange.
>
> - 35.13% of the same users provide contact information → information built based on a sample of 75 users from the Software Engineering forum by looking at their user profiles in the forum and

5.2.4 (6.) Publish Results

Heuristic 5: Report important information to practice

Why: To make practitioners relate to the findings and trust that the research results can be applicable/adaptable to their industrial context.

Description: If the research was based on a practical problem/issue/question, its description probably revealed some important information to guide the research. It is time to present the findings using the contextual information previously collected. Do not mistake burying the important information in too many methodological details. Instead, focus on reporting what was stated as important from the beginning. The contextual information to be reported can be identified using the model in Figure 5.4. Argumentation schemes can also be useful during this time. Practitioners usually support arguments based on cases, arguments from analogy, arguments from alternative, arguments from example, arguments from expert opinion, arguments from popular opinion, arguments from popular practice, and practical reasoning.

Using the Grey Literature: When looking at blog posts and forums, we can highlight that practitioners are usually right on target when exposing their responses. They are also usually very assertive and not afraid of strong statements. It happens because they usually ground their finding on industrial cases they participated. Following the way practitioners use blogs and forums, one can similarly report research findings, reporting the information through cases.

Heuristic 6: Present scientific knowledge in a way practitioners are used to consume information

Why: To make practitioners understand the scientific knowledge to be provided.

Description: We believe a feasible and adequate medium one can use to communicate information is the "written format," and here are some of the reasons why: i) researchers

already use writing for communication; ii) practitioners also use writing for communication; iii) there is no learning curve for using it. A good layout format that facilitates finding important info is more prone to successfully calling the target audience's attention. Make sure to split the answer into meaningful sections and items. Use bold/italic/underline to highlight important information. Mediums for knowledge transferring, such as those (GRIGOLEIT *et al.*, 2015), (CARTAXO *et al.*, 2016) and (STOREY *et al.*, 2017) can be used along with the written format as well.

> **Using the Grey Literature:** Looking at the blogs and forums guidelines for answering questions can be providential to make a research answer successful among practitioners.

Heuristic 7: Push scientific knowledge to practitioners

Why: To make practitioners aware of the scientific knowledge produced.

Description: Instead of expecting practitioners to look for your research work, be responsible for doing it. Direct your research to the practitioners involved in the practical problem/issue/question that originated the research. If the practitioners were the same ones that participated in the problem identification, you could contact them through the same communication channels used initially.

> **Using the Grey Literature:** It is possible to reuse the approach of gathering practitioners to participate in the research to identify those interested in receiving news from research results on a topic of interest.

Heuristic 8: Place scientific knowledge where practitioners are used to looking for information

Why: To guarantee a better diffusion of the scientific knowledge produced.

Description: Apart from publishing your scientific results in conferences and scientific journals, find places where the topics of your research results have been discussed (e.g., such as practical conferences, blogs, and Q&A forums) and place your results there.

> **Using the Grey Literature:** Usually, Q&A forums require consistent participation in it to allow users to post answers. Being active in practical forums is essential to make sure you can place scientific knowledge in practical repositories.

5.2.5 Heuristics Overview

Hermes' Eight Practical Research Heuristics

HERMES
SOFTWARE ENGINEERING

1. Focus on relevant topics for practice

To make your research draw practitioners' attention, make sure it deals with relevant topics for them.

Practical blogs and Q&A forums can be used to identify relevant topics for practice. Use surrogates such as numbers of posts, visualizations, comments, likes, sharing, etc., to measure what might be relevant to practice.

2. Pull practical issues from practitioners

To get to know about practitioners' problems and information needs, go gather them by yourself.

While interviews, ethnographies, and surveys are all good strategies to achieve such goal, they all require necessarily participation from practitioners. Of course, you can always use practical blogs and Q&A forums to accomplish a similar purpose when practice feedback is low.

3. Conduct context-driven research

Use the practical problem/issue/question context to guide your research and make practitioners relate to the findings.

Use the identified problem/issue/question description to plan and collect the information that matters for the research and practitioners. Additional contextual information can also be collected. However, watch out for the infinitude of the data!

4. Call practitioners to take part in the scientific knowledge building

Invite practitioners in the research field to participate in the knowledge building so that the findings can produce more credible results for practice.

Do not make it an "argument ad verecundiam," but make sure practitioners participate in your research either as an author or study participants. For example, people that produce posts related to the research topic on practical blogs and Q&A forums can be good alternatives for receiving an invitation for this collaboration.

5. Report important information to practice

Focus on reporting what was stated as important from the beginning to make practitioners relate to the findings.

If the research was based on a practical issue, probably its description revealed some important information to guide the research. It is time to present the findings using the contextual information previously collected. Do not mistake burying the important information in too many details. Focus on delivering what was stated as important from the beginning.

6. Present scientific knowledge in a way practitioners are used to consume information

It will be easier for your research results to be understood by practitioners if they are presented in a format that practitioners consume.

The written format is the most common and fasted way of communication among practitioners. However, a good layout format makes it easy to find important information and is more prone to be successful in drawing the attention of the target audience.

7. Push scientific knowledge to practitioners

Be responsible for making scientific knowledge reach its target audience.

Instead of expecting practitioners to look for your research work, be responsible for doing it yourself. Direct your research to the practitioners involved in the practical problem/issue/question that originated the research.

8. Place scientific knowledge where practitioners are used to looking for information

Place your research findings where practitioners can easily access them.

Apart from publishing your scientific results in conferences and scientific journals, find places where the topics of your research results have been discussed (e.g., such as practical conferences, blogs, and Q&A forums) and place your results there.

Figure 5.5 – Hermes' Eight Practical Research Heuristics Overview

5.3 A Computational Infrastructure to Support Gathering Practical Information to Software Engineering Research

The application of some heuristics using Q&A forums can be error-prone and time-consuming. After performing two observational studies (Chapter 6) with little computational support, we decided that a computational infrastructure would be providential to use Q&A forums as a source of practical knowledge to support research activities.

The heuristics formulated were the main source of requirements for building a computational infrastructure. In addition, the tool considers part of the data dumps provided continuously by Stack Exchange and provides specific functionalities to acquire practical knowledge that can support research.

Once we had the availability of the data and a tool to support handling it, we decided to add natural language processing to support the application of some of the heuristics presented.

5.3.1 (0.) Problem Identification / Need for Information

Concerning the heuristics involved in the problem identification (1. Focus on relevant topics for practice; 2. Pull practical issues from practitioners), we decided to make available some questions from part of the forums we identified as having the most relation with SE (see Figure 5.6 and Figure 5.7).
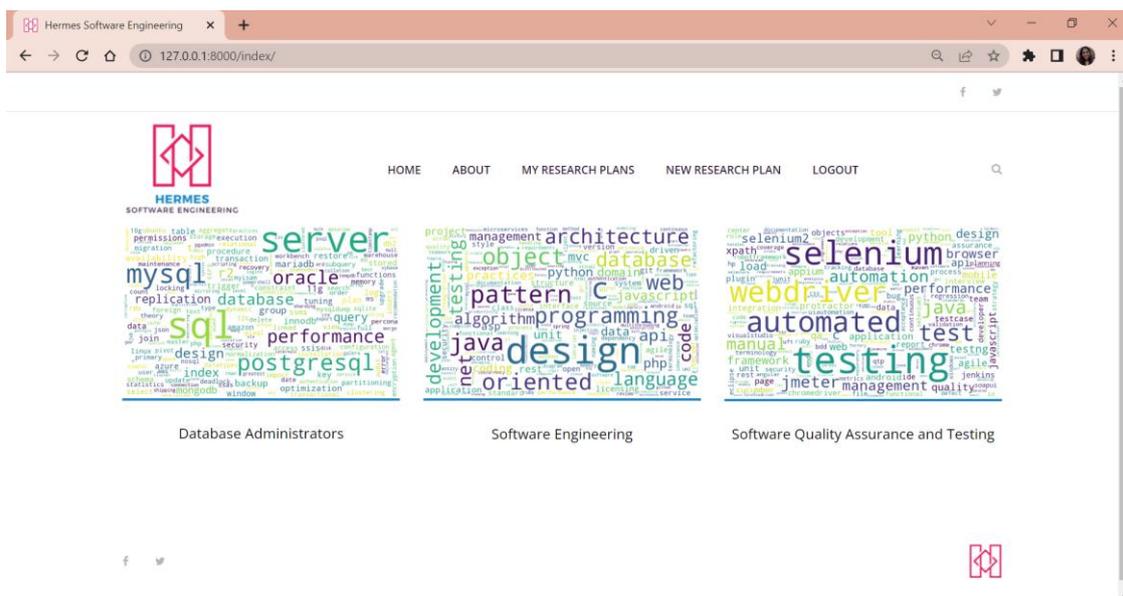
Figure 5.6 – List of Forums Available in Hermes (Source: Hermes)

By default, the tool searches questions by their tags and returns a sample (considering the calculation of sample size for 95% of confidence) ranked by the question's score, as depicted in Figure 5.7.

Figure 5.7 – List of Questions from Software Engineering Available in Hermes (Source: Hermes)

5.3.2 (1.) Formulate Research Questions

To support the formulation of research questions, we used data mining techniques that take into consideration the title, body, and tags of each practice question to identify: the most relevant terms of it, the possible answer source for it (whether "state-of-the-art" or "state-of-the-practice"), the meta-question related to it and its research purpose, as we detail in the next sections.

Data mining identifies valid, new, potentially useful, and possibly understood patterns embedded in the data (FAYYAD, PIATETSKY-SHAPIRO, and SMYTH, 1996). However, despite the data available and the feasibility of its acquisition, hardware constraints, such as memory space and processing time, limit the ability to automatically process the huge amount of information obtained, especially when this information has a vast number of features, such as data in text format.

The text provides a very rich source of knowledge. However, gathering insights from it can be very complex, even for a manual activity. Particularly for a computer to do so, the lack of natural pattern represents an additional challenge, and the vast amount of data (words) in a single text makes it hard to identify what is essential.

Problems in the data mining process usually relate to the need to handle a vast mass of data, commonly in the form of matrices with many columns representing data

characteristics. Such a huge number of features linked to another large amount of data is used to identify relevant information on the data. The bigger the data, the harder this process can be done in a timely way. It is defined as the curse of dimensionality (BELLMAN, 1957).

Techniques such as tokenization, stop-word removal, and stemming are required to reduce the dimensionality of texts before applying any other technique for discovering knowledge.
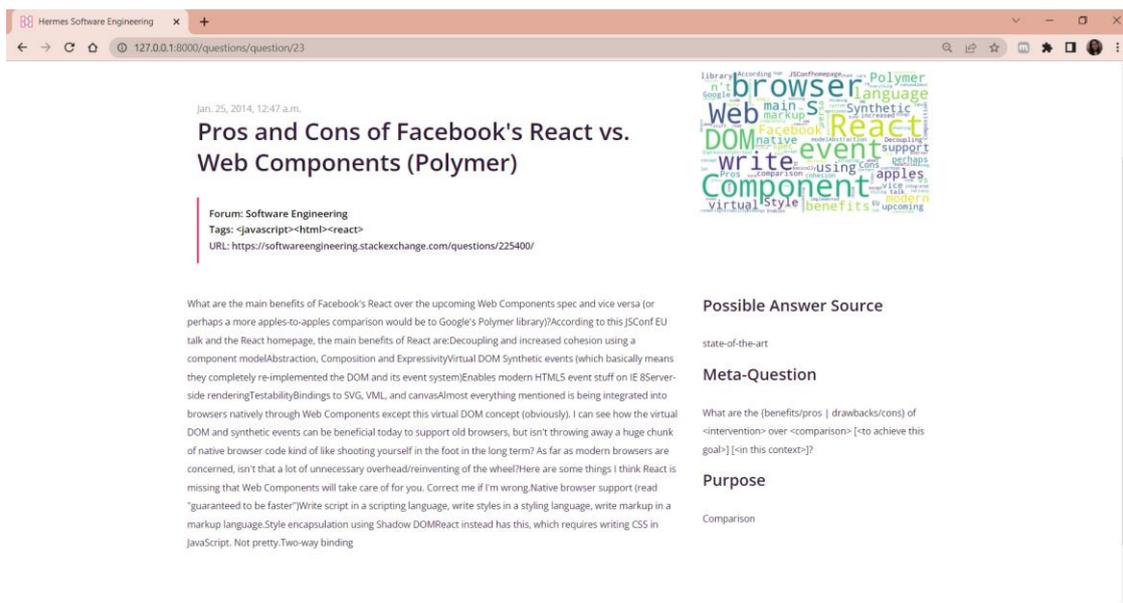
Figure 5.8 – Example of a Practical Question and the Support from Hermes to Building Research Questions (Source: Hermes)

5.3.2.1 Search Terms Identification

The search terms were identified after applying the stop-word removal from the title, body, and tags of the question, and a word cloud was built to visualize them better. It supports researchers in identifying keywords to formulate a research question and a search string to conduct different research strategies on the practical question.

5.3.2.2 State-of-the-art versus State-of-the-practice Expected Answer

A binary type of classification can be used to identify questions that are state-of-the-practice or state-of-the-art. As the name suggests, binary classification assigns a single category to a text. In the case of the expected answer source, there are two categories that a question can be: state-of-the-art and state-of-the-practice. For this classification, we trained 70% of the questions from the study presented in Chapter 4

using Logistic Regression, achieving 73% accuracy in the best scenario (considering just the question title for classification).

5.3.2.3 Research Meta-Questions and Purpose

The classification of a practical question into meta-questions, and consequently into a research purpose, is multilabel. It happens because one practical question can be related to more than one meta-question. In this case, the multi-class classification is important because a single practical question can be related to more than one research question, as seen in Chapter 4. For this classification, we trained 70% of the questions from the study presented in Chapter 4 using Logistic Regression, achieving 26% accuracy in the best scenario (considering just the question body for classification).

5.3.3 (2.) Design Study Protocol

Hermes provides a feature to create a research plan and relate it to a specific practical question for support in planning a research plan. It is possible to visualize all research plans created by a single researcher so that he can update them according to the research evolution.
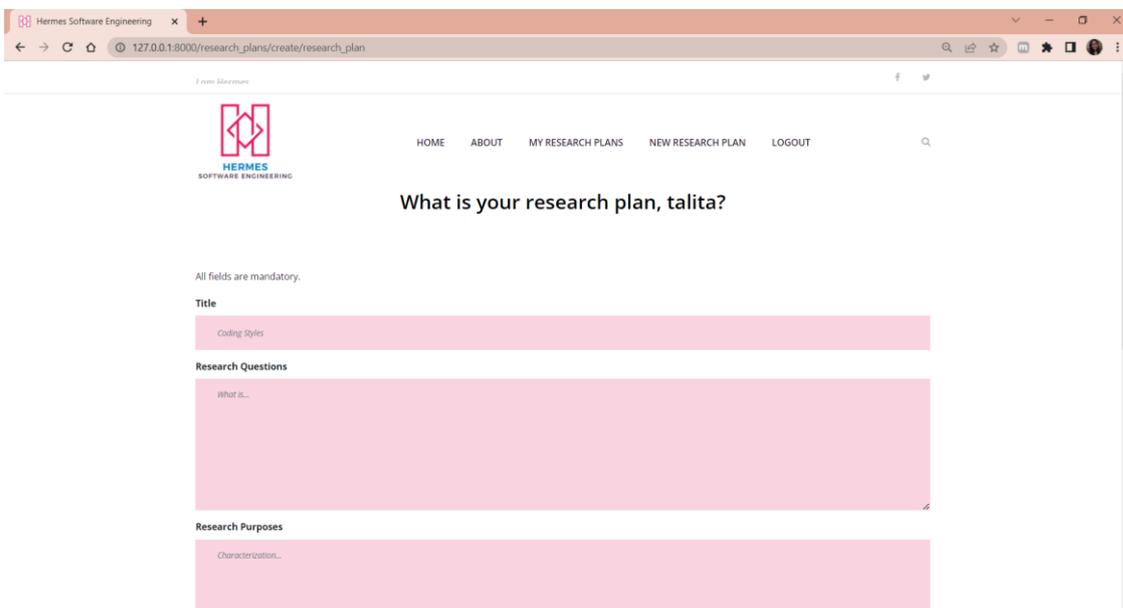
Figure 5.9 – Support from Hermes to Plan a Research Plan for a Question (Source: Hermes)

5.3.4 (6.) Publish Results

We still do not provide automated support for this last list of heuristics.

5.3.5 Assessment and Limitations

The main assessment of the computational infrastructure was its ability to correctly predict the binary classification concerning the expectations of answers (state-

of-art versus state-of-practice) and the multi-class classification concerning the meta-questions matching practical questions. Unfortunately, the results were not as good as expected for the binary classification (73% of accuracy – best results) and were underwhelming for the multi-class classification (26% of accuracy – best result).

At first, we thought that the bad results were caused by the corpus used for creating the word embedding for representing the questions as vectors of numbers. We used the whole set of posts from the Software Engineering forum as a corpus to achieve better results while representing the questions and classifying them. We achieved improvement (63% to 73% of accuracy in the case of binary classification and 21% to 26% of accuracy in the case of multi-class classification), but not enough to say we had achieved a proper classifier since a dummy classifier was just slightly worse than the one created in this work.

We believe the data used for training was not enough to build a proper classifier, specifically for the multi-class, since we did not have a good representation of questions from each meta-question. Therefore, the whole set of questions analyzed during the thematic analysis must be used to train the model to achieve better results. Perhaps even more than that.

Hermes as a computational infrastructure is still a prototype because several functionalities need to be built. Also, its artificial intelligence mechanism needs improvement to provide better guidance to researchers while using this type of data as a base for research.

5.4 Discussions and Conclusions of this Chapter

We must be dynamic if we are to keep the advance of science in software engineering. Our intention with Hermes is to provide an initial solution for bridging the gap between researchers and practitioners by supporting the planning and reporting of studies focused on more practical issues.

Hermes' eight heuristics and some tips from our experience dealing with practitioners' information repositories were provided. The computation infrastructure was seen as important to make sure some of the heuristics could be feasible, even though it has room for several improvements, especially concerning new functionality.

6 Hermes Assessment

"Assessment is today's means of modifying tomorrow's instruction." –

Carol Ann Tomlinson

6.1 Introduction

We can envision different ways of assessing this thesis concerning its deliverables. Each heuristic was based on conclusions of different studies and practitioners' opinions presented in the SE technical literature. In some sense, their importance is grounded on previous scientific findings. What we do not know is i) whether their application leads to producing scientific knowledge that is useful and used by practitioners in the industry; ii) whether the use of a Q&A forum is feasible to support applying the heuristics during research; iii) whether the computational infrastructure makes the application of the heuristics easier.

The first point is the most challenging to assess because confounding factors surround its assessment, and it can take a long time to provide a result. However, between the second and the third point, the second one can reveal the most interesting results concerning the main contribution of this work. Thus, we focus on the second point of the assessment. In this chapter, we present two observational studies conducted to assess whether Stack Exchange datasets can be used to support the application of two different heuristics and whether they are feasible to use: i) 1. Focus on relevant topics for practice, and ii) 4. Call practitioners to take part in scientific knowledge building.

In some way, the first heuristic was "applied" in the study presented in section 4.3 and by some related works presented in section 4.5. However, in both cases, a broad analysis of topic relevance was made. We wanted to know whether Q&A forums could be used to identify more specific analyses of relevance inside topics.

6.2 Can Stack Exchange Dataset be Used to Support "Focusing on Relevant Topics for Practice"?

6.2.1 Research Goal

Using the GQM approach (BASILI, SELBY, and HUTCHENS, 1986) (BASILI, CALDIERA, and ROMBACH, 1994), the objective of this study is as follows:

Analyze qualitative analysis reports on posts from Stack Exchange

For the purpose of characterization

With respect to generated SE knowledge on a specific topic and reported experience with the task

From the point of view of SE researchers

In the context of applying grounded theory to Stack Exchange posts by novice researchers.

6.2.2 Research Question

RQ1: Is it feasible to collect relevant SE knowledge about a particular topic from posts on Stack Exchange forums?

6.2.3 Participants

The study was executed in the Empirical Software Engineering course at COPPE/UFRJ (in its remote version due to the Covid-19 pandemic). The participants were graduate students (3 D.Sc. and 4 M.Sc.) in their first year of graduation (only taking disciplines at this period) in the System Engineering and Computer Science Program, and none of them had previous experience in the experimental topics taught in the course (Primary and Secondary Studies in SE). Therefore, the main assignment to grade the students in the module was the qualitative study planning and execution.

We organized two teams – one with three participants (Alpha) and the other with four participants (Beta). Members of the Experimental Software Engineering Group at COPPE/UFRJ attending the course were grouped, and part-time participants were scattered consistently among the teams (two in each group). Other characteristics, such as perceived knowledge (observed during the classes) and practical experience, were also used to organize the teams. For instance, no team was composed only of doctoral students, and all teams had participants with expertise in SE in practice (practitioners). It is important to state that the similarity of the groups was not necessary for this study since the reports would not be compared in any way but rather be used for an independent analysis.

6.2.4 Tasks

We asked the teams to select the topic they would feel more comfortable working with: database, object-oriented development, programming languages, programming practices, software architecture and design, software testing, and web software

development (the concepts identified in section 4.3.2). Both teams decided to work with topics more aligned with their research interest. Thus, team Alpha preferred working with software architecture and design, whereas team Beta decided on the database.

After 12 hours (four hours on three different days) of lectures on qualitative studies, especially using Grounded Theory (GT), the students received the assignment for applying GT on posts from Stack Exchange forums to answer specific research questions related to the topic they had chosen (see Table 6.1).

Table 6.1 – Research Questions for the GT Assignment

Research Questions for Alpha	Research Questions for Beta
RQ1: How architectural knowledge concepts (e.g., architectural component behavior, contextual constraints, software quality characteristics) are described by practitioners in the software engineering forum of stack exchange?	RQ1: Which features/aspects (e.g., structure, data manipulation) of the database are commonly used to compare different database models by practitioners in the software engineering and dba forums of stack exchange? How are they described?
RQ2: Which architectural knowledge concepts are reported together by practitioners in the stack exchange's software engineering forum?	RQ2: How are relational databases compared to non-relational databases in posts by practitioners in the software engineering and dba forums of stack exchange?
RQ3: Which software architecture phenomena can be drawn from practitioners' posts in the stack exchange's software engineering forum?	RQ3: Which database model statements can be drawn from practitioners' posts in the stack exchange software engineering and dba forums?

The assignment and the study package were presented to the students for one hour so they could have an opportunity to ask questions about the work they should perform. The GT application should be done for one month, and at the end, the students should present the answers to the research questions and deliver the reports with the results. During this one month, the students would be assisted by the teacher assistant of the module (this doctoral student).

6.2.5 Experimental Material

We separated the posts (questions and their respective answers) from Stack Exchange forums to each team according to the topic selection. We wanted them to receive a considerable number of posts. This decision was made to identify how the students would sample the data for analysis, which was advised to be done but not mandatory.

After querying the Software Engineering forum using the tag "architecture," 9560 posts returned for data analysis. However, after querying the Software Engineering forum using the tag "database," few works returned in comparison to the other topic, which made us also query the Database forum and add additional search terms related to the research question the Beta team should answer: "NoSQL" and "relational." In the end, 8735 posts returned from data analysis concerning the database topic.

In the end, each team received the following:

1. the description of the GT assignment and the description of the expected deliverables (with the specification for each section of the final expected report);

2. the research questions they should answer through the application of GT in posts from Stack Exchange;

3. GT guidelines and an example of work that applied GT in SE;

4. two research articles with concepts related to the topic they should investigate;

5. the dataset of posts in three formats (.csv, .html, and .sql).

6.2.6 Results

6.2.6.1 Team Alpha: Software Architecture and Design

The team first decided to select practical questions that could present some information to answer the research questions about architecture. Then, whether a question was selected for analysis, its answers would also be selected. Nevertheless, soon, they identified that the number of questions (3149) was not feasible to be analyzed in the time they had for the task. So, they decided to calculate a sample size with 95% confidence, leading to 343 questions (randomly picked through the years) for their selection phase.

Forty threads (questions and their respective answers) were selected for the GT application. In addition, the team could generate interesting practical SE knowledge concerning architecture out of the posts from Software Engineering.

Samples of the Findings:
"Scalability is the most mentioned characteristic related to service-oriented architecture."
"…YAGNI ("You ain't gonna need it"), KISS ("Keep it simple stupid"), and DRY("Don't repeat yourself") are generally used in a context such as architecture definition, data schema, documentation, restructure…"
"MVC and MVVM, which are paradigms based on software layer separation, are repetitively used together in practitioners' explanations, especially to use one as the explanation of the other."

6.2.6.2 Team Beta: Database

As it happened to the Alpha team, the students decided first to select the posts that could be related to the research question and then perform GT on them. However, unlike the Alpha team, no sampling was done in this regard, and all posts (8735 – both questions and answers) were analyzed concerning their relation to the research questions.

As time passed, the students had a hard time trying to fulfill a complete analysis of the posts, which made them reduce the number of posts for selection. Unfortunately, no number in this regard was provided, neither concerning the number of posts analyzed nor the number of posts selected for applying GT. An interesting comment reported by the team concerning the findings is the mismatch in terminology between practitioners and researchers, especially related to terms such as scalability, performance, and flexibility.

Samples of the Findings:
"[According to practitioners,] NoSQL databases are better to deal with a large number of queries per second, leading to an easier and cheaper data scalability than relational databases."
"[According to practitioners,] relational databases are more flexible and easier to maintain as long as there is no need for more than one server."

6.2.7 Discussions and Threats to Validity

Overall, the teams could produce some results from the forum posts, not necessarily aligned with the knowledge presented in the technical literature, as discussed in the reports. Nevertheless, even this finding can be interesting for research investigation.

All students mentioned how difficult it is to handle a huge amount of data such as the one they received, especially if a qualitative analysis is required. The Alpha team concluded the task of selecting the posts mainly because they performed a sampling beforehand.

Concerning the experience with analyzing the posts qualitatively, all students mentioned the difficulties encountered analyzing posts written by many people using different terms to describe the same thing and the same terms to describe different things. In addition, they all said the coding process should be performed in more iterations over the data because saturation could not be achieved, and they are insecure

over the taxonomies used for categorizing the posts since many presented incomplete information.

Using Stack Exchange forums is feasible for identifying practical, relevant knowledge, but a more in-depth observation requires a sampling strategy to reduce the amount of data to be analyzed. Some misinterpretations might happen when analyzing incomplete posts. It might indicate that a quality assessment of the selected posts can be of help, not to gather knowledge.

As for the threats to the validity of this assessment, we can highlight the time used to perform the tasks that were restricted to the course time. This fact hampered the proper application of GT and the reporting of the results. Additionally, the participants' lack of experience certainly negatively affected the research execution and the decisions made to reduce the number of posts to be analyzed.

6.3 Can Stack Exchange Dataset be Used to Support "Calling Practitioners to Take Part in the Scientific Knowledge Building"?

6.3.1 Research Goal

Using the GQM approach (BASILI, SELBY, and HUTCHENS, 1986) (BASILI, CALDIERA, and ROMBACH, 1994), the objective of this study is as follows:

Analyze user profiles from creators of posts

For the purpose of characterization

With respect to the presence of contact information

From the point of view of SE researchers

In the context of sampling software engineers for a survey on the Testing of Context-Aware Software System

6.3.2 Research Question

RQ1: Is it feasible to use Stack Exchange forums to identify software engineers with experience in a specific research topic to contact them?

6.3.3 Participants

The study was executed in the context of another study: a survey with software testing practitioners with real-world experience of CASS testing to understand their approaches to dealing with the CASS challenges. For this reason, the study participants, apart from this doctorate student, were three Doctors (professors) in computer science with more than 15 years of experience in research. The last three participants were the ones surveying CASS testing.

6.3.4 Tasks, Experimental Materials, and Results

Figure 6.1 depicts the tasks and respective results from this study which we describe in this section.
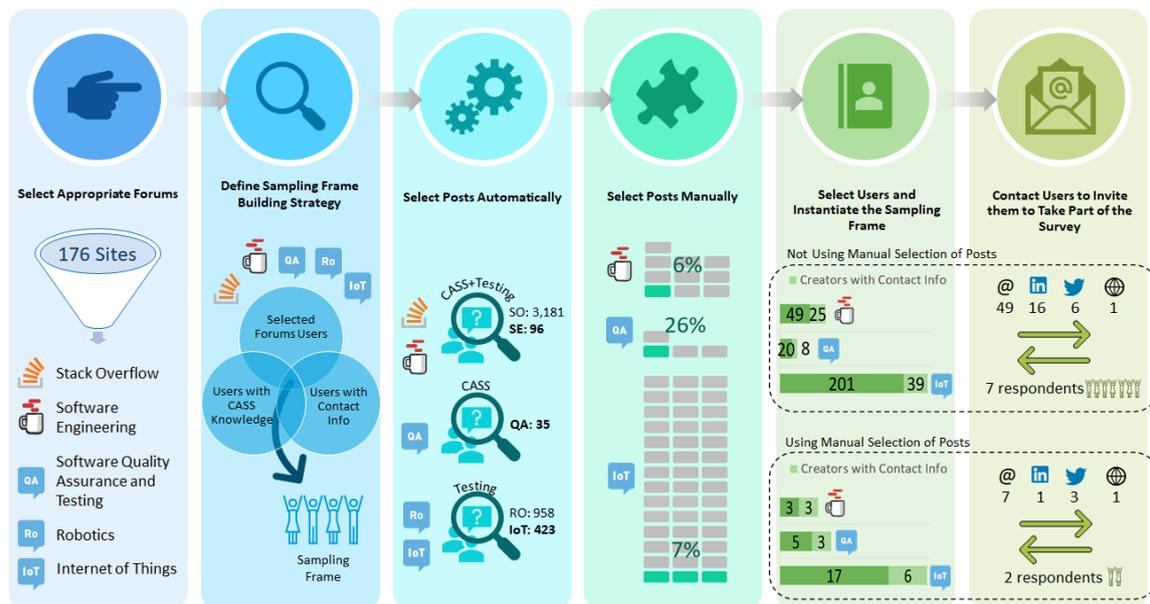
Figure 6.1 – Tasks and Results Overview

6.3.4.1 Select Appropriate Forums

Goal: To identify and select the forums whose contents are appropriate for the research topic.

Challenges: Not all forums from Stack Exchange deal with our field of interest. Thus, the search for users to be part of a survey sampling frame should be narrowed down to forums related to the questionnaire topic.

Solution: Manually select the forums of Stack Exchange forums based on their title and short description. Match the information of each forum with the field of interest.

Results for CASS Testing: Out of 176 forums, we identified five forums relevant to testing CASS:

- Stack Overflow – composed of professional and enthusiast programmers.

- Software Engineering – composed of professionals, academics, and students working within the systems development life cycle.

- Software Quality Assurance and Testing – composed of software quality control experts, automation engineers, and software testers.

- Robotics – composed of professional robotic engineers, hobbyists, researchers, and students.

- Internet of Things –builders and users of networked sensors and control devices in smart homes, industry automation, or environmental sensors.

6.3.4.2 Define Sampling Frame Building Strategy

Goal: To define the strategy that can be used to build the sampling frame.

Challenges: Even when the number of forums to prospect subjects for a study is cut down to five, the number of users in each can be overwhelming. Stack Overflow alone has more than 14 million users.

Solution: In the Stack Exchange datasets, all posts are associated with the users that created them. A strategy to search for the right users related to a topic of interest must be defined. Thus, we sought to identify posts whose content was related to the topic of interest. The size of the dataset solely means relying on manual filtering is infeasible. Hence a mixed strategy was decided as the best alternative.

Results for CASS Testing: Our strategy to define our survey's sampling frame involves identifying posts' creators that are part of the selected forums community and, simultaneously, know about CASS Testing. We defined two ways to accomplish this: i) identifying posts that automatically match a certain search string that combines terms related to CASS and Testing and simultaneously; ii) identifying posts that, upon manual assessment, have actual relation to CASS Testing. It is explained in the following two steps.

6.3.4.3 Select Posts Automatically

Goal: To select posts through an automatic filtering process.

Challenges: Going through all posts inside a forum to prospect subjects for a study. Stack Overflow hosts more than 50 million posts from different users, discussing various topics unrelated to the topic of interest.

Solution: Stack Exchange datasets can be exported to a relational database and analyzed through SQL queries. We built different queries to filter posts that match the topic of interest automatically.

Results for CASS Testing: To identify the posts related specifically to CASS Testing. We tailored the search string according to the contents of the topics addressed in the forums. It led to three groups:

- CASS+Testing: For the group of forums that we believed are related to general aspects of software development (Stack Overflow and Software Engineering), we had to use a query combining keywords regarding both CASS and Testing;

- Testing: For the group of forums that we believed is related to software testing only (SQA&T), we had to use a query presenting only CASS-related keywords;

- CASS: For the group of forums that we believed are related to common CASS domains CASS (Robotics and IoT), we had to use a query only presenting Testing related keywords.

The strategy used supported our search for more adequate posts and, consequently, posts' creators. The keywords used in the queries were extracted from the professors' previous works (MATALONGA *et al.*, 2022) and the number of posts achieved with this activity can be seen in Figure 1.

6.3.4.4 Select Posts Manually

Goal: To double-check the automatic selection process and validate its results.

Challenges: After the automatic filtering, the number of posts (and potential users) is still big. Also, we could not assure that all retrieved posts from automatic filtering would match the topic of interest. At the same time, performing a complete analysis of retrieved posts from all forums was still not feasible.

Solution: Since the main goal of this work was to assess the feasibility of using Stack Exchange to prospect participants in an empirical study, we decided to take post-samples and evaluate them manually. We sampled forums in each group to validate that the contents of the selected post described issues with the topic of interest. The three professors (the ones with more experience with CASS works) manually reviewed selected posts from the sampled forums (SE, SQA&T, IoT).

Results for CASS Testing: All (544) automated retrieved posts from SE (from general-purpose forums), SQA&T (from testing forums), and IoT (from CASS domain forums) were analyzed. The final selection of posts was based on the combination of

the three authors' evaluation of the posts, leading to a total of 43 posts related to CASS Testing in the sampled forums. Figure 6.1 provides the percentages of selected posts in each forum.

6.3.4.5 Select Users and Instantiate the Sampling Frame

Goal: To create a list of contactable users (sampling frame).

Challenges: Each user has a user profile in each forum of Stack Exchange, but not all users from Stack Exchange place their contact info directly on their user profile. Those who provide some contact info are neither the same type of information nor the same place, making it difficult to go through the users' profiles to get their contact information automatically.

Solution: Many users provide links to their GitHub accounts which can be used to gather additional information such as email, Twitter, and LinkedIn accounts. Social media can be used to contact the selected users.

Results for CASS Testing: We used two activities for user selection (automatic and manual), creating two groups within the sampling frame (selected by keywords and manually selected). We identified unique users (removing duplicates). Then manually browsed the user profile for ways to contact them. Out of the 342 potential users, 102 had some contact information. Interestingly 30 of those had fake contact information (e.g., "thisisnotawebsite.com"). We ended up with 72 users presenting actual contact information.

6.3.4.6 Contact Users to Invite them to Take Part in the Survey

Goal: To contact the selected Stack Exchange users with an invitation to participate in a survey.

Challenges: Stack Exchange does not offer direct messaging functionality between its users.

Solution: The sampling frame included all contact information we could retrieve for each user (email, social media accounts).

Results for CASS Testing: We contacted all contactable users. We used either email or the social media network according to the available information for each user. We prioritized email over social media accounts when users had multiple valid contact mechanisms.

Out of the 102 users, we were unable to contact 30. This group of users does not provide email accounts and has direct messaging disabled in their social media accounts. Out of the 12 manually selected users, two (17%) accepted our invitation. Out of the 72 unique contactable keyword-selected users, seven (10%) accepted our invitation.

6.3.5 Discussions and Threats to Validity

Representativity is certainly important in survey studies, and other works, such as (DE MELLO and TRAVASSOS, 2016) proposed different ways of identifying representative samples in survey studies. By evaluating the contents of Stack Exchange posts, we could identify members of the target population without requiring lengthy characterization questionnaires. However, the amount of content analyzed was considerably high. While it is feasible to use the Stack Exchange data set as a potential source of study participants, the whole process was very time-consuming, and certainly, we expected more than seven respondents as a result.

During this process, we analyzed alternatives to make it easier. The information about users provided in the Stack Exchange data dump is an avatar, username, self-description, age, and a link to a personal blog. No email or social media is provided. We noticed a correlation of 0.38 between the users that provide URLs for an external website and those that provide contact information (analysis made on a sample of 75 users from the Software Engineering forum). Thus, a way to narrow down the number of posts for analysis is to analyze posts whose creator provided an external URL for a website. As a result, the coverage is going to be reduced. However, there can be a gain in time.

As for the threats to the validity of this assessment, due to human limitations, we had to use a strategy to reduce the sample of posts to appraise. However, it could have affected the identification of users fit for our survey research. Also, we can highlight the structure of Stack Exchange that does not necessarily allow contacting its members. This fact can negatively impact reaching practitioners to participate in some research, even in the face of identifying many users eligible for a particular research purpose.

6.4 Discussions and Conclusions of this Chapter

Using the Stack Exchange dataset (especially the Stack Overflow one) to produce knowledge about SE is not new, and some related works were reported in section 4.5 in this regard. However, most of them rely on quantitative data analysis, which makes analyzing the amount of information less important.

We performed and presented some results on applying qualitative analysis to practical posts in Chapter 4. However, the observational studies presented in this chapter provided different alternatives for data reduction/sampling when a human analysis is done on the posts. Furthermore, these alternatives were used to improve the respective heuristic explanation of using Q&A forums in research.

Additionally, after the performance of these two studies, we discovered the need for a computational infrastructure to support the application of Hermes' heuristics in Q&A forums such as Stack Exchange ones. Building and evolving a sampling strategy is one of the main features expected of such a tool.

More studies are required to continue the improvement of Hermes' heuristics. We are working on the planning of the next study on two other heuristics: i) 2. Pull practical issues from practitioners, and ii) 3. Conduct context-driven research. Also, it is envisioned the assessment of Hermes as a whole (heuristics and tool) concerning its perceived usefulness, perceived ease of use, and user acceptance (DAVIS, 1989), after assessing the feasibility of the other heuristics applied in Q&A forums.

7 Final Considerations

"The whole of science is nothing more than a refinement of everyday
thinking." – Albert Einstein

7.1 Final Remarks

The expectation that practitioners can invest effort in software development to search and use scientific knowledge from its sources is unrealistic. They are not trained in scientific methods and are not used to following the rigorous process required in research (JURISTO and MORENO, 2001). In addition, practitioners and researchers see software development problems and solutions differently (GAROUSI, PETERSEN, and OZKAN, 2016).

Most of the research methods proposed in SE – including the knowledge translation – were borrowed from other fields of science, especially medicine, that do not share the compelling circumstances that drive SE (BURGE *et al.*, 2008) (VARGAS *et al.*, 2018). We cannot expect the same rules existing in one field to match the other completely. For instance, the literature in medicine advocates that the unit for knowledge translation should not be individual studies because it can be misleading (GRIMSHAW *et al.*, 2012). While this rule is legitimate, in the software development context, sometimes the "best solution" is the only one available (SHULL, FELDMANN, and SHAW, 2006), given the continuous evolution of software development technologies and the range and complexity of application contexts. Identifying which contextual settings to put research effort into is providential, and practitioners can support identifying them.

The most used approaches that deal with the research-practice gap are related to collaborations between researchers and practitioners in which there is an exchange of efforts between these two groups – most commonly from researchers that try to translate scientific knowledge into practice. However, guaranteeing success in more general terms and on a larger scale requires an integrated effort from both sides, which is the major challenge in many scientific fields, including SE.

A better way to acquire some success in making scientific knowledge be used in practice is by gathering daily SE issues from practice and employing efforts to push scientific productions to practitioners in a way they can easily access, understand, and appraise. In this scenario, knowing what to transfer, which format to present, and where to place are key success factors.

7.2 Contributions of this Thesis

This thesis presented several insights on transferring SE scientific knowledge to practice that can be used to guide future works intending to overcome difficulties in collaborations between research and practice or even to improve the planning, execution, and reporting of empirical studies in SE. First, we presented challenges when searching for and using SE scientific knowledge. Also, we presented a characterization of the concerns practitioners have while searching and using SE practical knowledge and of the ways (channels of communication, key contextual information, argumentation schemes, and information presentation) used by practitioners to exchange knowledge among themselves.

These insights led us to formulate heuristics to support identifying researchable topics and questions from SE Q&A forums and transforming practical questions into research questions, research purposes, and search strings to guide context-driven research on practical issues that can be reported directly to practitioners.

Other contributions of this work are the results from the performed studies related to i) the comparison of SLRs with the same purposes (RIBEIRO, MASSOLLAR, and TRAVASSOS, 2018); and ii) the identification of source code attributes impacting source code reading and comprehension, their impact (positive, negative and neutral) on source code quality, their measurement procedures, and the mediators and moderators that might influence their impact on source code quality (RIBEIRO and TRAVASSOS, 2018) (RIBEIRO, SANTOS, and TRAVASSOS, Submitted).

Results from the thematic analysis performed on the Stack Exchange questions and answers are also contributions presented in this thesis. Even though many questions lead to answers found in the state of the practice, several others can be the target of scientific works. Therefore, while conducting the thematic analysis, the research questions and purposes of the analyzed practical questions were provided along with the main terms that can be used to search for scientific evidence to answer them.

Hermes as a tool is another contribution of this thesis. Although its main idea is to support the transformation of practical questions into research questions, its base propitiates the means for working with data from Stack Exchange in research.

As prospected contributions of this thesis, we envision improving scientific production research and reports that consider topics and contextual information

important to practitioners. Also, we envision a common repository for sharing and exchanging SE knowledge by researchers and practitioners.

7.3 Limitations and Future Works

All seven empirical studies conducted during this doctoral research have limitations and threats to validity reported in their publications, which are also part of the limitations of this work. The limitations regard the characteristics and number of participants of the studies and the instruments used in the studies, among others.

Specifically related to identifying the challenges of searching and using SE scientific knowledge, we can highlight not using practitioners for performing the studies. Although we believe better feedback could have been gathered if practitioners were most of the participants trying to search and use scientific evidence in the studies, recruiting them for such studies would have been more challenging.

Specifically related to the characterization made on the SE knowledge exchange among practitioners, we can highlight the use of a single source (Stack Exchange) of practice questions for the analysis, even though the choice for Stack Exchange was based on its popularity. We believe the reasoning for analyzing questions and answers from Stack Exchange can also be used in other sources.

There are limitations regarding assessing the heuristics (since we assessed just two out of eight) and the computational infrastructure provided. Also, the computational infrastructure is still a prototype and there are several functionalities that need to be implemented to it.

Future works can target the limitations presented in this work. Especially concerning the analysis of the questions and answers, it is possible to perform a more focused analysis on topics to identify patterns since each topic may have a specific set of patterns that can be used to characterize the field better.

Discovering the relevance of practical questions to research, like what was done by (LO, NAGAPPAN, and ZIMMERMANN, 2015) (CARVER *et al.*, 2016) and (FRANCH *et al.*, 2022) but in the opposite direction can also be an interesting future work to be performed. Moreover, the results from such work can support a better selection of questions from practical forums to be researched.

Another interesting future work regards analyzing the reaction and interaction of practitioners when seeing empirical evidence as a reply to practical questions in Stack Exchange. It can support additional understanding of issues when reporting scientific knowledge to practitioners.

Also, future work assessing the quality of technologies documentations can be a good practical contribution for SE since we identified several questions on Stack Exchange related to content that should be addressed in the documentation of the mentioned technologies.

Regarding Hermes, features related to automatically assembling search strings based on each practice question's key terms (and synonyms) can be envisioned. Apart from the features that Hermes provides, other research activities can be targeted, especially those related to storing, communicating, and recommending evidence to stakeholders, either researchers or practitioners. In future activities, Hermes can be provided as a service, not limiting the use of its features to the ones available in the tool.

7.4 The Road I Have Followed

At the end of 2014, I engaged myself in this project that has been, so far, the biggest project of my life, not only because of the possible results it can provide me once it is done (Doctor Title) but also because of everything it contributed to my own personal and professional growth. During this path, some things were successful, and many others did not present the results I initially intended to accomplish.

Even though I participated in an Academia-Industry collaboration during my master's, the main topic of my thesis only became the relationship between research and practice some years after I started my Doctorate. At first, I thought about the problem of context representation in empirical studies concerning the key factors involved in the study planning, execution, and reporting. It was an issue that I encountered during my master's degree that got me thinking that I needed some extra help on how to identify important information in the studies I was reading. Years later, after reading some more works on the field, I realized this topic is more than challenging. Either people can hate you for not providing sufficient guidance on the data collection or for providing too much information to collect. That made me give up on proposing something related to the context and even matching contextual information from research and practice to recommend research software technologies to practice (proposal presented on my qualifying). Still, I brought something to discuss and found a way to balance this the best I could grasp.

During this time of changing thesis topics, I lost and found myself several times. I ended up participating in research works with different friends of mine (Breno, Cecília, Jobson, Luciana, Paulo, and Rafael) that had nothing to do with my thesis, and that made me grow a collection of paper rejections. Of course, rejections are part of the path. Still, for a novice researcher, this can be demotivating. Here and then, I got an "accept"

that thrilled me but not enough to make me think I was built for the academic world. See the list of rejections I received during this path:

Papers Graveyard:

RIBEIRO, T. V.; FRANÇA, B. B. N.; DOS SANTOS, P. S. M.; DE MELLO, R. M.; APA, C.; VALLESPIR, D.; TRAVASSOS, G. H. Experiences on Using Focus Group to Evolve Knowledge in the Software Engineering Field (Rejected on JSERD, and EMSE)

RIBEIRO, T. V.; DOS SANTOS, P. S. M.; TRAVASSOS, G. H. On the Investigation of Empirical Contradictions - Aggregated Results of Local Studies on Readability and Comprehensibility of Source Code (Rejected on ESEM and JSS; Resubmitted to EMSE – Under Review)

NASCIMENTO, L. M. A.; RIBEIRO, T. V.; TRAVASSOS, G. H. Evaluating Instruments to Support Validation Sessions (Rejected on SBES, SBQS, JSS, and JSME)

RIBEIRO, T. V.; MATALONGA, S.; AMALFITANO, D.; TRAVASSOS, G. H. Using Stack Exchange Datasets to Sample Software Engineering Practitioners (Rejected on IST)

I started working as a software requirement analyst. Never a work was so easy for me to do, and not because it was an easy job, but because I found it easy to do. I was so used to the hardness of science that having small achievements everyday/week was quite an excitement. Until today, at my work, we face software engineering problems that academia knows how to solve, but for some reasons (time, budget, people…), we do not get to know or use the existing solutions. However, against all odds, we deliver new software features every two weeks. People use them. It is working. I thought I had found my place.

Recently I was invited to be a lecturer at a private university in Rio de Janeiro. What a joy! I got very emotional when I saw a student presenting what he had learned during the course I lectured. That was the time I realized that I had finally found my place! It reminds me of the joy academia can provide. The first time I visited abroad was to present a scientific paper at CibSE in Peru. Ever since, I have attended three ICSEs. In two of them, I presented papers in co-hosted conferences and, in 2018, presented a journal's first paper in ICSE. I have attended talks of huge names in software engineering, such as Barry Boehm (whom I taught "*forró*" during ESEM 2019), Ivar Jacobson, Margaret Hamilton, Victor Basili, and so many others I cannot list now. In addition, I had the pleasure of being advised by spectacular professors to whom I owe who I am now as a professional. See the list of papers that were accepted during this journey:

Papers Accepted:

DE FRANÇA, B. B. N.; RIBEIRO, T. V.; SANTOS, P. S. M.; TRAVASSOS, G. H. Using Focus Group in Software Engineering: lessons learned on characterizing software technologies in academia and industry. In: CibSE – ESELAW Track, 2015, Lima.

NASCIMENTO, L. M. A.; RIBEIRO, T. V. É Possível Balancear Qualidade e Time-to-Deliver em Ambientes de Desenvolvimento de Software para Inovação?. In: SBQS – WQPS, 2017, Rio de Janeiro.

RIBEIRO, T. V.; TRAVASSOS, G. H. Who is Right? Evaluating Empirical Contradictions in the Readability and Comprehensibility of Source Code. In: CibSE – ESELAW Track, 2017, Buenos Aires.

RIBEIRO, T. V.; MASSOLLAR, J.; TRAVASSOS, G. H. Challenges and Pitfalls on Surveying Evidence in the Software Engineering Technical Literature: An Exploratory Study with Novices. Empirical Software Engineering, v.23, p. 1 - 70, 2017.

RIBEIRO, T. V.; MASSOLLAR, J.; TRAVASSOS, G. H. Challenges and Pitfalls on Surveying Evidence in the Software Engineering Technical Literature: An Exploratory Study with Novices. In: ICSE – Journal First Track, 2018, Gothenburg.

RIBEIRO, T. V.; TRAVASSOS, G. H. Attributes Influencing the Reading and Comprehension of Source Code – Discussing Contradictory Evidence. CLEI Electronic Journal, v.21, p. 5, 2018.

Academia has its challenges, and we must face them. There are so many factors (e.g., cultural, political, etc.) involved in bridging the gap between research and practice that we are not aimed at solving this issue. It is not even a problem exclusively in our field. With this thesis, we presented an idea for trying to do something about it. Once, I was talking about my thesis research to another junior researcher, and we ended up laughing so hard about it, basically because the idea is "let's get the issues from where practitioners place them, research on them, and report the findings where they look for the answers!". The reason we laughed so hard? It is such an obvious idea. As George Orwell said: "Sometimes the first duty of intelligent man is the restatement of the obvious."

As scientists who research practical subjects, we need to make some effort to put scientific knowledge into practice, even if this means being less systematic or scientific when reporting the results. We must provide ways of balancing what is ideal and possible. That is the academia I am all for!

References

ABDALKAREEM, R.; SHIHAB, E.; RILLING, J. What do developers use the crowd for? A study using Stack Overflow. **IEEE Software**, New York, 34, n. 2, March-April 2017. 53-60. DOI: 10.1109/ms.2017.31.

AKIYAMA, F. **An example of software system debugging**. Proceedings of IFIP Congress. Ljubljana: North-Holland. 1971. p. 353-359.

ALLAMANIS, M.; SUTTON, C. **Why, when, and what:** analyzing Stack Overflow questions by topic, type, and code. Proceedings of the Working Conference on Mining Software Repositories. San Francisco: IEEE. 2013. p. 53-56. DOI: 10.1109/msr.2013.6624004.

ASADUZZAMAN, M. et al. **Answering questions about unanswered questions**. Proceedings of the Working Conference on Mining Software Repositories. San Francisco: IEEE. 2013. DOI: 10.1109/msr.2013.6624015.

ATWOOD, J.; SPOLSKY, J. **Stack Overflow**, 2008. Available at: <https://stackoverflow.com/>. Access Date: January 2022.

BADAMPUDI, D. **Decision-making support for choosing among different component origins**. Blekinge Institute of Technology. Karlskrona, p. 279. 2018. (J1650).

BADAMPUDI, D.; WOHLIN, C.; GORSCHEK, T. **Contextualizing research evidence through knowledge translation in software engineering**. Proceedings of the Evaluation and Assessment on Software Engineering. Copenhagen: ACM. 2019. p. 306–311. DOI: 10.1145/3319008.3319358.

BAJAJ, K.; PATTABIRAMAN, K.; MESBAH, A. **Mining questions asked by web developers**. Proceedings of the Working Conference on Mining Software Repositories. Hyderabad: ACM. 2014. p. 112-121. DOI: 10.1145/2597073.2597083.

BALDASSARRE, M. T.; FRANÇA, C.; SILVA, F. Q. B. D. What aspects of context should be described in case studies about software teams? Preliminary results from a mapping study. **Product-Focused Software Process Improvement. PROFES 2016. Lecture Notes in Computer Science**, Cham, November 2016. 723-730. DOI: 10.1007/978-3-319-49094-6_61.

BARON, R. M.; KENNY, D. A. The moderator-mediator variable distinction in social psychological research: conceptual, strategic, and statistical considerations.

Journal of Personality and Social Psychology, Washington, 51, n. 6, 1986. 1173-1182. DOI: 10.1037/0022-3514.51.6.1173.

BARUA, A.; THOMAS, S. W.; HASSAN, A. E. What are developers talking about? An analysis of topics and trends in Stack Overflow. **Empirical Software Engineering**, New York, 19, June 2014. 619-654. DOI: 10.1007/s10664-012-9231-y.

BASILI, V. R. **Quantitative evaluation of software engineering methodology**. University of Maryland at College Park. College Park, p. 19. 1985.

BASILI, V. R. A personal perspective on the evolution of empirical software engineering. In: MÜNCH, J.; SCHMID, K. **Perspectives on the future of software engineering**. Heidelberg: Springer Berlin, 2013. p. 255-273. DOI: 10.1007/978-3-642-37395-4_17.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. Experience factory. In: MARCINIAK, J. J. **Encyclopedia of software engineering**. New York: Wiley, 1994. p. 469-476. DOI: 10.1002/0471028959.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. Goal question metric paradigm. In: MARCINIAK, J. J. **Encyclopedia of software engineering**. New York: Wiley, 1994. p. 528-532. DOI: 10.1002/0471028959.

BASILI, V. R.; SELBY, R. W.; HUTCHENS, D. H. Experimentation in software engineering. **Transactions on Software Engineering**, New York, SE-12, n. 7, July 1986. 733-743. DOI: 10.1109/ICSE.1996.493439.

BASILI, V. R.; SHULL, F.; LANUBILE, F. Building knowledge through families of experiments. **Transactions on Software Engineering**, New York, 25, n. 4, July 1999. 456-473. DOI: 10.1109/32.799939.

BECK, K.; CUNNINGHAM, W. **Using pattern languages for object-oriented programs**. Proceedings of the Workshop on Specification and Design for Object-Oriented Programming. Florida: ACM. 1987.

BEECHAM, S. et al. Making software engineering research relevant. **Computer**, New York, 47, n. 4, April 2014. 80-83. DOI: 10.1109/MC.2014.92.

BEGEL, A.; ZIMMERMANN, T. **Analyze this! 145 questions for data scientists in software engineering**. Proceedings of the International Conference on Software Engineering. Hyderabad: ACM. 2014. p. 12-23. DOI: 10.1145/2568225.2568233.

BELLMAN, R. **Dynamic Programming**. Princeton: Princeton University Press, 1957. 342 p.

BENIAMINI, G. et al. **Meaningful identifier names:** the case of single-letter variables. Proceedings of the International Conference on Program Comprehension (ICPC). Buenos Aires: IEEE. 2017. p. 45-54. DOI: 10.1109/ICPC.2017.18.

BENNETT, G.; JESSANI, N. **The knowledge translation toolkit - bridging the know-do gap:** a resource for researchers. New Delhi: Sage, 2011. 312 p.

BIOLCHINI, J. et al. **Systematic review in software engineering**. Federal University of Rio de Janeiro. Rio de Janeiro, p. 31. 2005. (RT-ES 679/05).

BORGES, A. et al. **Support mechanisms to conduct empirical studies in software engineering:** a systematic mapping study. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. Beijing: ACM. 2015. p. 1-14 (N. 14). DOI: 10.1145/2745802.2745823.

BÖRSTLER, J.; PAECH, B. The role of method chains and comments in software readability and comprehension - An experiment. **Transactions on Software Engineering**, New York, 9, February 2016. 886-898. DOI: 10.1109/TSE.2016.2527791.

BRAUN, V.; CLARKE, V. Using thematic analysis in psychology. **Qualitative Research in Psychology**, London, 3, n. 2, July 2008. 77-101. DOI: 10.1191/1478088706qp063oa.

BRINGS, J. et al. Approaches, success factors, and barriers for technology transfer in software engineering - Results of a systematic literature review. **Journal of Software: Evolution and Process**, New York, 30, n. 11, August 2018. DOI: 10.1002/smr.1981.

BUDGEN, D. et al. Presenting software engineering results using structured abstracts: a randomised experiment. **Empirical Software Engineering**, New York, 13, n. 4, August 2008. 435-468. DOI: 10.1007/s10664-008-9075-7.

BUDGEN, D.; KITCHENHAM, B.; BRERETON, P. **The case for knowledge translation**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Baltimore: IEEE. 2013. p. 263-266. DOI: 10.1109/ESEM.2013.41.

BURGE, J. E. et al. **Rationale-based software engineering**. Heidelberg: Springer Berlin, 2008. 316 p. DOI: 10.1007/978-3-540-77583-6.

BUSE, R. P. L.; WEIMER, W. R. **A metric for software readability**. Proceedings of the International Symposium on Software Testing and Analysis. New York: ACM. 2008. p. 121-130. DOI: 10.1145/1390630.1390647.

BUSE, R. P. L.; WEIMER, W. R. Learning a metric for code readability. **Transactions on Software Engineering**, New York, 36, n. 4, July-August 2010. 546-558. DOI: 10.1109/TSE.2009.70.

BUTLER, S. et al. **Exploring the influence of identifier names on code quality:** An empirical study. Proceedings of the European Conference on Software Maintenance and Reengineering. Madrid: IEEE. 2011. p. 156-165. DOI: 10.1109/CSMR.2010.27.

CALEFATO, F. et al. **Mining successful answers in Stack Overflow**. Proceedings of the Working Conference on Mining Software Repositories. Florence: IEEE. 2015. p. 430-433. DOI: 10.1109/MSR.2015.56.

CALEFATO, F.; LANUBILE, F.; NOVIELLI, N. How to ask for technical help? Evidence-based guidelines for writing questions on Stack Overflow. **Information and Software Technology**, Amsterdam, 94, February 2018. 186-207. DOI: 10.1016/j.infsof.2017.10.009.

CARTAXO, B. et al. **Mechanisms to characterize context of empirical studies in software engineering**. Proceedings of the Ibero-American Conference on Software Engineering. Lima: -. 2015. p. 281-294.

CARTAXO, B. et al. **Evidence briefings:** towards a medium to transfer knowledge from systematic reviews to practitioners. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Ciudad Real: ACM. 2016. p. 1-10 (N. 57). DOI: 10.1145/2961111.2962603.

CARTAXO, B.; PINTO, G.; SOARES, S. **The role of rapid reviews in supporting decision-making in software engineering practice**. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. Christchurch: ACM. 2018. p. 24-34. DOI: 10.1145/3210459.3210462.

CARTAXO, B.; PINTO, G.; SOARES, S. Rapid reviews in software engineering. In: FELDERER, M.; TRAVASSOS, G. H. **Contemporary empirical methods in software engineering**. Cham: Springer, 2020. p. 357-384. DOI: 10.1007/978-3-030-32489-6.

CARVER, J. C. **Towards reporting guidelines for experimental replications:** a proposal. Proceedings of the International Workshop on Replication in Empirical Software Engineering. Cape Town: IEEE. 2010. p. 1-4.

CARVER, J. C. et al. **Identifying barriers to the systematic literature**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Baltimore: IEEE. 2013. p. 203-213. DOI: 10.1109/ESEM.2013.28.

CARVER, J. C. et al. **How practitioners perceive the relevance of ESEM research**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Ciudad Real: ACM. 2016. p. 1-10 (n. 56). DOI: 10.1145/2961111.2962597.

CHECKLAND, P. From optimizing to learning: a development of systems thinking for the 1990s. **Journal of the Operational Research Society**, London, 36, n. 9, September 1985. 757–767. DOI: 10.1057/jors.1985.141.

CHOI, K. Software Engineering Blogs. **GitHub**, 2015. Available at: <https://github.com/kilimchoi/engineering-blogs>. Access Date: January 2022.

CHOWDHURY, I.; ZULKERNINE, M. Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities. **Journal of Systems Architecture**, Amsterdam, 57, n. 3, March 2011. 294-313. DOI: 10.1016/j.sysarc.2010.06.003.

CHUA, A. Y. K.; BANERJEE, S. Answers or no answers: studying question answerability in Stack Overflow. **Journal of Information Science**, Thousand Oaks, 41, n. 5, June 2015. 720-731. DOI: 10.1177/0165551515590096.

CLARKE, P.; O'CONNOR, R. V. The situational factors that affect the software development process: towards a comprehensive reference framework. **Information and Software Technology**, Amsterdam, 54, n. 5, May 2012. 433-447. DOI: 10.1016/j.infsof.2011.12.003.

CORREA, D.; SUREKA, A. **Fit or unfit:** analysis and prediction of 'closed questions' on Stack Overflow. Proceedings of the Conference on Online Social Networks. Boston: ACM. 2013. p. 201-212. DOI: 10.1145/2512938.2512954.

CRUZES, D. S.; DYBA, T. **Recommended steps for thematic synthesis in software engineering**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Banff: IEEE. 2011. p. 1-10. DOI: 10.1109/ESEM.2011.36.

CRUZES, D. S.; DYBÅ, T. Research synthesis in software engineering: a tertiary study. **Information and Software Technology**, Amsterdam, 53, n. 5, May 2011. 440-455. DOI: 10.1016/j.infsof.2011.01.004.

DAHL, O.-J. The roots of object-oriented programming: the Simula language. **Software Pioneers**, Berlin, 2002. 78-90. DOI: 10.1007/978-3-642-59412-0_6.

DAM, H. K. et al. Automatic feature learning for predicting vulnerable software components. **Transactions on Software Engineering**, New York, 47, n. 1, January 2021. DOI: 10.1109/TSE.2018.2881961.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, Minnesota, 13, n. 3, September 1989. 319-340. DOI: 10.2307/249008.

DE MELLO, R.; TRAVASSOS, G. H. **Conceptual framework to support sampling activities in software engineering surveys**. Proceedings of the Latin American School on Software Engineering. Porto Alegre: UFRGS. 2015. p. 30-41.

DE MELLO, R.; TRAVASSOS, G. H. **Surveys in software engineering:** identifying representative samples. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Ciudad Real, Spain: ACM. 2016. DOI: 10.1145/2961111.2962632.

DEVANBU, P.; ZIMMERMANN, T.; BIRD, C. **Belief & evidence in empirical software engineering**. Proceedings of the International Conference on Software Engineering. Austin: IEEE. 2016. DOI: 10.1145/2884781.2884812.

DEYOUNG, G. E.; KAMPEN, G. R. **Program factors as predictors of program readability**. Proceedings of the International Computer Software and Applications Conference. Chicago: IEEE. 1979. p. 668-673. DOI: 10.1109/cmpsac.1979.762579.

DIEBOLD, P. et al. **Interactive posters:** an alternative to collect practitioners' experience. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. Karlskrona: ACM. 2017. p. 230-235. DOI: 10.1145/3084226.3084272.

DIESTE, O. et al. **Comparative analysis of meta-analysis methods:** when to use which? Proceedings of the Annual Conference on Evaluation & Assessment in Software Engineering. Durham: IEEE. 2011. p. 36-45. DOI: 10.1049/ic.2011.0005.

DYBÅ, T.; KITCHENHAM, B.; JøRGENSEN, M. Evidence-based software engineering for practitioners. **IEEE Software**, New York, 22, n. 1, January 2005. 58-65. DOI: 10.1109/MS.2005.6.

DYBÅ, T.; SJøBERG, D. I. K.; CRUZES, D. S. **What works for whom, where, when, and why? On the role of context in empirical software engineering**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Lund: ACM. 2012. p. 19-28. DOI: 10.1145/2372251.2372256.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI Magazine**, Burnaby, 17, n. 3, September 1996. 37. DOI: 10.1609/aimag.v17i3.1230.

FEIGENSPAN, J. et al. **Exploring software measures to assess program comprehension**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Banff: IEEE. 2011. p. 127-136. DOI: 10.1109/ESEM.2011.21.

FELDERER, M.; TRAVASSOS, G. H. **Contemporary empirical methods in software engineering**. Cham: Springer, 2020. 525 p. DOI: 10.1007/978-3-030-32489-6.

FELDERER, M.; TRAVASSOS, G. H. The evolution of empirical methods in software engineering. In: FELDERER, M.; TRAVASSOS, G. H. **Contemporary empirical methods in software engineering**. Cham: Springer, 2020. p. 1-24. DOI: 10.1007/978-3-030-32489-6.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. Irvine. 2000.

FRANÇA, B. B. N. D.; TRAVASSOS, G. H. **Reporting guidelines for simulation-based studies in software engineering**. Federal University of Rio de Janeiro. Rio de Janeiro, p. 1-20. 2014.

FRANCH, X. et al. How do practitioners perceive the relevance of requirements engineering research? **Transactions on Software Engineering**, New York, 48, n. 6, June 2022. 1947-1964. DOI: 10.1109/TSE.2020.3042747.

GALLIS, H.; ARISHOLM, E.; DYBÅ, T. **An initial framework for research on pair programming**. Proceedings of the International Symposium on Empirical Software Engineering. Rome: IEEE. 2003. p. 1-11. DOI: 10.1109/ISESE.2003.1237972.

GAMMA, E. et al. **Design patterns:** elements of reusable object-oriented software. 1st. ed. California: Addison-Wesley, 1994. 416 p.

GAROUSI, V.; BORG, M.; OIVO, M. Practical relevance of software engineering research: synthesizing the community's voice. **Empirical Software Engineering**, New York, 25, March 2020. 1687-1754. DOI: 10.1007/s10664-020-09803-0.

GAROUSI, V.; FELDERER, M.; MÄNTYLÄ, M. V. **The need for multivocal literature reviews in software engineering:** complementing systematic literature reviews with grey literature. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. Limerick: ACM. 2016. p. 1-6 (N. 26). DOI: 10.1145/2915970.2916008.

GAROUSI, V.; FELDERER, M.; MÄNTYLÄ, M. V. Guidelines for including the grey literature and conducting multivocal literature reviews in software engineering. **Information and Software Technology**, Amsterdam, 106, February 2019. 101-121. DOI: 10.1016/j.infsof.2018.09.006.

GAROUSI, V.; PETERSEN, K.; OZKAN, B. Challenges and best practices in industry-academia collaborations in software engineering: a systematic literature review. **Information and Software Technology**, Amsterdam, 79, November 2016. 106-127. DOI: 10.1016/j.infsof.2016.07.006.

GLASER, B. G.; STRAUSS, A. L. **The discovery of grounded theory:** strategies for qualitative research. Piscataway: Aldine Transaction, 1967. 271 p.

GLASS, R. L.; VESSEY, I.; RAMESH, V. Research in software engineering: an analysis of the literature. **Information and Software Technology**, Amsterdam, 44, n. 8, June 2002. 491-506. DOI: 10.1016/S0950-5849(02)00049-6.

GORSCHEK, T. et al. A model for technology transfer in practice. **IEEE Software**, New York, 23, n. 6, November 2006. 88-95. DOI: 10.1109/MS.2006.147.

GORSCHEK, T.; WNUK, K. Third generation industrial co-production in software engineering. In: FELDERER, M.; TRAVASSOS, G. H. **Contemporary empirical methods in software engineering**. Cham: Springer, 2020. p. 503-525. DOI: 10.1007/978-3-030-32489-6.

GRIGOLEIT, F. et al. **In quest for proper mediums for technology transfer in software engineering**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Beijing: IEEE. 2015. p. 1-4. DOI: 10.1109/esem.2015.7321203.

GRIMSHAW, J. M. et al. Knowledge translation of research findings. **Implementation Science**, 7, n. 50, 31 May 2012. 1-17. DOI: 10.1186/1748-5908-7-50.

HALSTEAD, M. H. **Elements of Software Science**. New York: North Holland, 1977.

HAMERI, A.-P. Technology-transfer between basic research and industry. **Technovation**, Amsterdam, 16, n. 2, February 1996. 51-57, 91-92. DOI: 10.1016/0166-4972(95)00030-5.

HASSAN, A. E. **Mining software repositories to assist developers and support managers**. Proceedings of the International Conference on Software Maintenance. Philadelphia: IEEE. 2006. DOI: 10.1109/ICSM.2006.38.

HEVNER, A.; CHATTERJEE, S. **Design research in information systems:** theory and practice. New York: Springer, 2010. DOI: 10.1007/978-1-4419-5653-8.

HÖST, M.; WOHLIN, C.; THELIN, T. **Experimental context classification:** incentives and experience of subjects. Proceedings of the International Conference on Software Engineering. St. Louis: IEEE. 2005. p. 470-478. DOI: 10.1145/1062455.1062539.

HOVE, S. E.; ANDA, B. **Experiences from conducting semi-structured interviews in empirical software engineering research**. Proceedings of the International Software Metrics Symposium. Como: IEEE. 2005. DOI: 10.1109/METRICS.2005.24.

ISO/IEC. **ISO/IEC 25010 - Software engineering - Software product quality requirements and evaluation (SQuaRE) - system and software quality models**. Geneva, p. 34. 2011.

IVANOV, V. et al. **What do software engineers care about? gaps between research and practice**. Proceedings of the Joint Meeting on Foundations of Software Engineering. Paderborn: ACM. 2017. p. 890-895. DOI: 10.1145/3106237.3117778.

IVARSSON, M.; GORSCHEK, T. A method for evaluating rigor and industrial relevance of technology evaluations. **Empirical Software Engineering**, New York, 16, n. 3, June 2011. 365-395. DOI: 10.1007/s10664-010-9146-4.

JEDLITSCHKA, A. et al. **Relevant information sources for successful technology transfer:** a survey using inspections as an example. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Madrid: IEEE. 2007. p. 31-40. DOI: DOI 10.1109/ESEM.2007.60.

JEDLITSCHKA, A.; CIOLKOWSKI, M.; PFAHL, D. Reporting experiments in software engineering. In: SHULL, F.; SINGER, J.; SJøBERG, D. I. K. **Guide to advanced empirical software engineering**. London: Springer, 2008. p. 201-228. DOI: 10.1007/978-1-84800-044-5_8.

JEDLITSCHKA, A.; JURISTO, N.; ROMBACH, D. Reporting experiments to satisfy professionals' information needs. **Empirical Software Engineering**, New York, 19, n. 6, December 2014. 1921–1955. DOI: 10.1007/s10664-013-9268-6.

JøRGENSEN, A. H. A methodology for measuring the readability and modifiability of computer programs. **BIT Numerical Mathematics**, Norwell, 20, 1980. 393-405. DOI: 10.1007/BF01933633.

JURISTO, N.; MORENO, A. M. **Basics of software engineering experimentation**. 1. ed. New York: Springer, 2001. 396 p. DOI: 10.1007/978-1-4757-3304-4.

KAHLE, B.; GILLIAT, B. **Alexa Intenet**, 1996. Available at: <https://www.alexa.com/siteinfo/stackoverflow.com>. Access Date: January 2022.

KARAHASANOVIC, A. et al. Collecting feedback during software engineering experiments. **Empirical Software Engineering**, New York, 10, n. 2, April 2005. 113-147. DOI: 10.1007/s10664-004-6189-4.

KASOJU, A.; PETERSEN, K.; MÄNTYLÄ, M. V. Analyzing an automotive testing process with evidence-based software engineering. **Information and Software Technology**, Amsterdam, 55, n. 7, July 2013. 1237-1259. DOI: 10.1016/j.infsof.2013.01.005.

KASUNIC, M. **Designing an effective survey**. Carnegie Mellon. Pittsburgh, p. 143. 2005. (CMU/SEI-2005-HB-004). ESC-HB-2005-004.

KIRK, D.; MACDONELL, S. G. **Investigating a conceptual construct for software context**. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. London: ACM. 2014. p. 1-10 (N. 27). DOI: 10.1145/2601248.2601263.

KITCHENHAM, B. A. et al. Towards an ontology of software maintenance. **Journal of Software: Evolution and Process**, New York, 11, n. 6, December 1999. 365–389. DOI: 10.1002/(SICI)1096-908X(199911/12)11:6.

KITCHENHAM, B. A. et al. Preliminary guidelines for empirical research in software engineering. **Transactions on Software Engineering**, New York, 28, n. 8, August 2002. 721-734. DOI: 10.1109/TSE.2002.1027796.

KITCHENHAM, B.; BUDGEN, D. SEGRESS: Software Engineering Guidelines for REporting Secondary Studies. **Transactions on Software Engineering**, New York, Early Access, May 2022. 1-24. DOI: 10.1109/TSE.2022.3174092.

KITCHENHAM, B.; BUGDEN, D.; BRERETON, P. **Evidence-based software engineering and systematic reviews**. Danvers: Taylor & Francis Group, 2016.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. Keele University and University of Durham. Keele/Durham, p. 65. 2007. (EBSE-2007-01).

KITCHENHAM, B.; DYBÅ, T.; JøRGENSEN, M. **Evidence-based software engineering**. Proceedings of the International Conference on Software Engineering. Edinburgh: IEEE. 2004. p. 273-281. DOI: 10.1109/ICSE.2004.1317449.

KOZANIDIS, N.; VERDECCHIA, R.; GUZMÁN, E. **Asking about technical debt:** characteristics and automatic identification of technical debt questions on Stack Overflow. Proceedings of International Symposium on Empirical Software Engineering and Measurement. Helsinki: ACM. 2022.

KRIPPENDORFF, K. **Content analysis:** an introduction to its methodology. Thousand Oaks: Sage, 2004. 413 p.

LAVIS, J. N. et al. Assessing country-level efforts to link research to action. **Bulletin of the World Health Organisation**, Bethesda, 84, n. 8, August 2006. 620–628. DOI: 10.2471/blt.06.030312.

LAVIS, N. J. et al. How can research organizations more effectively transfer research knowledge to decision makers? **The Milbank Quartely**, New York, 81, n. 2, 2003. 221-248. DOI: 10.1111/1468-0009.t01-1-00052.

LAWRIE, D. et al. **What's in a name? A study of identifiers**. Proceedings of the International Conference on Program Comprehension. Athens: IEEE. 2006. p. 3-12. DOI 10.1109/ICPC.2006.51.

LAWRIE, D. et al. Effective identifier names for comprehension and memory. **Innovations in Systems and Software Engineering**, London, 3, n. 4, December 2007. 303-318. DOI: 10.1007/s11334-007-0031-2.

LEE, T.; LEE, J. B.; IN, H. P. Effect analysis of coding convention violations on readability of post-delivered code. **Transactions on Information and Systems**, Tokyo, E98-D, n. 7, 2015. 1286-1296. DOI: 10.1587/transinf.2014edp7327.

LINÅKER, J. et al. **Guidelines for conducting surveys in software engineering**. Lund University. Lund, p. 64. 2015.

LIPSEY, M. W.; WILSON, D. B. **Practical meta-analysis - applied social research methods series**. London: Sage, v. 49, 2001.

LO, D.; NAGAPPAN, N.; ZIMMERMANN, T. **How practitioners perceive the relevance of software engineering research**. Proceedings of the Joint Meeting on Foundations of Software Engineering. Bergamo: ACM. 2015. p. 415-425. DOI: 10.1145/2786805.2786809.

LÓPEZ, L. et al. Adoption of OSS components: a goal-oriented approach. **Data & Knowledge Engineering**, Amsterdam, 99, September 2015. 17-38. DOI: 10.1016/j.datak.2015.06.007.

MATALONGA, S. et al. Alternatives for testing of context-aware software systems non-academic settings: results from a rapid review. **Information and Software Technology**, Amsterdam, 149, September 2022. DOI: 10.1016/j.infsof.2022.106937.

MCCABE, T. J. A complexity measure. **Transactions on Software Engineering**, New York, SE-2, n. 4, December 1976. 308–320. DOI: 10.1109/tse.1976.233837.

MCGRATH, J. E. Methodology matters: doing research in the behavioral and social sciences. In: BAECKER, R. M., et al. **Human-computer interaction:** toward the year 2000. 1st. ed. San Francisco: Morgan Kaufmann Publishers Inc, 1995. p. 152-169. DOI: 10.1016/b978-0-08-051574-8.50019-4.

MENDEZ, D. et al. Open science in software engineering. In: FELDERER, M.; TRAVASSOS, G. H. **Contemporary empirical methods in software engineering**. Cham: Springer, 2020. p. 477-501. DOI: 10.1007/978-3-030-32489-6.

MENZIES, T.; WILLIAMS, L.; ZIMMERMANN, T. **Perspectives on data science for software engineering**. Amsterdam: Elsevier, 2016.

MIARA, R. J. et al. Program indentation and comprehensibility. **Communications of the ACM**, New York, 26, n. 11, November 1983. 861-867. DOI: 10.1145/182.358437.

MOLLÉRI, J. S.; PETERSEN, K.; MENDES, E. **Survey guidelines in software engineering:** an annotated review. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Ciudad Real: ACM. 2016. p. 1-6 (N. 58). DOI: 10.1145/2961111.2962619.

MOLLÉRI, J. S.; PETERSEN, K.; MENDES, E. CERSE - catalog for empirical research in software engineering: a systematic mapping study. **Information and Software Technology**, Amsterdam, 105, January 2018. 117-149. DOI: 10.1016/j.infsof.2018.08.008.

MONDAL, S. et al. **Early detection and guidelines to improve unanswered questions on Stack Overflow**. Proceedings of the Innovations in Software Engineering Conference. Bhubaneswar: ACM. 2021. p. 1-11 (N. 9). DOI: 10.1145/3452383.3452392.

MOURÃO, É. et al. On the performance of hybrid search strategies for systematic literature reviews in software engineering. **Information and Software Technology**, Amsterdam, 123, July 2020. 1-12. DOI: 10.1016/j.infsof.2020.106294.

NIELEBOCK, S. et al. Commenting source code: is it worth it for small programming tasks? **Empirical Software Engineering**, New York, 24, June 2019. 1418–1457. DOI: 10.1007/s10664-018-9664-z.

NOBLIT, G. W.; HARE, R. D. **Meta-ethnography:** synthesizing qualitative studies. Thousand Oaks: Sage, v. 11, 1988. 88 p. DOI: 10.1097/00005053-199007000-00016.

OSTERWEIL, L. J. et al. Determining the impact of software engineering research on practice. **IEEE Computer Society**, New York, 41, n. 3, March 2008. 39-49. DOI: 10.1109/mc.2008.85.

PAI, M. et al. Systematic reviews and meta-analyses: an illustrated, step-by-step guide. **The National Medical Journal Of India**, New Delhi, 17, n. 2, March-April 2004. 86-95. PMID: 15141602.

PASSOS, C. et al. **Analyzing the impact of beliefs in software project practices**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Banff: IEEE. 2011. p. 444-452. DOI: 10.1109/ESEM.2011.63.

PETERSEN, K. Guidelines for case survey research in software engineering. In: FELDERER, M.; TRAVASSOS, G. H. **Contemporary empirical methods in software engineering**. Cham: Springer, 2020. DOI: 10.1007/978-3-030-32489-6.

PETERSEN, K. et al. **Systematic mapping studies in software engineering**. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. Bari: ACM. 2008. p. 68-77.

PETERSEN, K. et al. Context checklist for industrial software engineering research and practice. **Computer Standards & Interfaces**, Amsterdam, 78, October 2021. 1-14. DOI: 10.1016/j.csi.2021.103541.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: an update. **Information and Software Technology**, Amsterdam, August 2015. 1-18. DOI: 10.1016/j.infsof.2015.03.007.

PETERSEN, K.; WOHLIN, C. **Context in industrial software engineering research**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Orlando: IEEE. 2009. p. 401-404. DOI: 10.1109/esem.2009.5316010.

PFLEEGER, S. L. Albert Einstein and empirical software engineering. **Computer**, New York, 32, n. 10, October 1999. 32-38. DOI: 10.1109/2.796106.

PFLEEGER, S. L. Understanding and improving technology transfer in software engineering. **Journal of Systems and Software**, Amsterdam, 47, n. 2-3, July 1999. 111-124. DOI: 10.1016/S0164-1212(99)00031-X.

PFLEEGER, S. L.; KITCHENHAM, B. A. Principles of survey research - Part 1: turning lemons into lemonade. **ACM SIGSOFT Software Engineering Notes**, New York, 26, n. 6, November 2001. 16-18. DOI: 10.1145/505532.505535.

PFLEEGER, S. L.; MENEZES, W. Marketing technology to software practitioners. **IEEE Software**, New York, 17, n. 1, January-February 2000. 27-33. DOI: 10.1109/52.819965.

PINTO, G.; KAMEI, F. **What programmers say about refactoring tools? An empirical investigation of Stack Overflow**. Proceedings of the Workshop on Workshop on Refactoring Tools. Indianapolis: ACM. 2013. p. 33-36. DOI: 10.1145/2541348.2541357.

PONZANELLI, L. et al. **Improving low quality Stack Overflow post detection**. Proceedings of the International Conference on Software Maintenance and Evolution. Victoria: IEEE. 2014. p. 541-544. DOI: 10.1109/icsme.2014.90.

POSNETT, D.; HINDLE, A.; DEVANBU, P. **A simpler model of software readability**. Proceedings of the Working Conference on Mining Software Repositories. New York: ACM. 2011. p. 73-82. DOI: 10.1145/1985441.1985454.

RAINER, A. Using argumentation theory to analyse software practitioners' defeasible evidence, inference and belief. **Information and Software Technology**, Amsterdam, 87, July 2017. 62-80. DOI: 10.1016/j.infsof.2017.01.011.

RAINER, A.; HALL, T.; BADDOO, N. **Persuading developers to "buy into" software process improvement:** a local opinion and empirical evidence. Proceedings of the International Symposium on Empirical Software Engineering. Rome: IEEE. 2003. DOI: 10.1109/ISESE.2003.1237993.

RAINER, A.; WILLIAMS, A. Heuristics for improving the rigour and relevance of grey literature searches for software engineering research. **Information and Software Technology**, Amsterdam, 106, February 2019. 231-233. DOI: 10.1016/j.infsof.2018.10.007.

RAINER, A.; WILLIAMS, A. Using blog-like documents to investigate software practice: benefits, challenges, and research directions. **Journal of Software: Evolution and Process**, New York, 31, n. 11, November 2019. 1-22. DOI: 10.1002/smr.2197.

RALPH, P.; KELLY, P. **The dimensions of software engineering success**. Proceedings of the International Conference on Software Engineering. Hyderabad: ACM. 2014. p. 24-35. DOI: 10.1145/2568225.2568261.

RAMANATHAN, K. **An overview of technology transfer and technology transfer models**. Asian and Pacific Centre for Transfer of Technology. Paris, p. 28. 2008.

RIBEIRO, T. V.; MASSOLLAR, J.; TRAVASSOS, G. H. Challenges and pitfalls on surveying evidence in the software engineering technical literature: An exploratory study with novices. **Empirical Software Engineering**, New York, 23, June 2018. 1594–1663. Available at: <http://rdcu.be/xNku>. DOI: 10.1007/s10664-017-9556-7.

RIBEIRO, T. V.; SANTOS, P. S. M. D.; TRAVASSOS, G. H. On the investigation of empirical contradictions - aggregated results of local studies on readability and comprehensibility of source code. **Empirical Software Engineering**, New York, Submitted.

RIBEIRO, T. V.; TRAVASSOS, G. H. **On the alignment of source code quality perspectives through experimentation:** an industrial case. Proceedings of the

International Workshop on Conducting Empirical Studies in Industry (CESI/ICSE 2015). Florence: IEEE. 2015. p. 26-33. DOI: 10.1109/CESI.2015.12.

RIBEIRO, T. V.; TRAVASSOS, G. H. **Who is right? Evaluating empirical contradictions in the readability and comprehensibility of source code**. Proceedings of the Ibero-American Conference on Software Engineering. Buenos Aires: Ibero-American Conference on Software Engineering. 2017. p. 99-112.

RIBEIRO, T. V.; TRAVASSOS, G. H. Attributes influencing the reading and comprehension of source code – discussing contradictory evidence. **CLEI Electronic Journal**, 21, n. 1, April 2018. 1-33. DOI: 10.19153/cleiej.21.1.4.

ROBINSON, H.; SEGAL, J.; SHARP, H. Ethnographically-informed empirical studies of software. **Information and Software Technology**, Amsterdam, 49, n. 6, June 2007. 540-551. DOI: 10.1016/j.infsof.2007.02.007.

RUNESON, P.; ENGSTRÖM, E.; STOREY, M.-A. The design science paradigm as a frame. In: FELDERER, M.; TRAVASSOS, G. H. **Contemporary empirical methods in software engineering**. Cham: Springer, 2020. p. 127-147. DOI: 10.1007/978-3-030-32489-6.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. **Empirical Software Engineering**, New York, 14, 2009. 131-164. DOI: 10.1007/s10664-008-9102-8.

SACKETT, D. L. et al. **Evidence-based medicine:** how to practice and teach EBM. 2nd. ed. London: Churchill Livingstone, 2000. 280 p.

SAHA, A. K.; SAHA, R. K.; SCHNEIDER, K. A. **A discriminative model approach for suggesting tags automatically for stack overflow questions**. Proceedings of the Working Conference on Mining Software Repositories. San Francisco: IEEE. 2013. p. 73-76. DOI: 10.1109/msr.2013.6624009.

SAINI, T.; TRIPATHI, S. **Predicting tags for Stack Overflow questions using**. Proceedings of the International Conference on Recent Advances in Information Technology. Dhanbad: IEEE. 2018. DOI: 10.1109/rait.2018.8389059.

SALVIULO, F.; SCANNIELLO, G. **Dealing with identifiers and comments in source code comprehension and maintenance:** Results from an ethnographically-informed study with students and professionals. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. London: ACM. 2014. p. 1-10 (N. 48). DOI: 10.1145/2601248.2601251.

SANTOS, P. S. M. D. **Evidence representation and aggregation in software engineering using theoretical structures and belief functions**. Federal University of Rio de Janeiro. Rio de Janeiro, p. 281. 2015. Doctoral Thesis.

SANTOS, P. S. M. D.; TRAVASSOS, G. H. Action research can swing the balance in experimental software engineering. **Advances in Computers**, Amsterdam, 83, 2011. 205-276. DOI: 10.1016/B978-0-12-385510-7.00005-9.

SANTOS, P. S. M. D.; TRAVASSOS, G. H. Scientific knowledge engineering: a conceptual delineation and overview of the state of the art. **The Knowledge Engineering Review**, Cambridge, 31, n. 2, March 2016. 167-199. DOI: 10.1017/S0269888916000011.

SANTOS, R. E. S.; SILVA, F. Q. B. D. **Motivation to perform systematic reviews and their impact on software engineering practice**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Baltimore: IEEE. 2013. p. 292-295. DOI: 10.1109/ESEM.2013.36.

SCANDARIATO, R. et al. Predicting vulnerable software components via text mining. **Transactions on Software Engineering**, New York, 40, n. 10, July 2014. DOI: 10.1109/TSE.2014.2340398.

SCOTT, E.; MILANI, F.; PFAHL, D. Data science and empirical software engineering. In: FELDERER, M.; TRAVASSOS, G. H. **Contemporary empirical methods in software engineering**. Cham: Springer, 2020. p. 217-233. DOI: 10.1007/978-3-030-32489-6.

SHARP, H.; DE SOUZA, C.; DITTRICH, Y. **Using ethnographic methods in software engineering research**. Proceedings of the International Conference on Software Engineering. Cape Town: IEEE. 2010. p. 491-492. DOI: 10.1145/1810295.1810445.

SHAW, M. What makes good research in software engineering? **International Journal on Software Tools for Technology Transfer**, New York, 4, n. 1, October 2002. 1-7. DOI: 10.1007/s10009-002-0083-4.

SHAW, M. **Writing good software engineering research papers**. Proceedings of the International Conference on Software Engineering. Portland: IEEE. 2003. p. 726-736. DOI: 10.1109/ICSE.2003.1201262.

SHEPPERD, M. Combining evidence and meta-analysis in software engineering. In: DE LUCIA, A.; FERRUCCI, F. **Software Engineering. ISSSE 2009,**

ISSSE 2011, ISSSE 2010. Lecture Notes in Computer Science. Heidelberg: Springer Berlin, v. 7171, 2013. p. 46-70. DOI: 10.1007/978-3-642-36054-1_2.

SHULL, F.; FELDMANN, R.; SHAW, M. **Building decision support in an imperfect world**. Proceedings of the International Symposium on Empirical Software Engineering. Rio de Janeiro: ACM. 2006. p. 33–35.

SIEGMUND, J. et al. Measuring and modeling programming experience. **Empirical Software Engineering**, New York, 19, n. 5, December 2014. 1299-1334. DOI: 10.1007/s10664-013-9286-4.

SJøBERG, D. I. K.; DYBÅ, T.; JøRGESEN, M. **The future of empirical methods in software engineering research**. Proceedings of the Future of Software Engineering Symposium. Minneapolis: IEEE. 2007. p. 358-378. DOI: 10.1109/fose.2007.30.

STACK EXCHANGE COMMUNITY. Sites. **Stack Exchange**, 2008. Available at: <https://stackexchange.com/sites>. Access Date: June 2021.

STACK EXCHANGE COMMUNITY. Software Engineering. **Stack Exchange**, 2008. Available at: <https://softwareengineering.stackexchange.com/>. Access Date: June 2021.

STACK EXCHANGE COMMUNITY. Stack Exchange Data Dump. **Archive**, 2014. Available at: <https://archive.org/details/stackexchange>. Access Date: June 2021.

STOREY, M.-A. et al. **Using a visual abstract as a lens for communicating and promoting design science research in software engineering**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Toronto: IEEE. 2017. p. 1-6. DOI: 10.1109/ESEM.2017.28.

STRAUS, S. E.; TETROE, J.; GRAHAM, I. D. **Knowledge translation in health care:** moving from evidence to practice. Hoboken: Wiley, 2013. DOI: 10.1002/9781118413555.

SUDSAWAD, P. **Knowledge translation: introduction to models, strategies, and measures**. The University of Wisconsin–Madison. Austin, p. 44. 2007.

SUSMAN, G. L.; EVERED, R. D. An assessment of the scientific merits of action research. **Administrative Sciences Quartely**, London, 23, December 1978. 582-603. DOI: 10.2307/2392581.

TASHTOUSH, Y. et al. Impact of programming features on code readability. **International Journal of Software Engineering and Its Applications**, Chennai, 7, n. 6, November 2013. 441-458. DOI: 10.14257/ijseia.2013.7.6.38.

TENNY, T. Program readability: procedures versus comments. **Transactions on Software Engineering**, Piscataway, 14, n. 9, September 1988. 1271-1279. DOI: 10.1109/32.6171.

TREUDE, C.; BARZILAY, O.; STOREY, M. A. **How do programmers ask and answer questions on the web?** Proceedings of the International Conference on Software Engineering. Honolulu: IEEE. 2011. p. 804-807. DOI: 10.1145/1985793.1985907.

VARGAS, E. L. et al. **Enabling real-time feedback in software engineering**. Proceedings of the International Conference on Software Engineering: New Ideas and Emerging Results. Gothenburg: ACM. 2018. DOI: 10.1145/3183399.3183417.

WALTON, D.; REED, C.; MACAGNO, F. **Argumentation schemes**. Cambridge: Cambridge University Press, 2008. 456 p.

WANG, S. et al. EnTagRec++: enhanced tag recommendation system for software information sites. **Empirical Software Engineering**, New York, 23, n. 2, July 2017. 800–832. DOI: 10.1007/s10664-017-9533-1.

WEYUKER, E. J. **Empirical software engineering research - the good, the bad, the ugly**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Banff: IEEE. 2011. DOI: DOI 10.1109/ESEM.2011.66.

WILLIAMS, A. **Do software engineering practitioners cite research on software testing in their online articles? A preliminary survey**. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. Christchurch: ACM. 2018. p. 151-156. DOI: 10.1145/3210459.3210475.

WOHLIN, C. **Guidelines for snowballing in systematic literature studies and a replication in software engineering**. Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. London: ACM. 2014. DOI: 10.1145/2601248.2601268.

WOHLIN, C. **Writing for synthesis of evidence in empirical software engineering**. Proceedings of the International Symposium on Empirical Software Engineering and Measurement. Torino: ACM. 2014. p. 1-4 (N. 46). DOI: 10.1145/2652524.2652559.

WOHLIN, C. et al. **Experimentation in software engineering**. Heidelberg: Springer Berlin, 2012. 236 p. DOI: 10.1007/978-3-642-29044-2.

WOHLIN, C. et al. Guidelines for the search strategy to update systematic literature reviews in software engineering. **Information and Software Technology**, Amsterdam, 127, November 2020. DOI: 10.1016/j.infsof.2020.106366.

WOHLIN, C.; RAINER, A. Challenges and recommendations to publishing and using credible evidence in software engineering. **Information and Software Technology**, Amsterdam, 134, February 2021. DOI: 10.1016/j.infsof.2021.106555.

WOHLIN, C.; RUNESON, P. Guiding the selection of research methodology in industry–academia collaboration in software engineering. **Information and Software Technology**, Amsterdam, 140, December 2021. DOI: 10.1016/j.infsof.2021.106678.

WOODFIELD, S.; DUNSMORE, H.; SHEN, V. **Effect of modularization and comments on program comprehension**. Proceedings of the International Conference on Software Engineering. San Diego: ACM. 1981. p. 215-223.

WU, A. D.; ZUMBO, B. D. Understanding and using mediators and moderators. **Social Indicators Research**, New York, July 2008. 367-392. DOI: 10.1007/s11205-007-9143-1.

WU, Y. et al. How do developers utilize source code from stack overflow? **Empirical Software Engineering**, New York, 24, April 2019. 637–673. DOI: 10.1007/s10664-018-9634-5.

YAZDANINIA, M.; LO, D.; SAMI, A. **Characterization and prediction of questions without accepted answers on Stack Overflow**. Proceedings of the International Conference on Program Comprehension. Madrid: IEEE. 2021. DOI: 10.1109/ICPC52881.2021.00015.

ZHANG, H. et al. An empirical study of obsolete answers on Stack Overflow. **Transactions on Software Engineering**, New York, 47, n. 4, April 2021. 850-862. DOI: 10.1109/tse.2019.2906315.

ZHOU, P. et al. **Scalable tag recommendation for software information sites**. Proceeding of the International Conference on Software Analysis, Evolution and Reengineering. Klagenfurt: IEEE. 2017. p. 272-282. DOI: 10.1109/saner.2017.7884628.

Appendix A – Overview of Demographics from Studies of Chapter 3

Figure A.1 and Table A.1 present the study participants' demographics from the study presented in section 3.2.

Table A.1 – Overview of Quantitative Data of Participants from the Study Presented in Section 3.2

| Year	Team's Name	# M.Sc. Students	# D.Sc. Students	ESE Group Members?	Part-time Students?
2010	Black	2	1	Yes (1)	Yes (3)
2010	Red	3	0	Yes (1)	Yes (1)
2010	Pink	2	1	Yes (3)	No
2012	Purple	1	2	No	Yes (1)
2012	Blue	1	2	No	Yes (1)
2012	Green	3	0	No	Yes (1)
2012	Yellow	2	1	Yes (2)	Yes (1)
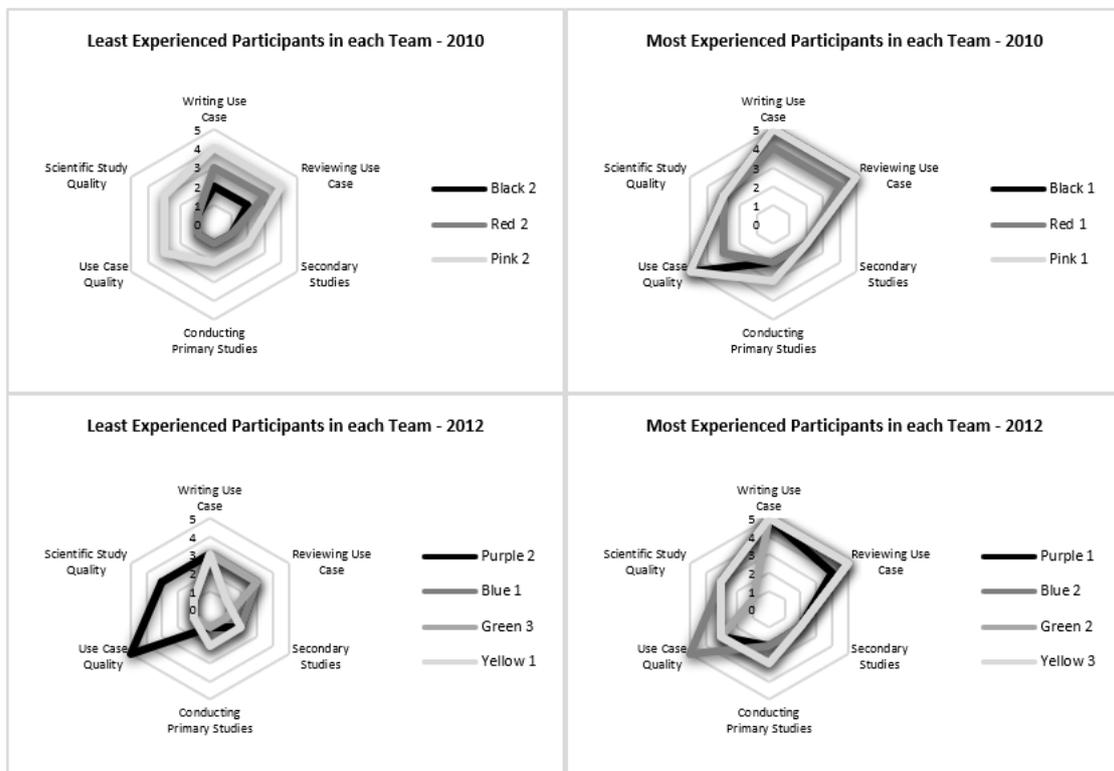
Figure A.1 – Participants Experience in the Main Topics Related to the Assignment from the Study Presented in Section 3.2

Figure A.2 and Table A.2 present the participants' demographics from the studies presented in section 3.3.2.

Table A.2 – Overview of Quantitative Data of Participants from the Study Presented in Section 3.3.2

Study	Quantitative and Description
S1	Thirty-eight undergraduate students (>= third term) of the computer and information engineering course learning software engineering and working on a software project – 19 are classified as novice programmers and 19 as experienced (according to their characterization forms). They participated in the study after delivering the first release of a software product they were developing in Python and before delivering its last release. Their motivation mainly resides in learning the source code quality attributes during a lecture on source code quality to support the improvement of their product for final release delivery.
S2	ten graduate students (eight MSc and two DSc) taking part in the empirical software engineering course with experience in different programming languages (including Python) and software development projects in the software industry – all of them are classified as experienced (according to their characterization forms). They participated in the study in the context of a lecture on empirical studies design and execution. Their primary motivation was on experiencing the participation in a real empirical study on software engineering so that they could use this experience to support the planning and execution of future empirical studies for their research.
S3	Eighteen undergraduate students (first term) of the informatics course learning to program in Python. All of them are classified as novices (according to their characterization forms) and participated in the study as the last activity of their programming classes before the final exam. The study represented an opportunity to review the central concepts for the final exam.
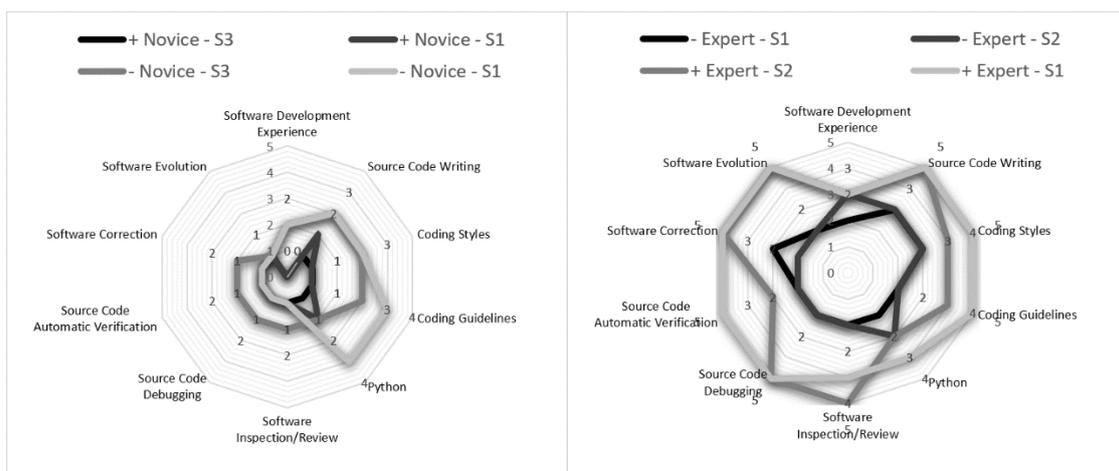
Figure A.2 – Experience Comparison in the Three Studies Presented in Section 3.3.2

Appendix B – Questions Created to Some Concepts During Familiarization

The next tables present some questions created during the familiarization phase of the thematic analysis of the practical questions.

Table B.1 – Questions Created to the Concept of Web Software Development During Familiarization

ID	Question	Intervention	Comparison	Outcome	Population/Context	Possible Answer Source
171203	What are the differences between [server-side programming] and [client-side programming] [in web programming]?	server-side programming	client-side programming	NULL	web programming	state-of-the-art
171203	Which [programming language] should be used [in server-side programming] [in web programming]?	programming language	NULL	NULL	server-side programming, web programming	state-of-the-art
171203	Which [programming language] should be used [in client-side programming] [in web programming]?	programming language	NULL	NULL	client-side programming, web programming	state-of-the-art
46716	What [web application technical practices] should be applied [to achieve stability, performance, security, and functional suitability] [before making a site public]?	web application technical practices	NULL	stability, performance, security, and functional suitability	before making the site public	state-of-the-art
~~99894~~	~~Why does [HTTP] not have [POST redirect] [to redirect users to another domain]?~~	~~HTTP, POST redirect~~	~~NULL~~	~~to redirect users to another domain~~	~~NULL~~	~~state-of-the-practice~~
38691	What are the differences between [Web API] and [Web Services]?	Web API	Web Services	NULL	NULL	state-of-the-practice

Table B.2 – Questions Created to the Object Oriented Development During Familiarization

ID	Question	Intervention	Comparison	Outcome	Population/Context	Possible Answer Source
61376	What are the differences between [aggregation] and [composition]?	aggregation	composition	NULL	NULL	state-of-the-practice
61376	What is [aggregation]?	aggregation	NULL	NULL	NULL	state-of-the-practice
143736	What is [private variable] [in classes] for?	private variable	NULL	NULL	in classes	state-of-the-practice
143736	Where should [private variable] be used?	private variable	NULL	NULL	NULL	state-of-the-practice
143736	What is [public variable] [in classes] for?	public variable	NULL	NULL	in classes	state-of-the-practice
143736	Where should [public variable] be used?	public variable	NULL	NULL	NULL	state-of-the-practice
143736	Where should [private variable] be used over [public variabale] [in classes]?	private variable	public variable	NULL	in classes	state-of-the-practice
34584	How to [explain] [object-oriented terminology and concepts] [to a non-technical person]?	explain object-oriented terminology and concepts	NULL	NULL	to a non-technical person	state-of-the-practice
176049	What are [encapsulation, association, aggregation, and composition]?	encapsulation, association, aggregation, and composition	NULL	NULL	NULL	state-of-the-practice
176049	What are [encapsulation, association, aggregation, and composition] for?	encapsulation, association,	NULL	NULL	NULL	state-of-the-practice

ID	Question	Intervention	Comparison	Outcome	Population/Context	Possible Answer Source
		~~aggregation, and composition~~				
~~176049~~	~~How should [encapsulation, association, aggregation, and composition] be used?~~	~~encapsulation, association, aggregation, and composition~~	~~NULL~~	~~NULL~~	~~NULL~~	~~state-of-the-practice~~
~~176049~~	~~How should [encapsulation, association, aggregation, and composition] be done?~~	~~encapsulation, association, aggregation, and composition~~	~~NULL~~	~~NULL~~	~~NULL~~	~~state-of-the-practice~~

Table B.3 – Questions Created to the Concept of Software Testing During Familiarization

ID	Question	Intervention	Comparison	Outcome	Population/Context	Possible Answer Source
7823	Are [multiple asserts] [in a single unit test] a good thing to do?	multiple asserts	NULL	NULL	single unit test	state-of-the-art
7823	Why are [multiple asserts] [in a single unit test] a good thing to do?	multiple asserts	NULL	NULL	single unit test	state-of-the-art
7823	Where should [multiple asserts] [in a single unit test] be used?	multiple asserts	NULL	NULL	single unit test	state-of-the-art
147134	How to [test] [randomness] [to achieve simple and robust unit tests]?	test randomness	NULL	to achieve simple and robust unit tests	unit test	state-of-the-art

ID	Question	Intervention	Comparison	Outcome	Population/Context	Possible Answer Source
147134	Is [unit test] good to use [for randomness test]?	unit test	NULL	NULL	randomness test	state-of-the-art
147134	What [test methods] should be used [for randomness test]?	test methods	NULL	NULL	randomness test	state-of-the-art
~~155880~~	~~How to [unit test] [multiple conditions] [in an IF statement]?~~	~~unit test multiple conditions~~	~~NULL~~	~~NULL~~	~~IF statement~~	~~state-of-the-practice~~
~~214154~~	~~How to [unit test] [methods that call a web service] [in Hadoop web services]?~~	~~unit test methods that call a web service~~	~~NULL~~	~~NULL~~	~~Hadoop web services~~	~~state-of-the-practice~~

Table B.4 – Questions Created to the Concept of Database During Familiarization

ID	Question	Intervention	Comparison	Outcome	Population/Context	Possible Answer Source
150669	Is [storing large files] [in the database] not a good thing to do?	storing large files	NULL	NULL	in database	state-of-the-art
150669	Why is [storing large files] [in the database] not a good thing to do?	storing large files	NULL	NULL	in database	state-of-the-art
150669	Is [storing large files] [in a file system] a good thing to do?	storing large files	NULL	NULL	in file system	state-of-the-art
150669	Why is [storing large files] [in a file system] a good thing to do?	storing large files	NULL	NULL	in file system	state-of-the-art
150669	Which [storing file method] should be used [to avoid slowing down data access] [in cases of large files]?	storing file method	NULL	to avoid slowing down data access	large files	state-of-the-art

190482	Why should [storing data in the database] be done over [saving data into disk]?	storing data in the database	saving data into disk	NULL	NULL	state-of-the-art
190482	Why is [database] used?	database	NULL	NULL	NULL	state-of-the-art
54373	Where should [non-relational databases] be used over [relational databases]?	non-relational databases	relational databases	NULL	NULL	state-of-the-art
54373	What are the differences between [non-relational databases] and [relational databases]?	non-relational databases	relational databases	NULL	NULL	state-of-the-art
54373	What is [non-relational database] for?	non-relational databases	NULL	NULL	NULL	state-of-the-art
~~120178~~	~~What are the differences between [MariaDB] and [MySQL]?~~	~~MariaDB~~	~~MySQL~~	~~NULL~~	~~NULL~~	~~state-of-the-practice~~
~~120178~~	~~What are the similarities between [MariaDB] and [MySQL]?~~	~~MariaDB~~	~~MySQL~~	~~NULL~~	~~NULL~~	~~state-of-the-practice~~
~~120178~~	~~What are the differences between [MariaDB query language] and [MySQL query language]?~~	~~MariaDB query language~~	~~MySQL query language~~	~~NULL~~	~~NULL~~	~~state-of-the-practice~~
~~288370~~	~~How to [synchronize] [databases] [to guarantee performance] [in two different SGBDs (from MySQL to DB2) with different and big schemas]?~~	~~synchronize databases~~	~~NULL~~	~~to guarantee performance~~	~~in two different SGBDs (from MySQL to DB2) with different and big schemas~~	~~state-of-the-practice~~

Table B.5 – Questions Created to the Concept of Programming Practices During Familiarization

ID	Question	Intervention	Comparison	Outcome	Population/Context	Possible Answer Source
49550	What [hashing algorithm] should be used [to achieve uniqueness and speed]?	hashing algorithm	NULL	to achieve uniqueness and speed	NULL	state-of-the-art
297160	Why does [mergesort] have [O(log n) complexity]?	mergesort, O(log n) complexity	NULL	NULL	NULL	state-of-the-practice
77757	What are the differences between [pseudo code] and [algorithm]?	pseudo code	algorithm	NULL	NULL	state-of-the-practice
77757	Where should [the word pseudo code] be used over [the word algorithm]?	pseudo code	algorithm	NULL	NULL	state-of-the-practice
146021	How to [determine] [an algorithm is O (log n)]?	determine an algorithm is O (log n)	NULL	NULL	NULL	state-of-the-practice
124233	Why can not [random numbers] be produced [by a deterministic device]?	random numbers	NULL	NULL	by a deterministic device	state-of-the-practice
194646	What [programming methods] should be used [to avoid stack overflow] [in a recursive algorithm]?	programming methods	NULL	to avoid stack overflow	in a recursive algorithm	state-of-the-practice
279004	How to [convert] [loop (while/for)] [to recursion]?	loop (while/for)	recursion	NULL	NULL	state-of-the-practice

279004	How to [convert] [recursion] [to loop (while/for)]?	recursion	loop (while/for)	NULL	NULL	state-of-the-practice
183842	Is [initializing a char[] with a string literal] not a good thing to do [in C]?	initializing a char[] with a string literal	NULL	NULL	C	state-of-the-practice
183842	Why is [initializing a char[] with a string literal] not a good thing to do [in C]?	initializing a char[] with a string literal	NULL	NULL	C	state-of-the-practice
165725	Where should [branching] be used over [tagging] [in Git]?	branching	tagging	NULL	Git	state-of-the-practice
127178	Is [reuse of id] not a good thing to do [in HTML]?	reuse of id	NULL	NULL	HTML	state-of-the-practice
127178	Why is [reuse of id] not a good thing to do [in HTML]?	reuse of id	NULL	NULL	HTML	state-of-the-practice
127178	Is [reuse of id] a good thing to do [in HTML]?	reuse of id	NULL	NULL	HTML	state-of-the-practice
127178	Why is [reuse of id] a good thing to do [in HTML]?	reuse of id	NULL	NULL	HTML	state-of-the-practice
20909	What are [procedures]?	procedures	NULL	NULL	NULL	state-of-the-practice
20909	What are [methods]?	methods	NULL	NULL	NULL	state-of-the-practice
20909	What are [functions]?	functions	NULL	NULL	NULL	state-of-the-practice
20909	What are [subroutines]?	subroutines	NULL	NULL	NULL	state-of-the-practice
20909	What are the differences among procedures, methods, functions, and subroutines?	procedures	methods, functions, subroutines	NULL	NULL	state-of-the-practice