# Technical Report


## Towards Efficient Support for Theory Identification from Published Experimental Research Reports

Stefan Biffl[1], Marcos Kalinowski[2], Fajar J. Ekaputra[1],
Guilherme G. Martins[2], Dietmar Winkler[1]

[1]Christian Doppler Laboratory CDL-Flex, Institute of Software Technology and Interactive Systems, Vienna University of Technology
Favoritenstrasse 9-11/188, AT 1040 Vienna, Austria
{stefan.biffl, fajar.ekaputra, dietmar.winkler}@tuwien.ac.at

[2]Federal University of Juiz de Fora, NEnC,
Rua José Kelmer s/n 36036-330 Juiz de Fora, Brazil
{Kalinowski, guilherme}@ice.ufjf.br

Technical Report No. IFS-CDL 14-01
Issued: January 2014

# Towards Efficient Support for Theory Identification from Published Experiment Research Reports

Stefan Biffl[a] Marcos Kalinowski[b] Fajar J. Ekaputra[a] Guilherme G. Martins[b] Dietmar Winkler[a]

[a] Vienna University of Technology
CDL-Flex[1], Favoritenstr. 9/188
1040 Vienna, Austria
+43 1 58801 18810
<firstname>.<lastname>@tuwien.ac.at

[b] Federal University of Juiz de Fora
NEnC, Rua José Kelmer s/n
36036-330 Juiz de Fora, Brazil
+55 32 2102-3311
{kalinowski, guilherme}@ice.ufjf.br

## ABSTRACT

[Context] Theory identification (TI) from research following the hypothetical path aims at making SE theory from published empirical research explicit. Challenges for TI include limitations of searching for theory in digital libraries and the absence of a platform with semantic support for theory identification and construct definition. [Objective] The aim of this paper is to provide process and tool support for efficiently identifying theory elements from published experiment research reports. [Method] We propose supporting the analysis of experimental evidence for theory element identification with a knowledge base (KB) and a glossary, providing semantically enabled functions for identifying and defining theory constructs. We evaluate the process and tool support in the context of the software inspection method Perspective-Based Reading (PBR). [Results] The proposed support helped effectively identifying 23 PBR theory constructs and candidate propositions. Theory element identification was found notably more efficient when compared to typical identification approaches without the KB and glossary support. [Conclusions] The support showed promising results when applied to PBR experiments and should be investigated in a wider area of empirical research.

## Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software Validation
I.2.4 [**Artificial Intelligence**]: Knowledge Representation

## General Terms

Measurement, Experimentation, Theory, Verification.

## Keywords

Theory building, theory identification, empirical evidence, software inspection, perspective-based reading.

## 1. INTRODUCTION

An important development in software engineering (SE) research has been the rise of evidence-based SE [1] to investigate a wide range of SE phenomena [2] in published empirical studies. Following a theory-building approach [3], SE researchers collaborate to generate, evolve, and evaluate theories on topics, such as defect detection methods for software inspection [4]. In this context, researchers apply a variety of research strategies to generate new theory and theory propositions (e.g., grounded theory) and to evaluate hypotheses derived from such propositions (e.g., by conducting experiments) for evolving theories [5]. However, in SE the hypotheses investigated by empirical studies are often not derived from theory propositions and despite the growing momentum of empiricism, theory building and evidence do not interact sufficiently [6]. Few empirical studies in SE relate phenomena under investigation to the underlying theory. Johnson et al. [7] argue that SE research is full of implicit theory, e.g., the assumption from practical observation that applying requirements inspections have positive impact on quality [4].

Theory identification (TI) from research following the hypothetical path [6] aims at revealing implicit SE theory in published empirical research. Nevertheless, it seems to be no clear process to identify theory and make it explicit for analysis and discussion in the scientific community. Typical TI activities include searching for empirical studies, extracting relevant empirical evidence, and analyzing the evidence to identify theory elements and relations, such as constructs and proposition candidates [3].

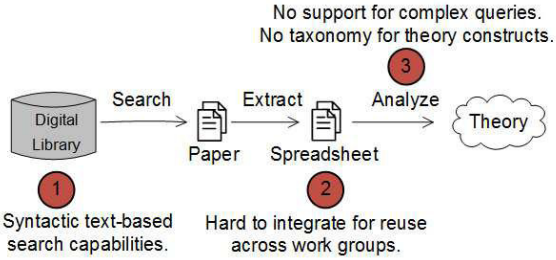



**Figure 1. Challenges in theory identification from published empirical research.**

Figure 1 illustrates challenges related to those TI activities: (1) Searching for published research is done in digital libraries, which do not provide structured access to relevant theory and empirical concepts (e.g., theory constructs, propositions, and hypotheses) and rely on syntactic search capabilities. (2) Extracted evidence is typically stored in spreadsheets, which are hard to integrate for reuse across work groups. (3) There is no support for complex queries to facilitate analyzing the extracted empirical evidence and no taxonomy defining relevant theory constructs in the different SE research topics.

In this paper we address these challenges aiming at supporting SE theory identification from published empirical research. We focus on empirically-based theories; i.e., theories that are built or evolved from empirical research [3] and on recovering such theory from a specific type of empirical study: experiments.

As method to support theory identification we build on previous work [8], the Systematic Knowledge Engineering (SKE) process

[1] "Christian Doppler Laboratory for "Software Engineering Integration for Flexible Automation Systems", Institute of Software Technology and Interactive Systems.

and its tool support[5]. SKE is based on the Systematic Literature Review (SLR) process [9] and Knowledge Engineering (KE) [10] practices to provide a Knowledge Base (KB) with semantic technologies that enables querying for empirical evidence. The resulting KB stores information on domain concepts of the research topic linked to the available empirical evidence, represented by concepts of empirical studies [5] (e.g., investigated hypotheses, treatments, response variables, results, and study findings).

The use of KE semantic technology with ontologies, embedded within the SKE tool, facilitates querying the KB on domain concepts, e.g., on synonyms and related concepts, which goes beyond the syntactic search capabilities of typical digital libraries and spreadsheets. In order to allow using the KB to support analyzing empirical evidence for TI we designed queries to facilitate identifying the theory elements presented in [3]. We also designed a glossary tool[5] to support the definition of relevant theory constructs.

We illustrate and evaluate the proposed process and tool support in the context of TI for the software inspection method Perspective-Based Reading (PBR). Applying SKE allowed integrating empirical evidence from 14 identified PBR experiments into the KB. Analyzing the query results enabled identifying 23 theory constructs, defining them in the glossary, and designing a cause-effect graph for representing theory proposition candidates.

Major findings of the evaluation were that the proposed process and tool support worked well and that the researchers found the provided query and glossary facilities usable and useful to facilitate analyzing empirical evidence for TI. Theory element identification was found notably more efficient when compared to typical identification approaches without the KB and glossary support (e.g., analyzing empirical evidence contained in spreadsheets for TI). Additionally, the proposed support with the online KB and glossary enable reuse of knowledge within scientific communities and can therefore be used beyond the scope of local work groups. Given those findings, we believe that the approach can represent a step towards reverse engineering SE theory on specific topics from published empirical research both with local research work groups and for cooperating work groups in a scientific community.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 motivates the research issues. Section 4 describes the proposed TI support illustrated with the evaluation use case of TI from PBR experiments. Sections 5 and 6 discuss the evaluation results, threats to validity, and lessons learned. Section 7 summarizes the research results and proposes future research.

# 2. BACKGROUND

This section describes the theoretical foundations for this research: Theory Building in SE (Section 2.1), Systematic Literature Reviews (Section 2.2), Knowledge Base Design and Population (Section 2.3), and Systematic Knowledge Engineering (Section 2.4). Further, we describe the foundations on Software Inspections (Section 2.5) as input to the evaluation use case.

## 2.1 Theory Building in Software Engineering

The importance of SE theory has been claimed by several researchers, but there has been little focus on how theory should be described and built [3]. In this sense, Sjøberg et al. [3] suggest to divide the description of a theory in four parts: the constructs (basic elements), propositions (how the constructs interact), explanations (why the propositions are as specified), and scope (in which context the theory is applicable).

Theory can exist at different levels of sophistication or complexity. Yin [11] presents three of such levels, adapted to the SE context by Sjøberg et al. [3] as:

- Level 1. Minor working relationships that are concrete and based directly on observations.
- Level 2. Theories of the middle-range that involve some abstraction but are still closely linked to observations.
- Level 3. All-embracing theories that seek to explain SE.

According to Sjøberg et al. [3], the development of SE theories from scratch is in early stages, and immediate efforts focus primarily on levels 1 and 2. They propose a diagrammatic notation for representing SE theory. In this notation, each construct should belong to one of four archetype classes: Actor, Technology, Activity, or Software System. These archetypes are related to typical SE phenomena, in which actors apply technologies to perform certain activities on a software system [3].

Johnson et al. [7] argue that SE research is full of implicit theory. Having this in mind, Stol and Fitzgerald [6] investigated ways of uncovering middle-range (Level 2) theories in SE. They focused on extracting "theory fragments" (partial theory that has not been completely developed yet) from single research papers. Therefore, they consider three research paths for theory building:

- Study design path. This path comprises engineering solutions for an element of the topic of study (e.g., developing a technique or tool to implement or support a conceptual model).
- Observational path. The goal of this path is to collect a set of observations and to explain them in terms of a set of meaningful concepts (for instance, using Grounded Theory [12]).
- Hypothetical path. This path refers to research that seeks to test theory, for instance, conducting experiments. In particular, there are two sources of potential hypotheses; they can originate from the topic of study (substance-driven research) or from a theory (concept-driven research).

Since the rise of evidence-based SE [1] empirical studies (following the hypothetical path) are being undertaken more frequently [2]. However, most of those studies are subject-driven and not directly related to theory. Thus, interest sprouts in TI from research following the hypothetical path. Typical activities in this context are searching for empirical studies, extracting empirical evidence, and analyzing the evidence to identify theory elements, such as constructs and proposition candidates [3].

We believe that SLRs can help to gather empirical evidence on a given research topic. Therefore, the foundations on SLRs follow.

## 2.2 Systematic Literature Reviews

Kitchenham and Charters [9] developed guidelines for performing Systematic Literature Reviews (SLRs) in the SE domain. In those guidelines they state that the main reasons for conducting SLRs are (a) summarizing the existing evidence concerning a treatment or technology; (b) identifying gaps in current research in order to suggest areas for further investigation; and (c) providing background to appropriately position new research.

The SLR guidelines [9] summarize three main phases of a systematic review: (a) Planning the Review, (b) Conducting the

Review, and (c) Reporting the Review. The PICO (Population, Intervention, Comparison, Outcome) strategy [13] has been suggested [9] in the planning phase for detailing the research question elements in order to support developing the review protocol. Conducting the review comprises identifying, selecting and assessing the quality of primary studies so that data can be extracted to facilitate data synthesis. A typical intermediate result of a SLR is a spreadsheet with extracted data to enable answering research questions. Reporting the review focuses on summarizing the SLR results with focus on pre-defined research questions. Lessons learned from applying SLRs to the SE domain are reported in [2].

In the context of this research, the main advantage of using SLRs is allowing to systematically identify evidence on a specific topic and afterwards enable incremental updates. An example of such updates is available in [14], where four independent SLR trials were conducted in different years to incrementally build evidence-based guidelines on defect causal analysis.

SLR reports and extracted spreadsheet data (i.e., intermediate SLR results) represent the foundation for addressing individual research questions, the starting point of an SLR. However, for TI researchers want the capability to query on empirical evidence (e.g., for hypotheses, factor treatments, and response variables) to analyze results for theory elements. Therefore, we believe that a KB can provide good tool support for bottom-up theory building.

## 2.3 Knowledge Base Design and Population

The process of building a knowledge base may be seen as a modeling activity [10]. Building a knowledge base means building a computer model with problem-solving capabilities comparable to a domain expert. For creating a knowledge base, it is essential to capture domain knowledge through content-specific agreements, so both human and knowledge-based systems can access and use the information [15]. For this purpose, formal ontologies have been successfully used since the 1990s [16]. Ontologies can provide standard terminologies and rich semantics to facilitate knowledge sharing and reuse [10]. OWL DL (Web Ontology Language - Description Logic) is the most widely used language for ontologies as it has the capability of supporting semantic interoperability to exchange and share context knowledge between different systems, and keeps a balance between expressiveness and automatic processing. In addition, ontologies enhance searching mechanisms, which may refer to precise semantic concepts rather than simple syntactic keywords, facilitating the use of the knowledge stored in the ontology [15].

Once the ontology or the data model of the KB is defined, it is necessary to capture the extracted data from information resources in accordance to the KB. This process is called KB population, and involves the creation, transformation and integration of individuals (instances) into the KB. In our case, the information resources for creating the KB are empirical study reports. The KB population process may face integration problems if the different information resources use varying structures to represent the same concepts. The Interchange Standard Approach, has been stated as one of the best solution options for semantic integration [17]. The currently available tools to manage ontologies usually require ontology experts. Therefore, ontology non-experts need to be provided with effective and efficient interfaces for both, importing and exporting knowledge, and for querying.

Concerning the use of such KB in the empirical SE context, the SKE process has been proposed for systematically building a KB containing empirical evidence on a specific research topic.

## 2.4 Systematic Knowledge Engineering

The SKE process [8] uses SLR-based empirical study identification and KB integration to support knowledge reuse and extension and semantic querying on empirical evidence.

The key innovation comes from decoupling data extraction from data synthesis (in SLRs both conducted within the Conducting the Review phase) by integrating extracted data into a KB rather than using it to apply a particular synthesis method for answering a specific research question in the format of a SLR report. The KB enables querying with structured access to concepts, such as, hypothesis, factors and response variables, facilitating to explore the evidence for different synthesis purposes, such as TI.

It comprises three stages [8]: Planning KB Creation, Conducting Data Extraction, and Creating/Updating the KB. Details on each of these stages follow:

- Planning KB Creation. The main goal of this phase is developing a review protocol to enable systematically identifying relevant primary studies. In SKE, differently from SLRs, there are no specific research questions, but a pre-defined purpose of building a KB on empirical evidence on a given research topic. This facilitates building the protocol based on a specific configuration of the PICO strategy [13]. In this configuration, the population represents the specified research topic. The intervention represents the specified empirical study types. The comparison is blank and the outcome represents the elements to extract from the empirical studies (e.g., hypotheses, findings).
- Conducting Data Extraction. This phase consists of following the protocol's search, selection, and assessment strategies and extracting data from the identified studies, according to information to be loaded into the KB's data model. Differently from the SLR process, data synthesis is not part of this second phase.
- Creating/Updating KB. In this phase, the knowledge engineer applies KE practices to design (or update) the KB data model and to populate it by integrating the extracted data. This role is also responsible for providing query facilities. Those facilities allow other researchers to query the KB on empirical evidence and using the results of such queries as input to apply their own research synthesis methods, according to their specific goals.

The SKE process is tool supported [8], the KB, implemented using the Protégé[1] framework, uses semantic technology with ontologies to facilitate semantic searches [18]. Besides the KB, the tool support comprises a spreadsheet data contribution interface and a web prototype for querying. The data contribution interface was automated in Java by using a spreadsheet reader library (e.g., Apache POI[2]) and an ontology library (e.g., Apache

[1] Protégé: http://protege.stanford.edu/

[2] Apache POI: http://poi.apache.org/

[3] Apache Jena; http://jena.apache.org/

[4] SPARQL: http://www.w3.org/TR/rdf-sparql-query/

[5] SKE-Tool: http://cdlflex.org/prototypes/ske/theory

Jena[3]). The Interchange Standard Approach integration [17] can be applied for heterogeneous data integration. The queries of the web prototype are implemented using SPARQL[4] query language. Using ontology-specific features, the knowledge engineer enhanced the KB by implementing semantic search functions (e.g., searching on synonyms and related concepts). The SKE-Tool prototype is available online[5].

A concrete example of applying SKE can be found in [8], where a KB with empirical evidence acquired through experiments was built for software inspections (integrating data from the 31 most recent identified research papers, ranging from 2006 to 2013). We then elicited relevant stakeholder queries on empirical evidence on software inspections through a survey with empirical SE researchers from 6 research groups. The knowledge engineer designed the queries and the KB allowed efficiently obtaining accurate results for them. Software inspections are also related to the evaluation use case of this paper.

## 2.5 Software Inspections

Software Inspections (SI) improve product quality by the analysis of software artifacts, detecting defects for removal before these artifacts are delivered to following software life cycle activities [4]. The traditional software inspection process by Fagan [19] involves a moderator planning the inspection, inspectors reviewing the artifact, a team meeting to discuss and register defects, passing the defects to the author for rework, and a final follow-up evaluation by the moderator on the need of a new inspection. In this context, inspection methods represent a means for supporting inspectors in detecting defects during their individual reviews. These methods include ad-hoc reading, checklist-based reading, and reading techniques, such as Perspective-Based Reading (PBR) [20] and Usage-Based Reading (UBR) [21]. Practitioners typically want to know which method is well suited to their context to find significant defects effectively and efficiently with the available skills of inspector candidates.

Reading techniques [22] require inspectors to work systematically and actively with the target artifact, providing guidance on how to read it and on what to look for. The research in this paper focuses on PBR. The basic idea of this technique is that reviewers assume one of several stakeholder perspectives, e.g., user, developer, or tester, to detect defects so the union of perspectives provides an extensive coverage of the entire artifact [20]. In this way, PBR is expected to offer benefits, such as increased effectiveness, goal-orientation, and transferability via training [20].

Several experiments have been conducted to evaluate expected benefits, investigating a range of hypotheses concerning PBR. Ciolkowski [23] applied a quantitative aggregation strategy to provide summarized information on 12 PBR experiments to investigate whether PBR improves effectiveness when compared to other inspection methods. His findings showed that there was no clear positive effect of PBR. Compared to ad-hoc inspection PBR was more effective, but compared to checklist-based inspections PBR was more effective when inspecting design documents and code, but not when inspecting requirements. While Ciolkowski's analysis [23] mainly concerned effectiveness, those experiments investigate a wide range of hypotheses. In terms of theory building, the experiments follow substance-driven research on the hypothetical path [6]. Therefore, the acquired knowledge is not organized in terms of the underlying theory and an overview of the observations and evidence is missing.

# 3. STRATEGY AND RESEARCH ISSUES

The overall goal of this research is taking a step towards supporting TI on a given research topic based on published experiment research reports. The idea is trying to bottom-up reverse engineer Level 1 theory from substance-driven research, generated following the hypothetical path (see also Section 2.1). In software inspection, like in other SE topics, there is no consolidated overview on theory [7], which makes a top-down approach difficult.

The strategy to address the challenges illustrated in Figure 1 consists of providing a platform where researchers can query empirical evidence to analyze it for identifying theory elements, and define and discuss terms related to theory constructs. Figure 2 shows this strategy. Concerning the challenges related to the TI activities, the search and extract activities are conducted by applying SKE. While the search still relies on syntactic text-based search capabilities of digital libraries (1), extracted data is stored in an extensible KB, which facilitates integration across work groups (2). The analysis (3) is supported by online querying the KB for theory elements and enabling the definition of identified theory constructs in a glossary.

It is noteworthy that, depending on the extent of the KB (e.g., if data of relevant empirical studies has already been extracted and integrated by different work groups) the search for empirical evidence (1) could also be performed directly on the KB, using its semantic search capabilities. Figure 2 also shows the role of the knowledge engineer, supporting the SKE process (conducted by empirical SE experts, familiar to SLRs) by providing the data model and query updates, and maintaining the KB.

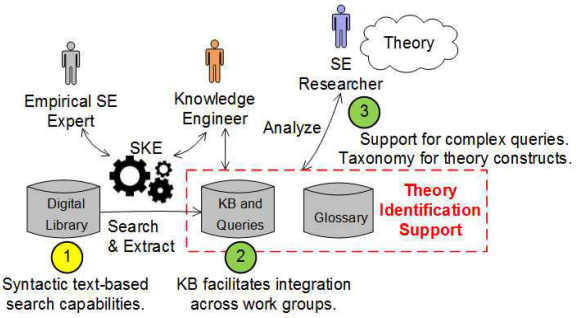



**Figure 2. Strategy to Support Theory Identification from Published Research Results.**

Considering this strategy, we derive three research issues (RIs) related to the steps of the overall approach. RI-1: how to design the KB data model and how to explore this data model to design queries for efficiently supporting the TI analysis activity. RI-2: how to populate the KB appropriately with data from relevant experiment research reports to enable high-quality query results. RI-3: how to use the proposed KB and glossary support in the TI analysis activity to identify theory elements. Thus, the first two RIs are related to preparing the TI support, while the third RI is related to applying the support for identifying theory.

**RI-1: KB Data Model and Queries.** What are the relevant data elements contained in experiments? How to link such data to the specific research topics of the area of interest? How to query such data for identifying theory elements?

We analyzed the data model defined in our previous work for hosting data on experiments on software inspections [8], which already was designed to contain the relevant data elements on

experiments linked to inspection topics. We considered that the same data model can be used as a basis for identifying theory elements. We follow the suggestion of Sjøberg et al. [3] on relevant theory elements (i.e., constructs, propositions, explanations, and scope) and investigate how this data model can be explored to identify those theory elements.

**RI-2: KB Population.** How to systematically import data on relevant experiments into the KB with appropriate data validation?

To address this research issue we suggest applying the SKE process [8] for building the KB from published experiments. Key idea is to conduct SKE's data extraction step based on the structure of the defined data model (see RI-1). SKE's resulting KB can then be queried on empirical evidence according to specific needs. To address theory identification needs, in our evaluation use case we designed and implemented the queries to reveal theory elements based on the investigation done concerning RI-1.

**RI-3: Identifying Theory.** How to use the proposed support to analyze empirical evidence for identifying theory from substance-driven experimental research?

To support this analysis, SE researchers can use the online KB queries to help identifying theory elements and register identified theory constructs into the glossary. They may also represent candidate theory propositions. Therefore, they can use the diagrammatic notation defined in [3], although we faced some practical issues in doing so and ended up using a cause-effect graph (as described in further details in Section 4). On the other hand, researchers can also look for theory constructs in the glossary, and then use the KB to query related proposition candidates.

# 4. THEORY IDENTIFICATION SUPPORT

The following subsections provide details on how each of the research issues was addressed in our PBR evaluation use case. Note that the sequence of activities in the research issues can be seen as a process template for supporting theory identification on other research topics. If a SKE KB has already been designed and populated on the topic of interest the effort for analysis of the theory elements can significantly benefit from reusing the existing knowledge and directly addressing RI-3, which again is facilitated if theory constructs are already defined in the glossary.

## 4.1 RI-1: KB Data Model and Queries

We analyzed and adapted the data model designed in our previous work for hosting data on software inspection experiments [8], which already contains the relevant data elements on experiments linked to inspection topics. This model is based on the areas shown in Figure 3, which represent a high-level abstraction view on the context in which empirical studies are conducted. Empirical studies have data and artifacts, contribute to a Body of Knowledge (BoK) on specific topics, and are performed by researchers who provide publications.

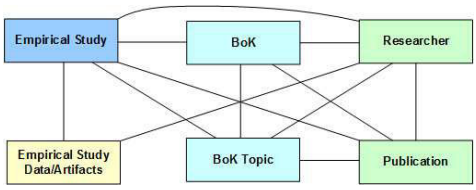



**Figure 3. Major data model areas adapted from [8].**

Figure 4 shows the data model entities for hosting data on experiments, based on these areas and on experimental concepts described by Wohlin et al. [5].

To link data from the empirical studies to the inspection BoK topics, each topic was designed as relating to a set of inspection parameters, extended from the list of parameters by Laitenberger and Debaud [24]. This tailoring is shown in Figure 5. The complete data model is available online[5].

This data model allows querying for evidence from experiments available on specific inspection BoK topics with queries, such as: "Which hypotheses have been investigated by experiments on PBR?" In this case, the query result lists the hypotheses of all experiments related to BoK topics with parameter "inspection method" equal to PBR (and its synonyms). According to specific needs, it is also possible to list the results for each hypothesis in the available experiment runs (confirmed/rejected) and information on their statistical confidence. Moreover, the measurements that led to each of those results can be obtained.

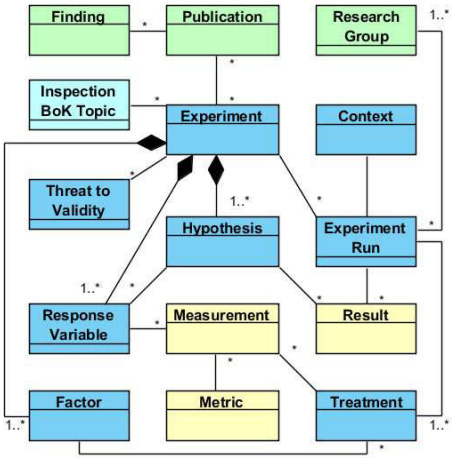



**Figure 4. KB data model overview adapted from [8].**

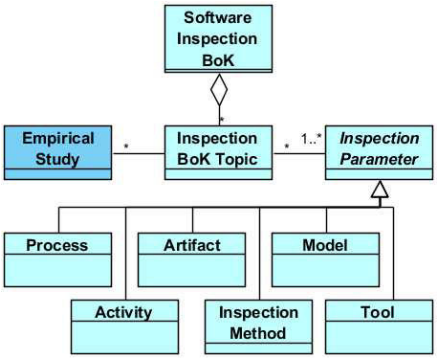



**Figure 5. Empirical studies linked to inspection BoK topics adapted from [8].**

In our opinion, a data model describing experiments as shown in Figure 4 and relating them to BoK topics, as suggested in Figure 5 (topics are combinations of relevant parameters), can be used as a basis for identifying empirically-based theory elements from experiments on different BoK topics. Therefore, we investigated how the data model can be explored to help identifying the theory elements suggested by Sjøberg et al. [3]: constructs, propositions, explanations, and scope. Figure 6 depicts the result of our investigation.

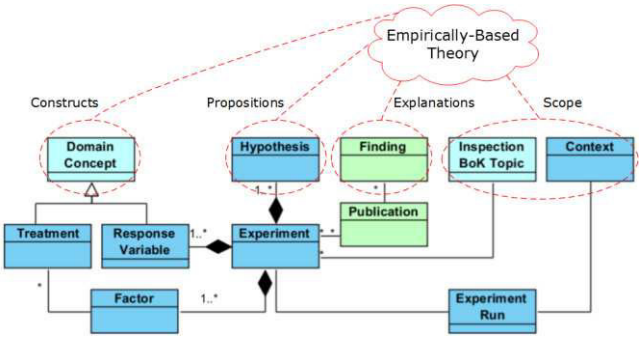



**Figure 6. Identifying theory elements.**

We believe that, concerning the concepts of experiments, in general, abstracting domain concepts from factor treatments and response variables can help to identify theory constructs. Regarding propositions, the investigated hypotheses can provide useful insights. For identifying explanations, sometimes the textual description of findings included by authors in publications can be helpful. Finally, the scope can be delimited based on the investigated BoK Topic (and the configuration of its parameters) and on the context of the experiment runs. Having this in mind, in our evaluation use case we found the following queries of interest to support theory identification on PBR based on the KB's data model:

- Constructs/FT: Which domain concepts have been used as factor treatments in experiments on inspection method PBR?
- Constructs/RV: Which domain concepts have been used as response variables in experiments on inspection method PBR?
- Propositions/Hyp: Which are the hypotheses investigated in experiments on inspection method PBR?
- Propositions/HypDC: Which are the hypotheses investigated in experiments on inspection method PBR that include domain concept [domain concept] and synonyms?
- Propositions/HypFT: Which are the hypotheses investigated in experiments on inspection method PBR and construct [factor treatment] and synonyms?
- Propositions/HypRV: Which are the hypotheses investigated in experiments on inspection method PBR and construct [response variable] and synonyms?
- Explanations: Which are the reported findings of experiments on inspection method PBR?
- Scope: What are the BoK Topics (parameters) and contexts of experiments on inspection method PBR?

Having defined the KB data model and identified relevant queries for identifying candidate theory elements, interest sprouts in KB population (RI-2).

## 4.2 RI-2: KB Population

We propose applying SKE for systematically populating the KB with data on relevant experiments. To populate the KB in our PBR evaluation use case, we incrementally extended the content of the software inspection KB created in previous work [8]. In that occasion, following the search protocol defined in SKE's planning phase, 102 research papers with experiments on software inspections (14 of them concerning PBR) were identified. Data had been extracted from the 31 most recent ones (5 of them on PBR) by a team of six independent empirical SE experts and integrated into the KB by a knowledge engineer.

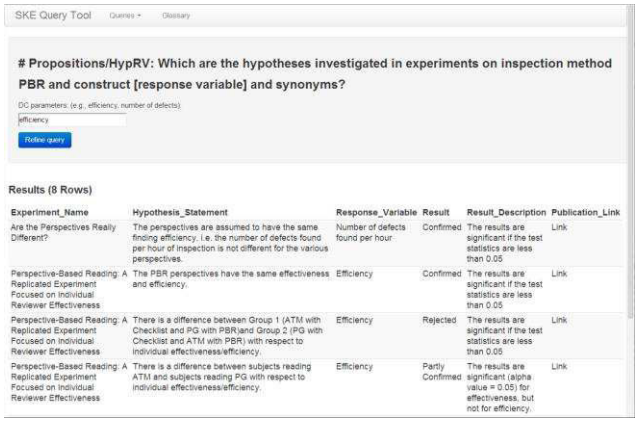
As in SLRs, the SKE protocol includes the search strategy, the study selection criteria, the quality assessment procedures, and the data extraction strategy. In our case, there was no extensive effort invested in getting a complete set of inspection experiments, rather a fair and objective sample to use as evaluation use case. Thus, a single digital library was chosen: Scopus, which according to [9] claims to be the largest database of abstracts. The study selection and quality assessment criteria were also relaxed. The study should be an experiment identifying investigated hypotheses and published in a peer-reviewed publication medium. Details on the search string derived from SKE's suggested PICO configuration's synonyms can be found in [8]. For data extraction, a spreadsheet template was prepared to enable gathering the information to be loaded into the KB's data model.

To allow identifying theory elements on PBR we completed data extraction of the remaining 9 identified PBR experiments and integrated the extracted data into the KB. As in the previous case, data extraction took on average 2 person hours per paper. Data checking took additional 0.5 person hours per paper. Data integration using the interface for KB data contributions was automated and took less than one minute. Once the data was integrated into the KB, the KE designed the queries so that they could be used to support theory identification (RI-3) together with the glossary tool's online term definition facility.

## 4.3 RI-3: Identifying Theory

To address this research issue, we propose that researchers use the support of the online KB queries to identify theory elements and of the online glossary to register and define identified theory constructs. They may also represent candidate theory propositions. If the online glossary already contains theory construct definitions, researchers can also look for those constructs in the glossary and use the KB to query related proposition candidates.

In the PBR evaluation use case the glossary initially did not hold entries in advance. Therefore, they were registered as they were identified. Thus, we started by executing the queries. Figure 7 shows a screenshot with partial results for query Propositions/HypRV. This query lists all the hypotheses (8) and results for experiments containing efficiency or synonyms in their response variables. Running the queries facilitated analyzing the experiment data in order to identify theory elements. Complete results of the queries can be seen online[5]. Some of the identified theory elements are discussed hereafter.


SKE Query Tool Queries ▾ Glossary

**# Propositions/HypRV: Which are the hypotheses investigated in experiments on inspection method PBR and construct [response variable] and synonyms?**

DC parameters: (e.g., efficiency, number of defects)

efficiency

Refine query

**Results (8 Rows)**

| Experiment_Name | Hypothesis_Statement | Response_Variable | Result | Result_Description | Publication_Link |
|---|---|---|---|---|---|
| Are the Perspectives Really Different? | The perspectives are assumed to have the same finding efficiency, i.e. the number of defects found per hour of inspection is not different for the various perspectives. | Number of defects found per hour | Confirmed | The results are significant if the test statistics are less than 0.05 | Link |
| Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness | The PBR perspectives have the same effectiveness and efficiency. | Efficiency | Confirmed | The results are significant if the test statistics are less than 0.05 | Link |
| Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness | There is a difference between Group 1 (ATM with Checklist and PG with PBR)and Group 2 (PG with Checklist and ATM with PBR) with respect to individual effectiveness/efficiency. | Efficiency | Rejected | The results are significant if the test statistics are less than 0.05 | Link |
| Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness | There is a difference between subjects reading ATM and subjects reading PG with respect to individual effectiveness/efficiency. | Efficiency | Partly Confirmed | The results are significant (alpha value = 0.05) for effectiveness, but not for efficiency. | Link |

**Figure 7. Hypotheses investigated on inspection method PBR.**

Based on the results of query Propositions/Hyp it could be seen that overall 90 hypotheses (including null and alternative hypotheses) were investigated by the 14 PBR experiments. By analyzing those hypotheses and accessing the results of queries Constructs/FT and Constructs/RV a set of 23 constructs, which enable to express the hypotheses used in the experiments, were identified. Due to space restrictions, the complete set of identified constructs is available only online in the Glossary tool[6]1. A short description of 12 key theory constructs follows:

- C01 Effi – Inspection Efficiency: Number of defects detected per unit of effort.
- C02 Effe – Inspection Effectiveness: Percentage of defects detected during the inspection (0 to 100%), after a team meeting or as consolidated outcome of the defect reports from all team members in a nominal team.
- C03 Effo – Inspection Effort: Sum of all individual defect detection efforts and the team meeting effort (person hours).
- C04 TMeetEffe – Team Meeting Effectiveness: Percentage of defects detected after the inspection meeting (0 to 100%).
- C06 TMeetEffo – Team Meeting Effort: Effort from discussion of the individual defect reports in the inspection team.
- C07 TDDT – Team Mix of Defect Detection Techniques (DDTs): Set of DDTs used in the team.
- C10 IEffe – Individual Effectiveness: Percentage of defects detected by an inspector (0 to 100%).
- C11 IDDT – Inspector Defect Detection Technique (DDT): The specific DDT used by an inspector, e.g., PBR.
- C13 IEffo – Inspector Effort: Duration of the defect detection activity for one inspector (hours).
- C14 SAT – Software Artifact Type: Requirements/design/code artifact and relevant languages used.
- C15 SAS – Software Artifact Size: Size of the artifact.
- C16 TDN – Total Defect Number: Number of defects in the inspected artifact.

Although all the constructs are related to the analyzed experiment hypotheses, the most addressed constructs in those hypotheses were related to effectiveness, C10 IEffe (40%) and C02 Effe (22%). Given the absence of a taxonomy, we found many variations in the terms and semantics of PBR experiment reports while identifying constructs. The KB queries made it easy to see the variety, facilitating the construct identification. The Glossary allows to capture multiple definitions of the terms used as constructs. The synonyms of the terms can be represented in the KB to enhance the precision and completeness of future query results.

To correctly link the terms identified in the hypotheses, factor treatments, and response variables with the proper theory constructs took considerable expertise regarding the interpretation of the experiment context. For instance, average effectiveness can mean individual effectiveness (C10) or inspection effectiveness (C02), depending on the context of an experiment design.

We tried to represent candidate theory propositions coming from experiment hypotheses applying the framework for describing SE theories presented in [3]. However, we faced practical difficulties. For instance, some constructs, such as effort and effectiveness, fit to several archetypes and could not be consistently placed into the proposed diagram. Both could be related to actor, technology, and software system. Concerning the archetypes, we noted that the activity archetype in our case, as in other theory representation experiences [3] [25], would be used only for scoping (to "defect detection") and not have any related constructs. Moreover, mapping several constructs lead to tangle of proposition arrows, not well supporting further analyses.

Therefore, we represented those theory proposition candidates using a directed graph containing potential cause-effect relationships between the constructs and different colors for the archetypes (applying the color that seems to fit best). The resulting diagram for the 12 constructs previously described can be seen in Figure 8. This figure misses some relevant identified constructs, which may have effects on the ones presented (e.g., team size, inspector capability, and defect classes). The complete current theory construct graph for all the 23 identified constructs is available online[5].

This graph shows, for instance, that the software artifact type (C14 – SAT) and its size (C15 – SAS) influence the defect detection technique (C11 – IDDT), which in turn has an effect on the individual effectiveness (C10 – IEffe), as the total defect number (C16 – TDN) may also have. Thus, each of the arrows represents candidate propositions with specific construct values and effects to be evaluated by empirical studies. These propositions were added mainly using researcher background and expertise. Note that queries Propositions/Hyp* can be used to search for empirically supported propostions on a given construct and its synonyms. The available empirical evidence is not represented in the cause-effect graph and can be stated as future work.

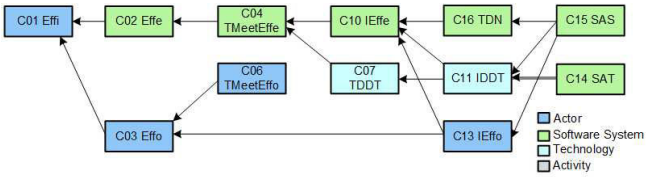



**Figure 8. PBR theory constructs and candidate propositions.**

The query Explanations retrieves the main findings of the papers, which might provide useful insights into explaining the empirically-based theory propositions. For instance, in the replicated experiment focusing on individual reviewer effectiveness described in [26], the investigated hypotheses obtained from query PropositionsHyp include: "Individuals applying each PBR perspective perform better than individuals applying Ad-hoc reading with respect to their mean defect detection rate" which can be seen as "Partly Confirmed". The finding, obtained from query Explanations which explains this observation is: "The results showed that PBR was more effective than Checklist for one of the two requirements documents used".

Finally, query Scope showed that the scope of the empirically-based theory is mainly related to defect detection activities in in-vitro experiments conducted offline with toy problems containing seeded defects (typically about 20 to 30) in simple and small artifacts inspected by students. This might represent a problem, since in practice artifacts tend to be larger and more complex.

# 5. EVALUATION AND DISCUSSION

This section reports on the evaluation of the proposed theory identification approach and its support based on our PBR evaluation use case. The effort spent on each research issue (steps) for

[6] http://glossary-sis.herokuapp.com/tags/Theory_Construct_SI

PBR theory identification is shown in Table 1. It is important to state the RI-1 (KB data modeling and query design) and RI-2 (KB population) are related to preparation, while RI-3 is related to analyzing empirical evidence for TI.

**Table 1. Effort for PBR theory identification.**

| | |
|---|---|
| **Data Model & Queries (RI-1)** | Data Model: 0 ph (reused).<br>Query Definition: 4 ph (reusable). |
| **KB Population (RI-2)** | Protocol: 0 ph (reused).<br>Data extraction: 2.0 ph per experiment (reusable).<br>Data validation: 0.5 ph per experiment (reusable). |
| **Identifying Theory (RI-3)** | Theory elements analysis based on queries: 6 ph.<br>Dependency graph building: 3 ph. |

It was possible to completely reuse the data model from previous work [8], since the same research area of software inspections was addressed. For experimental data on other research areas, the data model shown in Figure 4, based on the empirical study context and experimental concepts could also be reused. However, the tailoring of the research topics into specific parameters (or combinations of parameters) would be necessary (see Figure 5). Nevertheless, we believe that such tailoring (identifying appropriate parameters) requires only reasonable effort.

It can be seen that main effort was related to data extraction and validation (in total 35 person hours for 14 experiments). However, based on our previous SLR experiences, such as [14], we found the data extraction effort comparable to the data extraction effort in SLRs. It is also noteworthy that the extracted data can be reused beyond the scope of a local workgroup and that new data can be extracted and incrementally integrated into the KB.

The pre-project and KE setup effort are shown in Table 2. The pre-project effort is related to building the pre-existing SKE tool support and the newly developed and reusable glossary facility. The KE setup effort relates to the activities of the knowledge engineer to create the KB ontology, implement the queries, and using the data contribution interface to populate the KB.

**Table 2. Pre-project and KE setup effort.**

| | |
|---|---|
| **Pre-Project Tool Support** | SKE tool support dev.: 80 ph (reused).<br>Glossary implementation: 60 ph (reusable). |
| **KE setup effort** | Creating the KB ontology: 16 ph (reused).<br>Query implementation: 32 ph (reusable).<br>Data import (automated): < 0.5 ph. |

SKE's data contribution interface was effective in enabling contributions from researchers and efficient by requiring low effort and little time for such contributions to be imported. Considering all the new data model entities such as, experiment (9 entries), hypotheses (54 entries), factors (11 entries), response variables (79 entries), experiment runs (13 entries), treatments (72 entries) and measurements (392 entries), among others (see Figures 4 and 5), more than 2,500 data elements (cells) were effectively imported into the KB within less than one minute.

Considerations on the effectiveness, from the point of view of the researchers, of the solutions provided to address the research issues in the PBR theory identification use case are summarized in Table 3.

On the effectiveness of the queries (defined in RI-1), we highlight the semantic technology, which allows querying on domain concepts, synonyms and related concepts (gathered during data extraction or added to the KB afterwards). For instance, with synonym search enabled, query Propositions/HypRV, when searching for hypotheses with response variable similar to domain concept "efficiency", is able to retrieve hypotheses with response variables as "number of defects per hour" or "number of faults per hour", since those variables represent the same concept.

**Table 3. Effectiveness of solutions provided for research issues in the PBR theory identification use case.**

| | |
|---|---|
| **Data Model & Queries (RI-1)** | Data Model: Effective. Allowed characterizing inspection experiments (model of similarities and variations) and their results. Supported both, data extraction and querying.<br>Queries: Effective. Provided correct results against test cases elaborated based on the 14 PBR experiments in the KB. The semantic technology allows querying on domain concepts, synonyms and related concepts. |
| **KB Population (RI-2)** | Effective. We were able to identify relevant experiments, extract data from them and integrate it into the KB. |
| **Identifying Theory (RI-3)** | Effective. Theory elements could be identified based on the query results and theory constructs were registered into the glossary. |

Concerning the effectiveness of the planned search protocol (related to RI-2), it is important to mention that it did not intend to get the complete set of inspection experiments but a fair and objective sampeand that the SKE KB can always be extended by adding data from more experimental studies. Therefore, as expected, when scoping to PBR, the 14 identified experiments did not match the 12 analyzed by Ciolkowski [23]. In fact, the merged set contains 20 experiments. Figure 9 shows the citation graph of these 20 PBR experiments. The 6 missing ones are shown in grey. Concerning the missing ones: 4 (20%) were not indexed in Scopus (one dissertation, one technical report, and two conference papers) and 2 (10%) did not match the search string in the title, abstract, and keywords because of using different synonyms. The latter reflects the limitation of syntactical search capabilities in digital libraries.

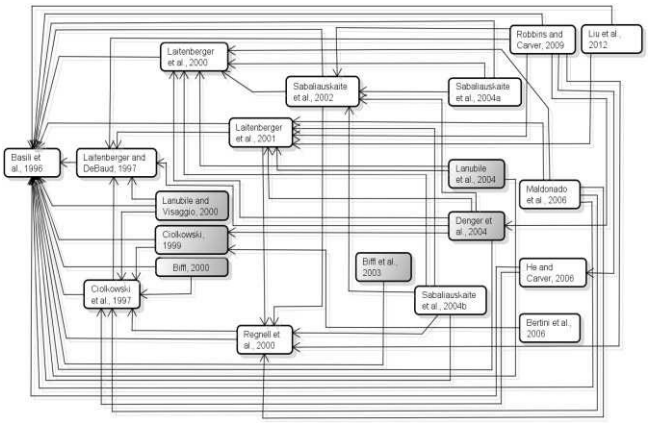



**Figure 9. PBR experiment paper citation graph.**

Finally, we believe that for the analysis activity for theory identification from empirical evidence (RI-3) the KB+glossary support is more effective and efficient when compared to both, doing so directly from digital libraries or based on spreadsheets containing extracted empirical evidence. Concerning effectiveness, with the

support from the query results it was possible to identify 23 theory constructs and to represent candidate propositions graphically. The glossary allowed registering definitions for the identified constructs. Moreover, information on theory elements explanations and scope could also be retrieved.

Compared to analyzing empirical evidence for theory identification directly from digital libraries, the support offers significant improvement by providing queries with structured access to concepts, semantic search capabilities, and retrieving candidate theory elements instead of complete research papers. The total effort for identifying theory elements by reading the 14 papers would certainly be higher than 6 hours (informally comparable to the time for reading up to three papers). Compared to analyzing empirical evidence directly from spreadsheets, this would also require more effort, since spreadsheets have limited support for complex querying and traversing all the data elements, manually identifying synonyms and filtering concepts would not be extremely difficult for large numbers of experiments. Based on the spreadsheets of our evaluation use case we perceive that doing the same task by directly and manually using them would certainly be error prone and take much more effort.

In the context of this comparison, it is noteworthy that by using the KB the extracted data can be reused and extended beyond the scope of a local workgroup, since new data can be extracted and incrementally integrated (eventually using semantic integration facilities in the case of heterogeneous data sources). As a result of our evaluation use case, for instance, the PBR KB and the glossary are now available online[5] for researchers to extend and to use for analyzing empirical evidence for identifying and discussing theory based on the query results and term definitions.

# 6. THREATS TO VALIDITY AND LESSONS LEARNED

The goal of this research was taking a step towards the identification of theory on a given research topic based on information from experiments, following the hypothetical path [6]. Also, we focused on identifying Level 1 theories and on how to make these theories explicit (although insights into more abstract Level 2 theories may have been provided).

**Threats to Validity.** A major threat to internal validity was having experts involved in analyzing the empirical evidence to identify theory based on the proposed support. This task involves interpretation and reasoning and therefore results may be influenced by expertise. Three of the authors of this paper are experts in software inspections. Therefore, they could have identified the constructs and drawn the theory graph directly from reading research papers, although this would require considerably more effort than the 6 person hours using the KB query results as input to analysis.

Concerning external validity, we identified two threats related to decisions taken in our evaluation use case: (i) the chosen research area of software inspections, and (ii) gathering knowledge from experiments. Software inspections are widely spread in academia and industry and many empirical studies have been conducted in this area, which may have facilitated the KB building and theory element extraction. Regarding the experiments, valuable theories may rise from results of other empirical strategies, such as case studies. We did not evaluate the feasibility of supporting theory identification based on such strategies.

**Lessons Learned.** Main lessons learned concern effort, and success and risk factors for effectively and efficiently supporting identifying SE theory from empirical studies. On the effort, we believe that the real bottleneck to uncover theories at the pace of new empirical studies being conducted is getting the community involved in contributing. Of course, starting to relate empirical SE research directly to theory and encouraging the use of concept-driven instead of substance-driven research also could help to swing the balance in the direction of SE theory.

A major success factor is properly involving a knowledge engineer for KB administration, data integration and providing query facilities. An identified risk factor, on the other hand, is not getting the research community involved for long-term collection and use of data.

# 7. CONCLUSIONS

SE research is full of implicit theory [7]. In this paper, we addressed challenges in supporting reverse engineering theory from published research, such as limitations of searching for theories in digital libraries and the absence of a platform where researchers can query for candidate theory elements to analyze and define terms related to theory constructs.

We focused on supporting recovering theory from published experiment research reports. Our strategy consisted of providing online support[5] in which the search for theory elements is supported by querying an extensible KB and the definition of theory constructs is supported by a glossary tool. For building the KB we proposed using the SKE process [8], which builds on the SLR process and on KE practices to provide a KB with semantic technology that enables querying for empirical evidence. A set of queries for identifying candidate theory elements was designed by analyzing a common data model for hosting experiments.

For evaluation, we applied our strategy to identify PBR-related theory elements. In this context, the proposed process and tool support was effective and efficient. SKE was used to extend a software inspection KB with knowledge acquired from PBR-related experiments. Based on the query results it was possible to identify 23 theory constructs and to represent candidate theory propositions as a dependency graph. The glossary supported defining theory constructs. The researchers found the provided query and glossary facilities usable and useful to support analyzing empirical evidence for theory element identification.

The support prototype is available online[5], enabling researchers beyond the scope of a local workgroup to discuss and evolve PBR-related theory based on the queries and on the glossary's theory construct definitions. This alternative process and support offers significant improvement over searching for theory elements in digital libraries (e.g., providing a constructs taxonomy, semantic querying with structured access to concepts, and retrieving candidate theory elements instead of papers) or in local spreadsheets with extracted data (e.g., providing a constructs taxonomy, semantic querying, and integration facilities for reuse across work groups). Therefore, we believe that this research can represent a step towards supporting reverse engineering SE theory from published research in a scientific community and that it should be investigated in a wider area of empirical research. Given the growing volume of empirical SE research, a critical factor is getting the community involved in this quest for uncovering decades of collected implicit theory, and in adopting concept-driven approaches for new empirical research.

Future work includes providing additional synonyms to the KB for the identified theory constructs, enabling new queries to retrieve empirical evidence on theory propositions relating two or more theory constructs (and their synonyms). We have received promising interest from empirical researchers and meta researchers to investigate the use of the proposed support in other SE research topics and on other empirical strategies. Finally, we plan to extend the tool support, setting up a platform to allow building knowledge and identifying theories based on the collective intelligence in empirical SE communities.

## 8. ACKNOWLEDGMENTS

This work was supported by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Economy, Family and Youth and the National Foundation for Research, Technology and Development - Austria, and CNPq - Brazil.

## 9. REFERENCES

[1] B. A. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in 26th International Conference on Software Engineering (ICSE 2004), 2004, pp. 273–281.

[2] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," J. Syst. Softw., vol. 80, no. 4, pp. 571–583, Apr. 2007.

[3] D. I. K. Sjøberg, T. Dybå, B. C. D. Anda, and J. E. Hannay, "Building Theories in Software Engineering," in in Guide to Advanced Empirical Software Engineering SE - 12, F. Shull, J. Singer, and D. K. Sjøberg, Eds. Springer London, 2008, pp. 312–336.

[4] B. Boehm and V. R. Basili, "Software Defect Reduction Top 10 List," IEEE Comput., vol. 34, no. 1, pp. 135–137, 2001.

[5] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Experimentation in Software Engineering," Springer, 2012.

[6] K.-J. Stol and B. Fitzgerald, "Uncovering theories in software engineering," 2013 2nd SEMAT Work. a Gen. Theory Softw. Eng., pp. 5–14, May 2013.

[7] P. Johnson, M. Ekstedt, and I. Jacobson, "Where's the Theory for Software Engineering?," IEEE Softw., vol. 29, no. 5, pp. 96–96, Sep. 2012.

[8] S. Biffl, M. Kalinowski, F. J. Ekaputra, E. Serral, and D. Winkler, "Building Empirical Software Engineering Bodies of Knowledge with Systematic Knowledge Engineering," Tech. Rep. IFS-CDL 13-03, Vienna University of Technology, October 2013. Available at http://qse.ifs.tuwien.ac.at/publication/IFS-CDL-13-03.pdf

[9] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Review in Software Engineering," EBSE Tech. Rep., EBSE-2007-01, 2007.

[10] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," Data Knowl. Eng., vol. 25, no. 1–2, pp. 161–197, Mar. 1998.

[11] R. Yin, Case Study Research: Design and Methods. Sage, Beverly Hills, CA., 1984.

[12] B. Glaser and A. Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Transaction, 1967.

[13] M. Pai, M. McCulloch, J. D. Gorman, N. Pai, W. Enanoria, G. Kennedy, P. Tharyan, and J. M. Colford, "Systematic reviews and meta-analyses: an illustrated, step-by-step guide.," Natl. Med. J. India, vol. 17, no. 2, pp. 86–95, 2004.

[14] M. Kalinowski, D. N. Card, and G. H. Travassos, "Evidence-Based Guidelines to Defect Causal Analysis," IEEE Softw., vol. 29, no. 4, pp. 16–18, Jul. 2012.

[15] J. Ye, L. Coyle, S. Dobson, and P. Nixon, "Ontology-based models in pervasive computing systems," Knowl. Eng. Rev., vol. 22, no. 04, pp. 315–347, Dec. 2007.

[16] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," Int. J. Hum. Comput. Stud., vol. 43, no. 5–6, pp. 907–928, Nov. 1995.

[17] N. F. Noy, "Semantic integration," ACM SIGMOD Rec., vol. 33, no. 4, p. 65, Dec. 2004.

[18] F. J. Ekaputra, E. Serral, D. Winkler, and S. Biffl, "An Analysis Framework for Ontology Querying Tools," in 9th International Conference on Semantic Systems, 2013, pp. 1–8.

[19] M. E. Fagan, "Design and code inspections to reduce errors in program development," IBM Syst. J., vol. 15, no. 3, pp. 182–211, 1976.

[20] F. Shull, I. Rus, and V. Basili, "How perspective-based reading can improve requirements inspections," IEEE Comput., vol. 33, no. 7, pp. 73–79, Jul. 2000.

[21] T. Thelin and P. Runeson, "Usage-based reading - an experiment to guide reviewers with use cases," Inf. and Soft. Tech., vol. 43, no. 15, pp. 925-938, 2001.

[22] F. Shull, I. Rus, and V. Basili, "Improving Software Inspections by Using Reading Techniques," in 23rd International Conference on Software Engineering (ICSE 2001), 2001, pp. 726–727.

[23] M. Ciolkowski, "What do we know about perspective-based reading? An approach for quantitative aggregation in software engineering," 3rd Int. Symp. Empir. Softw. Eng. Meas., pp. 133–144, Oct. 2009.

[24] O. Laitenberger and J.-M. DeBaud, "An encompassing life cycle centric survey of software inspection," J. Syst. Softw., vol. 50, no. 1, pp. 5–31, Jan. 2000.

[25] P. M. Dos Santos and G. Travassos, "On the Representation and Aggregation of Evidence in Software Engineering: A Theory and Belief-based Perspective," Electron. Notes in Theor. Comp. Sc., pp. 1–27, 2013.

[26] J. C. Maldonado, J. Carver, F. Shull, S. Fabbri, E. Dória, L. Martimiano, M. Mendonça, and V. Basili, "Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness," Empir. Softw. Eng., vol. 11, no. 1, pp. 119–142, Feb. 2006.