Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 292 (2013) 95–118

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

On the Representation and Aggregation of Evidence in Software Engineering: A Theory and Belief-based Perspective

Paulo Sérgio Medeiros dos Santos and Guilherme Horta Travassos[1]

Systems Engineering and Computer Science Department
PESC/COPPE/UFRJ
Rio de Janeiro, Brazil

Abstract

An adequate representation and a feasible aggregation procedure of evidence represents a challenging problem in many disciplines. The right representation can help scientists discuss and present the results of their findings and, if it is simple enough, it can be useful for practitioners to base their decisions on improvement implementations. The aggregation strengthens confidence in comparison to single evidence and is an important contribution to the body of knowledge. In this paper, we present a preliminary proposal to use empirically-based theories and belief functions as a means to represent and aggregate evidence. By having evidence explained by the same theory, we used belief functions to combine them in a way that the theory propositions (cause-effect values) result from combined evidence. We suggest this can be an useful way to obtain a good estimate of multiple evidence combination. In addition, we indicate its possible usefulness for practitioners to formalize and reuse their experiences. A real-case application of the approach is presented by formulating a theory for Usage-Based Reading inspection technique and aggregating the evidence acquired in three related empirical studies. This application indicated that the approach can give compatible results with the aggregated evidence.

Keywords: software engineering, theory, belief functions, dempster-shafer theory, evidence representation, evidence aggregation, post-mortem, research synthesis.

1 Introduction

Even with more than 10 million software engineers around the world and the intense research activities involving the discipline Software Engineering, there are still limited means professionals and researchers can use to share and aggregate the benefits, drawbacks, or limits of software technologies used in everyday software development activities.

[1] Email: pasemes@cos.ufrj.br, ght@cos.ufrj.br

doi:10.1016/j.entcs.2013.02.008

In general, practitioners tend to use informal and social communication channels to share their lessons learned and best practices with peers, including blogs, question-answer or informative wikis and technical conferences. In these mediums, the preferred format for presenting experiences is to use a narrative mode with support of stories and metaphors to persuade the audience [1]. Anecdotes and small demonstrations are commonly used too, and the so-called 'experts' are respected and have more influence than other sources of information [2,3].

Project post-mortems represent another significant source of lessons experienced in software development. Usually performed ad-hoc by a team or group of developers, they generate results which are commonly textually reported intending to detail what went well and what went wrong in the project course [4]. In addition to textual reports, graphical representations, such as Root Cause Analysis with Ishikawa diagrams [5] and Cognitive Maps [6], have also been used as a practical way to codify and reuse the experiences generated and reported in software projects' post-mortems analysis. Besides its application in post-mortems, the visual format is largely used by practitioners to share experiences through diagrams and raw sketches. And together with the narrative format, they are useful to describe situations involving uncertainty which is commonly found in the practitioners' environment [1].

As it might be expected, rigour is not usually present in the majority of these informal representations. On the other hand, relevance is usually high as they directly relate to practice [2]. However, although all these channels and formats used by professionals can be effective to disseminate experiences, its organization into a practical body of knowledge remains loose (there is plenty of information available in many places) and difficult to track (information available in many places is continuously evolving).

Therefore, accumulating and coping with the rigour of produced evidence challenges both researchers and practitioners. The academic community seems to be the most evolved in addressing these issues. Eight years have gone by since the introduction of Evidence-Based Software Engineering and more than 122 secondary studies have been conducted [7]. The main goal of Evidence-Based Software Engineering is "to provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision-making process regarding the development and maintenance of software" [8].

Still, even with the efforts in reporting the 'best evidence', many of the reports being produced by the Academia have difficulty in presenting their findings to practitioners in an accessible way by predominantly offering highly technical expositions [9] with an intensive use of the propositional format. According to [1], there are three knowledge representational modes: (i) the propositional one, which is the Academia's preferred mode, and (ii) narrative, and (iii) visual, which are the practitioners' preferred modes. The propositional format use in the Academia is justifiable if we think about the science pursuit to represent its reasoning in the most objective way to legitimate its results. Thus, by using propositional statements in its discourse it tends to increase the testability of the investigation claims validity. However, in many cases this results in the incidental detriment of the pragmatic

utility to industry [9]. Nevertheless, proposals for graphically representing evidence synthesis, such as by using Cognitive Maps in conducting thematic synthesis [10] and graphs for systematic reviews [11], do exist and point out the importance of finding models that can facilitate not only understanding the evidence but also their aggregation.

Thus, the mismatch in the orientation of how practitioners and researchers perceive evidence usefulness in terms of its representation and properties constitutes a research challenge. The question is how can evidence be represented and disseminated to both Academia and Industry in a way that it could be at the same time applied to practice and accumulated within the Software Engineering body of knowledge [12]. In other words, it is necessary to represent industrial experiences and research evidence in the same perspective so that all stakeholders can benefit from it, regardless of its use being to support scientific inquiries or pragmatic decision-making in software projects. The first step in moving in this direction is to expand the concept of evidence, as it is commonly conceived by the Academia, and start to accept all kinds of evidence from the weak and incomplete (*e.g.*, lessons learned and experiences from practice) to the rigorous and well-documented (*i.e.*, originated in experimental studies) [9]. By doing so, software engineers will have the opportunity to make use of evidence even when it is not so 'perfect', for instance if there are no statistically significant conclusions associated with it. But at the same time they can have the chance to aggregate them and obtain a 'consensual opinion' on the use of software technologies so that insights can be gained from the current body of 'imperfect' evidence [9]. However, to make such an aggregation possible, the evidence must be empirically-based, that is, true evidence about results from application in practice, not arguments based neither on expectation nor pure speculation.

An interesting way to capture and represent evidence is through theories. In this paper, we will try to leverage theories as a framework to devise a mechanism to represent and aggregate evidence. In most scientific disciplines, theories represent a solid ground upon which scientific knowledge is formulated and accumulated. Theories support scientists in providing an orderly depiction of some phenomenon in the real world in a manner that some complexity of the real phenomenon is reduced in the representation [13]. As a result, theories can facilitate the communication of ideas and knowledge by offering a common conceptual framework for structuring knowledge in a concise and precise manner [14].

Based on these properties, we supported our proposal on the minimal set of characteristics defining a theory as stated by Bacharach [13]: (1) a system of constructs related to each other by propositions and (2) a boundary defining its applicability. Joining efforts of researchers concerned with the application of theories in the field [15,16], we have used the Sjøberg et al. [15] theory conceptualization that allowed both representing the theory's propositions and its boundary using a well-defined visual notation. The notation was formulated to be used with empirically-based theories. The visual format has a good fit with the concept of theories as the format is best employed when the objective is the simplification or aggregation of complex

information into meaningful patterns using a well-defined set of rules [1]. Thus, it can facilitate the use of knowledge for both Industry and Academia by improving the readability and understandability for practitioners as well as assigning it the necessary rigour required by researchers.

Our approach takes empirical evidence explained by the same theory and combines their effects using belief functions to form the combined theory's propositions, so that we can have an estimative of the effect of all evidence taken together. By employing belief functions (also known as Dempster-Shafer Theory [17]) we evaluate two dimensions for each proposition: the proposition value (*e.g.*, negative/positive effect) and the confidence on its value. In essence, this could be similar to the 'classical' or quantitative meta-analysis which uses an effect-size metric and a criterion to 'weight' it, to produce a more precise effect size as opposed to a result derived from a single study [18].

The remainder of this paper is organized as follows. Section 2 details the theory conceptualization as well as the Dempster-Shafer Theory used as the aggregation mechanism. Section 3 presents the approach. Section 4 discusses its benefits and limitations. Section 6 comments on the industry perspective and Section 7 points to future works.

2 Background

2.1 Theories

In accordance with the general understanding of theories, Sjøberg et al. [15] suggest that the description of theories in Software Engineering should be split into four parts: *constructs* (the basic elements), *propositions* (how the constructs relate), *explanations* (why the propositions were specified) and *scope* (what is the universe of discourse applicable to the theory). We choose to use the Sjøberg et al. theory conceptualization as it is already tailored to Software Engineering and defines a visual representation with specific notational semantics.

Figure 1 shows the graphical theory schema using the notation given by [15]. This theory was extracted from a real world action research study on the use of source code refactoring in a medium-to-large scale Web software project [19]. The notational semantics is partly based on UML. A *construct* is represented as a class or class attribute. A class is represented by a box with its name written at the top, such as, for instance, 'Distributed Project'. A class can have a subclass (using the same generalization notation as in UML) or a component class (drawn as a box inside another box such as, for instance, 'Source Code'). Usually, if the *construct* represents a particular variable value, then the *construct* is modelled as a subclass or component class (*e.g.*, 'Large Scale Web Systems'). Otherwise, if the focus concerns the values variations, then the *construct* is a variable modelled as a class attribute, such as 'Effort'. An attribute is placed in the class box bottom (below the horizontal line). Usually, classes will represent the study's independent variables (value constructs) and attributes the dependent ones (variable constructs).

A *proposition* relationship is modelled as an arrow. An arrow from A to B means

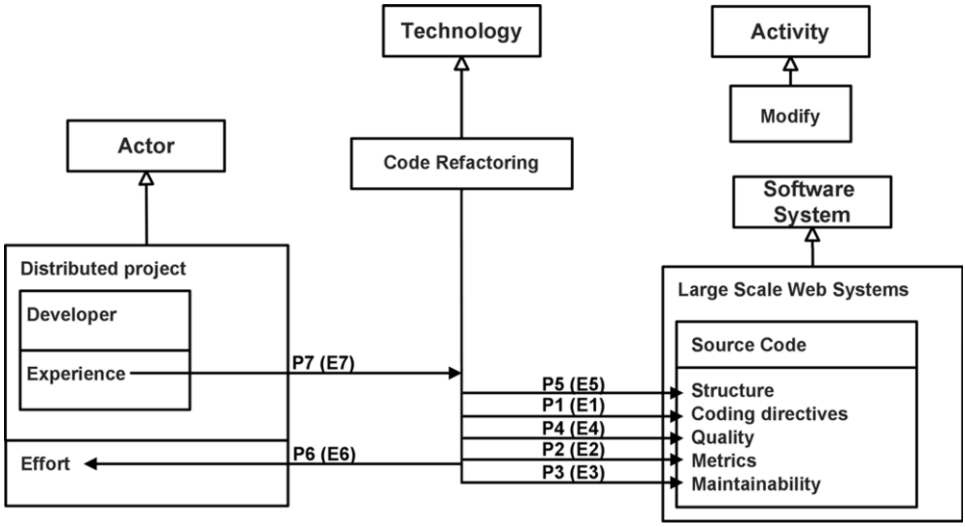

Fig. 1. Refactoring the theory diagram from [19]

that A affects B, where A is a class or an attribute and B is an attribute. In addition, B can also be a relationship itself. In this case, A is called a moderator, as in the case of the 'Experience' construct. It means A affects the direction and/or intensity of the B relationship effect. The moderators are also defined as *propositions*.

In defining a typical SE scenario a theory should present four elements, called archetypes (actor, technology, activity, and software system), represented by the inheritance roots. Driving these archetypes selection is a SE practice generalization described as an *actor* applying a *technology* to perform *activities* in a *software system* [15]. The word technology is also used as synonym for method and methodology. In addition, notice that the *technology* archetype is a central element in the theory, as it defines the hypothesis 'by applying technology X these effects are expected'. This will be used as aggregation criterion later on.

To supplement the graphical representation some additional details are needed. It is necessary to define each *construct* and describe each *proposition* with its associated *explanation* of why it holds. The P and E labels attached to the relationships in Figure 1 were used to index these definitions on the original text. To illustrate it, Table 1 describes some of these definitions. For a full description see [19]. It is important to notice that propositions values are specified in qualitative terms (*e.g.*, *Code Refactoring positively affects Maintainability*). And lastly, the theory scope is outlined which typically consists of theory subclass and component class elements (*i.e.*, value constructs).

2.2 Dempster-Shafer Theory

The Dempster-Shafer Theory (DST) is a theory of uncertain reasoning designed to deal with the distinction between uncertainty and ignorance. The main motivation for its creation was the desire to liberate the probability theory from the need to attach a measure of uncertainty to every hypothesis under consideration [17]. There

Table 1
Some theory elements' description from [19]

Constructs	
C1	*Code Refactoring* (development practice the act of modifying software structure without changing its observable behaviour)
C2	*Source Code Structure* (structural properties perceptible in the source code, ex.: readability, algorithm structure)
Propositions	
P5	Code refactoring positively influences code structure
Explanations	
E5	Source code structure improves: • It becomes more homogeneous throughout the entire software project, according to the previous knowledge of the developers. • Its size and complexity is reduced.

are three main concepts in DST: the *frame of discernment*, the *basic probability assignment* function (*bpa* or m) and the *belief* function (*Bel*). This section summarizes these main concepts; however the reading of [20] is suggested for a complete introduction.

In the DST, the set of hypotheses is called *frame of discernment*, usually denoted as Θ. Any subset A of Θ is also considered a hypothesis representing an important difference from the classical probability theory. Beliefs can be assigned to all possible subsets of Θ, denoted as 2^{Θ}. The core of any DST model is the *bpa* function, which represents the impact of each distinct evidence on the subsets of Θ. To do that, the *bpa* function assigns a number in [0, 1] to every subset of Θ given by m: $2^{\Theta} \mapsto$ [0,1], where the only restrictions on $m(\cdot)$ below:

$$\sum_{x \in 2^{\Theta}} m(x) = 1 \text{ and } m(\emptyset) = 0. \tag{1}$$

so that all assigned probabilities sum to unity and there is no belief in the empty set.

If there are two or more independent *bpa* functions over the same *frame of discernment*, say functions m_1 and m_2, the Dempster's rule of combination can be used to aggregate the information contained in m_1 and m_2 into a combined *bpa*

function $m1 \oplus m2$. The combination rule is defined as:

$$m_3(C) = \frac{\sum_{\substack{i,j \\ A_i \cap B_j = C}} m_1(A_i) m_2(B_j)}{1 - K}, \text{ where} \tag{2}$$

$$K = \sum_{\substack{i,j \\ A_i \cap B_j = \emptyset}} m_1(A_i) m_2(B_j)$$

To calculate the function m_3 (*i.e.*, $m1 \oplus m2$) value for the C set, the equation above sums all products of the form $m_1(A_i)m_2(B_j)$, where A intersection B yields the C set. K represents the basic probability mass associated with conflict, and it is given by all belief allocated to the empty set – $m(\emptyset)$. Conflicts usually arise from combining *bpa* functions which represent (partly) contradictory evidence. They can be eliminated by redistributing the value of $m(\emptyset)$ among the other *frame of discernment* subsets. That is what the 1-K denominator does in the above equation 2. An example of using all these concepts will be given in next section, but for a preview of how it works see Table 2.

The last important concept of DST is the *belief* function. *Belief* functions are what allow decisions to be made based on combined evidence – usually the singleton hypothesis (*i.e.*, the subset with one element) with the highest associated belief. A *belief* function, corresponding to a specific *bpa* function (*e.g.*, m_1 or m_2), assigns to every subset A of Θ the sum of the beliefs committed exactly to each subset of A defined by m. That is:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{3}$$

There are several criteria on how decision-making can be made with *belief* functions. One commonly used is to choose the hypothesis (*i.e.*, the subset) with the highest associated belief. Usually, this is done only with the singleton hypotheses, which are the subsets with one element. In the next section we present a different criterion to support decision-making to better frame our problem.

3 Aggregating Evidence Using Dempster-Shafer and Theories

The goal is to aggregate related evidence and see what they 'say' together as regards one specific technology. This can be roughly compared with the quantitative meta-analysis requirement on dealing with evidence generated in studies that have the same hypothesis [18].

Assuming the process of the building theory is correct and *constructs* are well defined, each new evidence will instantiate a new theory if it does not fit any known

theory, otherwise it will be aggregated into an existing one. As evidence is generally associated to a particular context, the resulting aggregated theory will initially have a small level of empirical support. The existence of evidence supporting a theory is what can support its definition as hypothesis or, at the other end of the spectrum, a law [14]. Figure 2 shows the process of aggregating evidence.

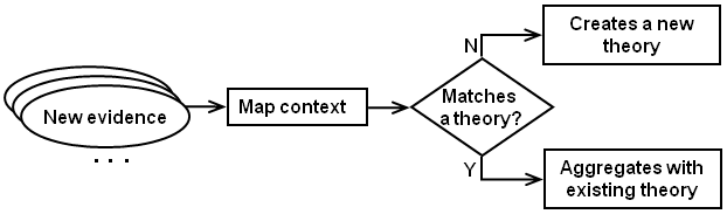

Fig. 2. Simplified view of the evidence aggregation process

In order to combine evidence, it is necessary to define what makes it match a known theory or not. It is possible to do that by mapping the context to the theories' *value constructs*. Since the goal is to evaluate software technologies, the first element that must match is the technology *construct* (the one that inherits from the Technology archetype). If there is a theory for such technology, then other *value constructs* (representing the theory *scope*) have to match the conditions where the evidence was observed (*i.e.*, its context). For example, if evidence is observed in a large-scale Web system project, then the theory should have an equivalent *construct* mapping this context. As it is possible to see, the fit operation is dependent on well-defined *constructs* so that different evidence observed in similar contexts can be related to the same theory. Taxonomies can be of great help in this regard.

After evaluating if the evidence fits an existing theory, it is necessary to describe the software technology impacts on the observed environment. This is done by using *propositions* and *variable constructs*. At this stage, almost all theory elements are defined and, if the evidence context does not match any known theory, a new theory can be created (see Figure 2). Otherwise, the evidence *propositions* are combined into an existing theory. This is the moment where DST is used.

When applying DST to represent evidence, as *value constructs* are predetermined by the context, only the *propositions* and corresponding *variable constructs* need to be considered. The left term of the combination in Figure 3 contains the evidence representation related to the theory presented in Figure 1. It defines a *bpa* function for each *variable construct* (*e.g.*, $m_{1-effort}$), specifies the values for each *proposition* using the *bpa* function (*e.g.*, {WN} for a weakly negative effect proposition) and assigns the belief associated with the evidence (*e.g.*, 0.65). The expression $m_{1-effort}(\{\text{WN}\}) = 0.65$ should be read as 'code refactoring weakly affects negatively effort with high certainty (0.65)'. Negative means prejudice, in this case. The belief value of 0.65 was an interpretation of our study [19].

It is important to notice that we have assigned the same belief (0.65) to all theory *propositions*, but this is not mandatory. In fact, in we have used qualitative (Grounded Theory) and quantitative (Categorical Regression) data analysis. The data type differentiation (*i.e.*, qualitative and quantitative) and the analysis

results (*e.g.*, *p-value* for quantitative analysis) could have been used to individually determine the belief in each *proposition*. However, we currently do not have a systematic procedure to determine that. As we have not yet fully addressed how the belief is assigned considering these aspects, we have assigned the belief of 0.65 for all *propositions* taking into account the fact that the evidence was originated in a real-world environment through an Action Research study. This is just an estimate to illustrate this aggregation approach.

The *frame of discernment* for the *proposition* values consists of the set of qualitative options that can be used to qualify a *proposition*. Based on the *Likert* scale, we defined seven options: strongly negative (SN), negative (NE), weakly negative (WN), indifferent (IF), weakly positive (WP), positive (PO) and strongly positive (SP). Thus, the *frame of discernment* is $\Theta = \{\text{SN, NE, WN, IF, WP, PO, SP}\}$. For the moderator *propositions* we define three options: inversely proportional (IP), neutral (NU) and directly proportional (DP). So, if a *construct* directly moderates the effect of a software technology, it means that the higher the value for a construct the greater the moderation effect is. The *frame of discernment* for the moderator *propositions* is $\Theta = \{\text{IP, NU, DP}\}$.

When belief is assigned to a set with more than one element (*e.g.*, $m_{1-metrics}(\text{WP, PO}) = 0.65$), it should be interpreted as a range of values (*i.e.*, between weakly positive and positive). In addition, notice the importance of the 'indifferent' value. Suppose we have a situation where $Bel_{-quality}(\{\text{IF}\}) = 0.9$. As there is in this case a high level of certainty that the quality *construct* is indifferent for the theory, it could be used as a criterion to remove this *construct* (and the *proposition*) from it. On the other hand, this gives us freedom to add new *variable constructs* at any time. If new evidence suggests the relevance of a variable construct which was not being considered in the theory, then we just add it to the theory and create a new *bpa* function for it.

Having defined how evidence can be represented using DST, the combination of two evidence related to the same theory can be done by applying the Dempster's combination rule (*i.e.*, combining all *bpa* functions defined for each of the theory's *propositions*). As an example, suppose the two pieces of evidence in Figure 3 have to be combined.

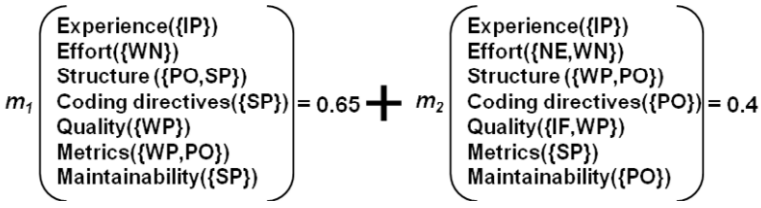

Fig. 3. Combining evidence related to the same theory

Table 2 shows the combination of two bpa functions related to the structure proposition. This is done for each proposition individually by using the combination rule. The tabular format is used just for illustrative purposes.

From Table 2, we have the combined *bpa* function values for the structure *propo-*

Table 2
Combination of two bpa functions (structure proposition)

$m_{1-structure}$ \ $m_{2-structure}$	{WP,PO} (0.4)	Θ (0.6)
{PO,SP} (0.65)	{PO} (0.26)	{PO,SP} (0.39)
Θ (0.35)	{WP,PO} (0.14)	Θ (0.21)

sition:

$m_1 \oplus m_2$ ({PO}) = 0.26,
$m_1 \oplus m_2$ (PO,SP}) = 0.39,
$m_1 \oplus m_2$ ({WP,PO}) = 0.14,
$m_1 \oplus m_2$ (Θ) = 0.21,
$m_1 \oplus m_2$ is 0 for all other subsets of Θ.

From the combined *bpa* functions it is possible to obtain the respective *belief* functions. The *belief* function values for the structure *proposition* associated with the combined *bpa* function m_3 (i.e., $m_1 \oplus m_2$) are:

$Bel_{3-structure}$({PO}) = 0.26,
$Bel_{3-structure}$(PO,SP}) = 0.39 + 0.26 = 0.65,
$Bel_{3-structure}$({WP,PO}) = 0.14 + 0.26 = 0.40,
$Bel_{3-structure}$(Θ) = 1.

Based on the *belief* functions' results we have to decide what the new *proposition* values will be for the theory representing the combined evidence. There is no universal rule for doing this as it depends on the problem being modelled, especially when there is semantics associated with the compound hypothesis (*i.e.*, a subset of Θ with two or more elements) in addition to the singleton hypothesis (*i.e.*, a subset of Θ with only one element). In our case, a two-element subset represents a scenario where we are not sure about a specific *proposition* value and specifies a range for it (*e.g.*, 'somewhere between positive and strongly positive'). Considering this, we have defined the following criteria. A compound hypothesis is chosen whenever it has the highest belief and if the singletons contributing to its belief do not contribute with more than 75% percent of its total belief. Otherwise, if a singleton hypothesis contributes with more than 75% to the belief associated with the compound hypothesis with the highest belief, it should be chosen. In our example, it has happened with the structure *proposition*. Subset {PO,SP} is the compound hypothesis with the highest associated belief and the singletons hypothesis contribution for it does not go beyond the 75% threshold ({PO} contributes with 40% – 0.26/0.65 = 0.4 –, and {SP} contributes with 0% – 0/0.65 = 0). Using these criteria we have the following *proposition* values:

IP for experience, since $Bel_{3-experience}$({IP}) = 0.79,
WN for effort, since $Bel_{3-effort}$({WN}) = 0.65,
PO-SP for structure, since $Bel_{3-structure}$({PO,SP}) = 0.65,
SP for coding direct, since $Bel_{3-codingdirectives}$({SP}) = 0.53,

WP for quality, since $Bel_{3-quality}(\{WP\}) = 0.65$,
WP-PO for metrics, since $Bel_{3-metrics}(\{WP,PO\}) = 0.72$,
PO for maintainability, since $Bel_{3-maintainability}(\{SP\}) = 0.53$.

This is the aggregate evidence considering the theory representation. In this short example, which is based on two evidence and associated belief with each one, the theory in 1 would be supported by the aforementioned proposition values. Besides the *proposition* value itself (e.g., 'weakly positive for quality') chosen with the defined criteria, another important information associated with each theory *proposition* is the belief values. The belief derived from the evidence combination reflect two aspects of the aggregation: (1) the degree of agreement among the aggregated evidence and (2) the strength of each evidence. Thus, a high belief as an outcome of an aggregation can be, for instance, a result of a large number of weak evidence reporting a similar observations or a small number of strong evidence also reporting similar phenomena. On the other hand, small belief can represent either insufficient number of weak evidence available or few conflicting strong evidence. In any case, what is interesting in using DST is that it not only indicates a trend (e.g., negative or positive) for each *proposition* based on aggregation, but also gives an intuitive parameter for interpreting how reliable the final (aggregated) *proposition* values are based on the derived *belief* values from the DST combination rule.

Finally, it is important to see that, although only the *proposition* value with the highest belief is taken for the combined theory, all *bpa* functions values resulting from the first combination are used for the next one. Take again, for example, the case of the *bpa* function associated with the structure *proposition*. If we have new evidence with $m_{4-structure}(\{SP\})$=0.9 , all m_3 values would be used in the combination as shown in Table 3. In this case, the new *proposition* value would be SP, since according to the $Bel_{4-structure}(\{SP\}) = 0.84$.

Table 3
The combination of a third evidence for the structure proposition

$m_{3-structure}$ \ $m_{4-structure}$	{SP} (0.9)	Θ (0.1)
{PO} (0.26)	∅ (0.23)	{PO} (0.03)
{PO,SP} (0.39)	{SP} (0.35)	{PO,SP} (0.04)
{WP,PO} (0.14)	∅ (0.13)	{WP,PO} (0.01)
Θ (0.21)	{SP} (0.19)	Θ (0.02)

The combined *bpa* function values for the structure proposition (note that, as defined by equation 2, all probability associated with the same subset is summed up – *e.g.*, {SP}):

$\kappa = 0.36$ and 1 - $\kappa = 0.64$,
$m_3 \oplus m_4$ ({PO}) = 0.03/0.64 = 0.05,
$m_3 \oplus m_4$ ({SP}) = (0.35 + 0.19)/0.64 = 0.84,
$m_3 \oplus m_4$ ({PO,SP}) = 0.04/0.64 = 0.06,

$m_3 \oplus m_4$ ({WP,PO}) = 0.01/0.64 = 0.02,
$m_3 \oplus m_4$ (Θ) = 0.02/0.64 = 0.03,
$m_3 \oplus m_4$ is 0 for all other subsets of Θ.

As we could see in this section, the evidence aggregation proposal is a viable mechanism for evidence synthesis. It is capable of capturing the reliability in each of the evidence considered and translating it into a synthesized result which includes the uncertainty involved.

4 Real Case Example

In this section we apply our approach to a real situation where evidence related to the same software technology and originated in similar contexts is aggregated. The topic chosen was purposely specific and narrow so that we could concentrate on the analysis of the approach.

Before we describe the evidence used for the example, we need to characterize the steps and procedures taken in aggregating evidence using our approach. Notice, however, that at this point in time only a high abstraction level process description is going to be explained. Additional research efforts are being applied to detail such process. Nevertheless, for the purposes of a proof of concept, the steps described next should be sufficient to understand the core concepts:

(i) **Definition**: define goals, pre-select an initical theory according to the goals (if one already exists) and determine the inclusion criteria for studies.

(ii) **Study select**: collect primary studies in a systematic way, according to the pre-selected theory and defined goals.

(iii) **Study quality assessment**: estimate confidence in the evidence (i.e., in the belief asserted to its propositions).

(iv) **Data extraction**: for each collected evidence sketch a theory following the steps described in [15], defining the important constructs, propositions, and proposition values.

(v) **Data synthesis**: guided by the procedure presented in Figure 2 combine the evidence. Next, evaluate and visualize the result of aggregation.

For this proof of concept, the main goal from the aggregation perspective is to understand the effect of the Usage-Based Reading (UBR) inspection technique. It is interesting to see that the concept of exploring multi-variables and their inter-relationships is very distinct from the traditional meta-analysis focus in one cause-effect relationship at a time. Due to this, we have no restriction to only select studies with the same hypothesis, except when different studies compare distinct software technologies or are conducted in a significantly disparate context using other dependent or independent variables. In fact, this is what Figure 2 tries to capture.

The evidence used came from a family of experiments on the UBR inspection technique [21], [22], [23] and [24]. Studies were performed to investigate UBR

performance in identifying faults on software artefacts. UBR is a reading technique whose primary goal is to drive the reviewers to focus on crucial parts of a software artefact from the user's point-of-view. In UBR, faults are not assumed to be of equal importance, and the technique aims at finding the faults that have the most negative impact on the users' perception of system quality. For this, reviewers are given use cases in a prioritized order and trace the use cases through the software artefact in that order to identify faults. A central element on focusing the inspection effort in UBR is the prioritization of use cases. UBR assumes that the set of use cases can be prioritized in a way reflecting the desired focusing criterion. If the inspection aims at finding the faults that are most critical to a certain system quality attribute, the use cases should be prioritized accordingly.

All the experiments used the same set of instruments. The subjects inspected a real-world high level design document which consisted of an overview of the software modules and communication signals that are sent to/received from the modules. The application domain regards a taxi management system and the design document specifies the three modules that composes the system, one taxi module for each vehicle, one central module for the operator and one communication link in-between these ones. All faults were classified into three classes depending on the fault importance from the user's point-of-view. Class A faults represent faults in system functions that are crucial for a user (*i.e.*, functions that are important for an user and that are often used). Class B faults represent those which affect important functions for an user (*i.e.*, functions that are either important and rarely used or not as important but often used). Class C faults are those which do not prevent the system from continuing to operate.

In total, four experimental studies were conducted. Two researchers participated in all studies. The first experiment [21] compared UBR against ad hoc inspection. A second experiment [22] investigated the amount of information in use cases that is required to make UBR useful. And the last two studies [23] and [24] compared UBR against a checklist based reading (CBR). Given the purpose of this example we have not used evidence from the second study, as it addressed a different research question and thus it would not be possible to aggregate its results with the other evidence.

To begin, we read and noted all the relevant information from the first paper comparing UBR against *ad hoc*. From our paper interpretation and the results it presented, we created the theory to represent this first evidence. The process of theory building, albeit systematic and with clear steps [15], is not exact in the sense that it depends on the interpretation and individual reasoning that is elaborating it. This way, the theory constructed represents our understanding of the evidence, but we are confident that it represents a widely accepted view as software inspection has been extensively studied by the empirical SE community and the authors already have conducted several studies related to this topic. Again, it was not by chance that this topic was chosen for this example.

The theory generated is shown in Figure 4. Following the protocol, all theory elements were described using a template as shown in Table 4. One major differ-

ence from the one presented in Figure 1 is that we are now comparing two software technologies of the same class (*i.e.*, software inspection) and not only describing the impacts of the use of the technology in itself. This new theory type will be named *comparative theory* and the type previously used (Figure 1) named *descriptive theory*. To represent this new theory type, the notation of [15] has been extended. The technology hierarchy was deepened to characterize UBR and *ad hoc* as software inspection techniques. In addition, all *propositions* were linked to both technologies symbolizing that they are being compared. That is why the box line pattern used for UBR and *ad hoc* constructs were replicated in the *proposition* lines, to denote which technique has better performance in comparison to each other. In this case, as it can be seen in Figure 4, *ad hoc* was better only in detecting a larger number of minor faults. Another significant difference in comparative theories is the set of *proposition* values. To qualify the proposition when comparing software technologies, the Likert scale can be used with the following options: strongly worse (SW), worse (WO), weakly worse (WW), indifferent (IF), weakly better (WB), better (BE) and strongly better (SB).

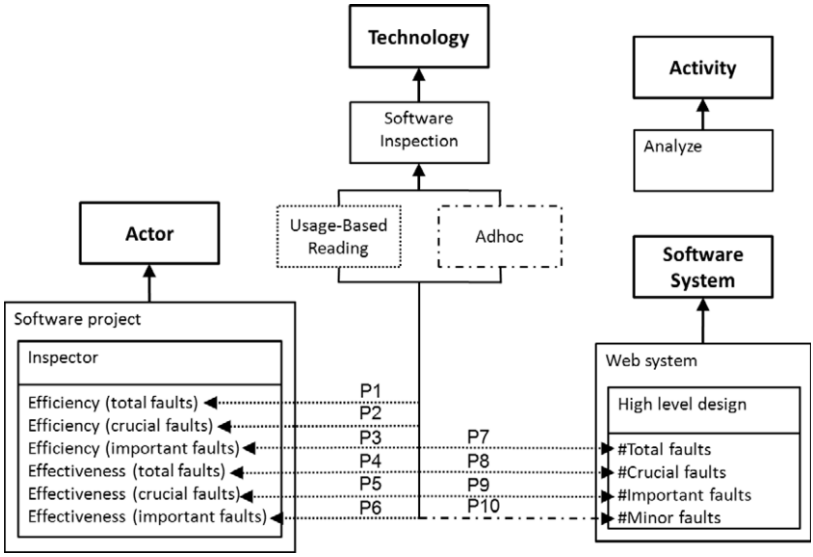

Fig. 4. Comparative theory for UBR and ad hoc inspection techniques

Almost all data and its analysis reported in the paper are quantitative. This facilitated the construction of the theory as the *variable constructs* are basically the dependent variables defined for the experiment. The *value constructs* were based on the instruments used in the experiment, the study design and the considered independent variables. Having defined the constructs, the next step was the definition of the *proposition* values. As the study data and results are in quantitative form and, in our approach, the proposition values have to be specified in a qualitative way, we had to translate from one form to another. The quantitative values are given in percentage as two technologies are being compared. The intervals (0%, 33%], (33%, 66%] and (66%, 100%] were used to, respectively, derive the qualitative values weakly better, better and strongly better *proposition* values. We attempted

Table 4
Some theory elements for the comparison of UBR and ad hoc inspection

Constructs	
C1	*Usage-Based Reading* (inspection technique focusing the reading effort in the most critical faults, from the user perspective, using a set of use cases as a guide to steer the inspection)
C3	*Effectiveness* (percentage of total number of faults found)
C10	*Web system* (system that uses the Internet infrastructure to operate)
Propositions	
P2	Usage-Based Reading performs strongly better than *ad hoc* inspection in relation to the efficiency associated with crucial faults.
P8	Usage-Based Reading performs strongly better than *ad hoc* inspection in relation to the identification of crucial faults.
Explanations	
E2	The inspector identifies more crucial faults by time units (hour) • More faults are found in the first part of an inspection and the longer an inspection last the less faults are found due to lack of concentration. • *p-value* = 0.0004 • The author does not explicitly present the efficiency, but as the experiment was time boxed (2.5h) and inspection time was almost the same for all inspectors, the efficiency is directly associated to the number of faults identified (see E8).
E8	The source code structure improves: • The result of the experiment shows it is possible to control reviewers in order to make them focus on important parts of a software artefact. • UBR detects 88% more crucial faults than ad hoc on average.

to keep this same criterion to determine all qualitative *proposition* values for all evidence from the three studies considered, but this was not possible all the time. Some results from the studies were not explicitly available in the technical papers and, in these cases, other means for obtaining the data were used whenever possible (e.g., calculated or indirectly obtained value or a visual medium such as figures and graphics). This was the case of proposition P2 in Table 4 – the efficiency for crucial faults was derived from the number of crucial faults. When the implicitly derived values were too imprecise in our interpretation we used qualitative interval such as WB-BE for the proposition values.

Having accrued all the necessary information about the first experiment, the

proposition values and the respective belief were determined. Figure 5 presents the result (note index 1 for the m function indicating it represents the first experiment). All *proposition* values are given for comparing UBR and *ad hoc*, in that order. Thus, for instance, #Minor faults ({WW}) should be interpreted as 'usage-based reading performs weakly worse than *ad hoc* in relation to the identification of minor faults'. It is possible to see that the same belief has been assigned to all theory *propositions* to keep this proof of concept simple in this example. As previously discussed this is not mandatory, and aspects such as the *p-value* of the statistical tests and the statistical effect size could have been considered in determining belief values for each one of the theory's *propositions*. Just as an example, the statistical test of the hypothesis related to the effectiveness considering the total number of faults gave a *p-value* of 0.0652 which was not considered statistically significant. Still the same belief of the other *propositions* was assigned to the *proposition* associated with the construct 'Effectiveness (total faults)'.

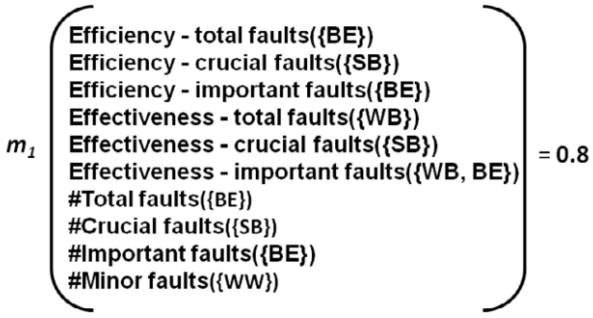

Fig. 5. DST representation of the comparative evidence from [21] – UBR x ad hoc

Yet, although the same belief has been assigned to all *propositions*, the belief value itself was not a mere interpretation of the authors as it was done in the example in the previous section but determined in a more objective way. This is our first attempt in structuring the estimation of the belief, but there is still plenty of room for refinement considering the aforementioned aspects. For the estimate we used two scoring schemas (or questionnaires) from the technical literature. The first, from [9], is a 20-point scale rating for the quality of evidence. In total, four questions are answered to determine the evidence's quality score: (1) how the technology was applied (0..7 points), considering the study type; (2) how the results were measured (0..5 points), considering if it is a subjective opinion or a rigorous comparison with another practice; (3) how the evidence was reported (1..5 points), considering the type of publication used to report the evidence; and (4) who reported the evidence (0..3 points), considering if the results were published by the same person who produced it or not. The second, from [25], is a screening questionnaire used to evaluate the quality of the papers found in a meta-ethnography study. The questionnaire has yes or no questions and focuses on more specific aspects of an empirical study such as the research design, the use of a control group and data analysis. In total, the questionnaire has eleven questions for 10 more points (the first question was discarded as it had already been covered in the other questionnaire).

Summing all the possible points the max achievable is 30 points. Based on that, the belief in the evidence was determined as the total number of points obtained by a paper divided by 30. For instance, the evidence in Figure 5 obtained 15 (3+5+4+3) points from the first questionnaire and 9 from the second, giving a total of 24 points that divided by 30 yields the 0.8 belief value.

With the evidence from the first study, the body of knowledge has now one comparative evidence (and the associated comparative theory). It can now answer questions related to the comparison of UBR and *ad hoc*, although it would be even more useful if it not only answered that kind of comparative question, but informed on about the impacts of each software technology alone. We devised a method to extract that kind of information from the comparative evidence, denoted as evidence dismembering operation (Figure 6). The method consists of taking the comparative evidence as a reference and generating two informative evidence where the difference between them is determined by the comparative evidence. To be consistent in dismembering the three comparative evidences from the three studies, the following criterion was adopted: for a comparative proposition qualified as a weak difference (WW or WB), the difference between the informative *proposition* values is half an unit in the Likert scale (*e.g.*, if one informative *proposition* value is WB the other must be the interval WP-PO). Analogously, if we have a significant difference (WO or BE), then the difference of the informative values is an integral unit of the Likert scale (*e.g.*, if one informative *proposition* value is PO the other must be SB). And, at last, if we have a strong difference (SW or SB), then the difference of the informative values is an integral and half unit of the Likert scale (*e.g.*, if one informative *proposition* value is WP-PO the other must be SB).

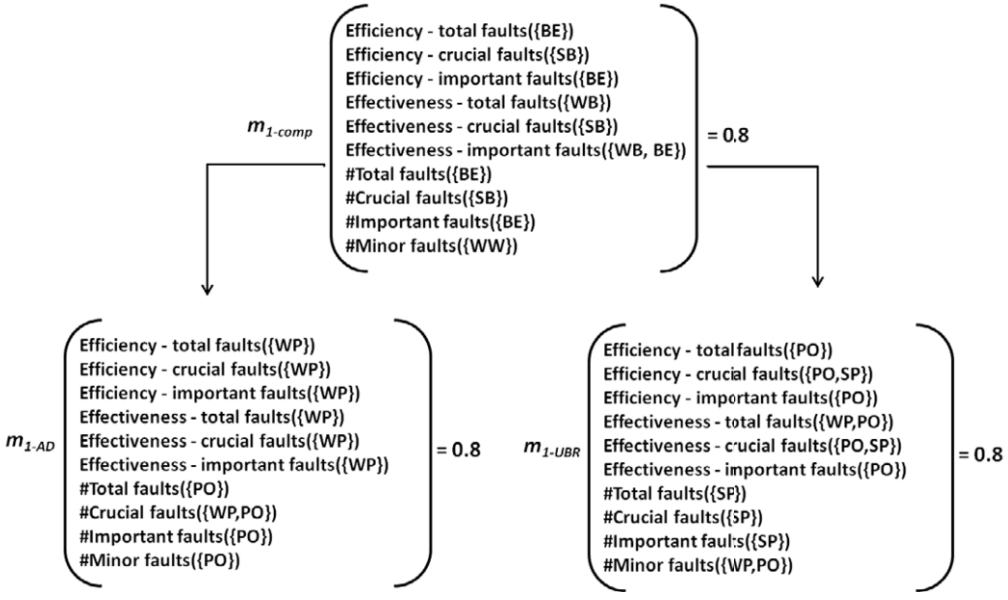

Fig. 6. Dismembering the comparative evidence from [21]

It is important to notice that comparative evidence only determines the difference between the two dismembered informative evidences, but the informative

proposition values themselves (*e.g.*, WP, PO or SP) are an interpretation from the study results. In addition, it should be stated clearly that we provide no *a priori* conclusive argument for the dismembering operation, only that it has a reasonable fit within the research reasoning process.

We repeated the described process so far for the two remaining studies comparing UBR and CBR. In all, considering the dismembering operation, nine evidence were generated. Table 5 enumerates them putting together the ones which are combinable. In brief, as the evidence came from a family of studies, all of them were combinable given that similar contexts were reproduced. The characteristics that prevented evidence from being combined were its type (comparative or informative) and the technology evaluated.

Table 5
Evidence generated from the analyzed studies

***m* function**	**Description**
m_{1-comp}	The comparative evidence (UBR x ad hoc) from study [21]. Not combinable with m_{3-comp} and m_{4-comp}, as different technologies were compared.
m_{3-comp} m_{4-comp}	The comparative evidence (UBR x CBR) from studies [23] and [24]. Combinable.
m_{1-ad}	The informative evidence for ad hoc inspection from [21].
m_{3-cbr} m_{4-cbr}	The informative evidence for CBR studies [23] and [24]. Combinable.
m_{1-ubr} m_{3-ubr} m_{4-ubr}	The informative evidence for UBR studies [21], [23] and [24]. Combinable.

To get an idea of what the outcome is in the combination of this actual evidence, we present the aggregation results for the UBR informative evidence. In addition to the m_{1-ubr} *proposition* values shown in Figure 6, Figure 7 displays the *proposition* values and the respective *beliefs* associated with the UBR informative evidence from the other two studies. The *beliefs* (0.83 and 0.87) were calculated according to the defined criterion. Also, note that these two evidence have two more constructs (efficiency and effectiveness for minor faults). Although the first study did not consider these variables, the *proposition* values associated with them were normally aggregated based only on the other two studies. The final aggregation results were:

PO-SP for efficiency-total faults; $Bel(\{\text{PO,SP}\}) = 0.97$,
PO-SP for efficiency-crucial faults; $Bel(\{\text{PO,SP}\}) = 0.97$,
PO for efficiency-important faults; $Bel(\{\text{PO}\}) = 0.99$,
PO for efficiency-minor faults; $Bel(\{\text{PO}\}) = 0.98$,

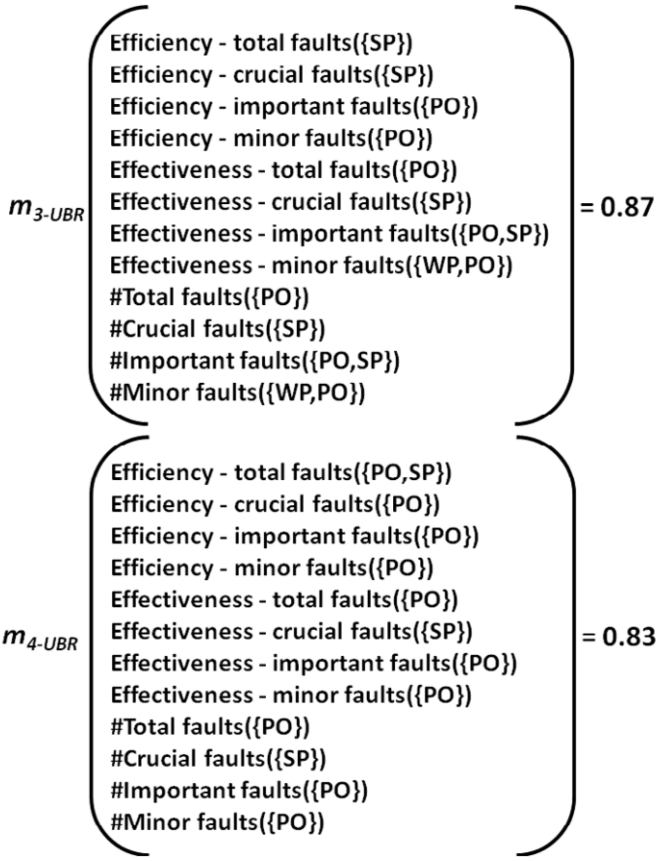

Fig. 7. DST representation for the UBR informative evidences from studies [23] and [24]

PO for effectiveness-total faults; $Bel(\{\text{PO}\}) = 0.98$,
SP for effectiveness-crucial faults; $Bel(\{\text{SP}\}) = 0.98$,
PO for effectiveness-important faults; $Bel(\{\text{PO}\}) = 0.96$,
PO for effectiveness-minor faults; $Bel(\{\text{PO}\}) = 0.83$,
PO for #total faults; $Bel(\{\text{PO}\}) = 0.98$,
SP for #crucial faults; $Bel(\{\text{SP}\}) = 0.98$,
PO for #important faults; $Bel(\{\text{PO}\}) = 0.96$,
PO for #minor faults; $Bel(\{\text{PO}\}) = 0.83$.

The final results show high belief values for most of the *proposition*. One possible interpretation when we look at a result like that is that the evidence aggregated until now had a high level of agreement for the *proposition* values (*i.e.*, all defined the same qualitative values for the *proposition*) and a great level of certainty associated with it (*i.e.*, the evidence came from reliable sources). That was exactly the case of the studies considered. The family of experiments had similar results and were all rigorously controlled studies generating reliable evidence.

However, given the relative small number of evidence considered , one could argue that the *belief* value of 0.98 is too high. While a valid argument, a possible answer for this is that the aggregation only takes known evidence into account, and

thus it represents the current body of knowledge. Moreover, as some aspects such as the *p-values* of the statistical tests were not considered we could have obtained lower belief values. Nevertheless, in this regard we are still learning how to deal with the belief values both in its determination before aggregation as in its interpretation after aggregation.

5 Discussion

Although our research is in an early and preliminary stage, the aggregation approach seems to be a feasible mechanism for evidence synthesis. It offers a simplified way of capturing the reliability in each evidence considered and translating them into a synthesized result including involved uncertainty. The resulted theory can be used as any 'normal' theory in the prediction and explanation of phenomena, and thus being potentially useful for decision making in practice [15]. In addition, it allows the incremental extension of the body of evidence by aggregating new evidence and at the same time shows how this evidence contributes to a theory.

Another important approach feature regards its capability to benefit from the theories' level of generalization to develop a growing evidence-based body of knowledge. This is implicitly present in the theory matching procedure shown in Figure 2. As the theories' levels of generalization indicate how coupled the theory is with a specific context, a less general theory is determined by the use of *value constructs* representing specific context characteristics or by a larger number of *value constructs* specifying the context in more details. So, evidence matching a less general theory will usually match a more general theory provided that the 'more specific' relationship between two *value constructs* can be defined. Using this, we could have different theories for the same technology mapping different or less specific contexts, resulting in a hierarchy of theories from the more specific to the more general level which could be used to define scope and range of evidence aggregation.

However, as said before, theories are only a simplification of the real world. And aligned to this idea, the evidence aggregation approach using theories aims at providing just an estimate of the effects of software technologies given a set of available evidence. In addition, in assuming that the process of building theories is correctly performed and the *constructs* are well-defined we left out a significant part of the reasoning involved in the construction of theories, which directly affects the quality of aggregation and could represent a threat to aggregation validity. Improper evidence representation by a theory (and a corresponding evidence-theory matching procedure), inaccurate *belief* estimation and misinterpretation of the theory's *constructs* can all be a source of bias or interference in the final result. As well summarized in the sentence 'garbage in, garbage out' [26] no research synthesis method is free from these issues.

To deal with this kind of issues and improve the aggregation rigour, the participation of more than one researcher can be recommended. The inter-rater agreement is widely used as a reliability criterion in many aggregation procedures such as case surveys [27] and is indicated in research involving qualitative content analysis in

general [28]. The agreement reliability can be used to address the subjective aspects of many steps of the proposed approach including the selection of the relevant *constructs* to represent each evidence/study and the definition of the *proposition* values for them. For belief estimation, besides the use of a questionnaire to assess the quality of evidence we plan to consider the strength of evidence [29]. The use of an evidence strength grading scheme (e.g., GRADE [30]) in conjunction with the assessment of its quality can maximize the accuracy of belief estimation and better represent the weight with which each evidence contributes to the whole body of evidence. This is particularly important, considering the fact that in expanding the concept of evidence and starting to accept all kinds of evidence including the weak and incomplete form. The next section, discusses how the proposed approach can be used with this type of evidence which is commonly produced in real-world settings in the form of lessons learned.

6 The Industry Perspective on the Generation, Use and Aggregation of Evidence

The research progress for the industry perspective is not as advanced as the Academia. So, this section only outlines how we expect to use the evidence representation and aggregation approach to support software development practice use in the continuous improvement process.

There are many approaches to promote continuous improvement in a software project, but in general they can be classified into two types: benchmark-based and feedback-based [31]. The first is characterized by methods guiding organizations towards the identification of the potential general improvements while the second helps an organization enhance its problem-solving capabilities. Categorized in the feedback-based type, post-mortems represent a learning activity with a knowledge management nature. As our approach deals with knowledge representation and aggregation, we believe post-mortems represent an interesting source of evidence (weak and informal) representing an input for our approach. Adding to this is the fact that post-mortems are simple to organize and conduct [4].

By using the proposed approach with post-mortems for continuous improvement, we expect the following benefits: improvement goals can be set based on existing theories providing, as result, better focus on post-mortem analysis activities, and the knowledge elicited can be formalized and reused in other projects and even combined with evidence produced in other projects. The current focus of our research is the elaboration of heuristics to map types of knowledge generated in post-mortems such as reports, Post-It [4], Ishikawa diagrams [5] and Cognitive Maps [6] for the representation used in the theories.

Two issues are expected to be addressed when using the proposed approach to support continuous improvement with post-mortems. In the first, which is relatively simpler, an evidence-based theory already exists and the development team decides to use a software technology based on the theory's expectations. After an iteration or an important project milestone, the team meets again to discuss the effects of

the software technology used in a post-mortem. At this point, the post-mortem activities could be more focused and be basically concentrated on evaluating theory propositions (*i.e.*, attribute a value with an associated belief given a set of defined criteria). We have yet to define those criteria. In addition, notice that the concept of a theory does not have to be directly given or explained to practitioners. The adoption of the proposed approach will be probably facilitated if the theory concept is mischaracterized as 'a simple visualization tool'. The second situation is more complex as if there is no theory about a given software technology then a set of heuristics should be provided to support the translation of post-mortems results into theory representation. The feasibility in providing such heuristics is indicated by the existence of procedures for extracting cause-effect relationships in post-mortem meetings using cognitive maps such as in [6].

By translating the produced knowledge to theory representation, even if it is already in a visual format such as Ishikawa diagrams, we gain the possibility of aggregating the evidence using the approach proposed. The body of knowledge can then, be used by practitioners in other projects for decision-making as well as by researchers proposing new hypotheses based on real-world data. However, even with these potential benefits, we do recognize that it will be quite difficult to make industrial practitioners use the approach in an unbiased way. Nevertheless, even with possible imperfections, we believe that this kind of evidence can be important to observe patterns emerging in real-world data, not only regarding the evaluation itself (i.e., negative/positive *proposition* values), but also in terms of technologies and environments (*i.e.*, *constructs*) as cited by practitioners.

7 Future Work

As ongoing research there are many opportunities regarding the approach proposed in this paper. One regards better detail in the procedure to match evidence to theories. This would improve the approach's degree of replication. Additionally, it is necessary to evaluate this approach in relation to other meta-analysis methods commonly used in Software Engineering. For this, we intend to replicate a published meta-analysis and/or simulate its results as described in [32]. Another ongoing work regards the investigation of how other theory properties such as testability, explanatory power and parsimony [15] can be defined in the aggregated theories. For testability, the propositions' refutation would be sufficient, but how can we do that when the evidence that could refute a theory has been aggregated by the theory itself? A possible criterion could be the definition of a level of conflict among the aggregated evidence. However, this deserves further investigation.

An additional investigation branch is the possibility of tailoring the systematic review process, as the approach using theories will probably affect the research question, as well as the information collected and analyzed from the articles. The concept of exploring multi-variables and their inter-relationships via a single meta-analysis is a topic of intense debate in many experimental disciplines [18]. At last, we can also expect to apply this approach to aggregate industrial experiences since

the visual notation of theories and the DST computation automation can bring great appeal to practitioners.

Acknowledgement

Prof. Travassos is a CNPq Researcher (grant 304795/2010-0). Authors thank the Experimental Software Engineering Group at COPPE/UFRJ and ESEM and CLEI anonymous reviewers for all the suggestions regarding previous versions of this work.

References

[1] Nicolay Am Worren, Karl Moore, and Richard Elliott. When theories become tools: Toward a framework for pragmatic validity. *Human Relations*, 55(10):1227 –1250, October 2002.

[2] Judith Segal, Antony Grinyer, and Helen Sharp. The type of evidence produced by empirical software engineers. In *Proceedings of the 2005 workshop on Realising evidence-based software engineering*, REBSE '05, page 1–4, New York, NY, USA, 2005. ACM.

[3] Austen Rainer, Dorota Jagielska, and Tracy Hall. Software engineering practice versus evidence-based software engineering research. In *Proceedings of the 2005 workshop on Realising evidence-based software engineering*, REBSE '05, page 1–5, New York, NY, USA, 2005. ACM.

[4] Torgeir Dingsøyr. Postmortem reviews: purpose and approaches in software engineering. *Information and Software Technology*, 47(5):293–303, March 2005.

[5] A. Birk, T. Dingsoyr, and T. Stalhane. Postmortem: never leave a project without it. *Software, IEEE*, 19(3):43 –45, June 2002.

[6] Finn Olav Bjørnson, Alf Inge Wang, and Erik Arisholm. Improving the effectiveness of root cause analysis in post mortem analysis: A controlled experiment. *Information and Software Technology*, 51(1):150–161, January 2009.

[7] Fabio Q.B. da Silva, André L.M. Santos, Sérgio Soares, A. César C. França, Cleviton V.F. Monteiro, and Felipe Farias Maciel. Six years of systematic literature reviews in software engineering: An updated tertiary study. *Information and Software Technology*, 53(9):899–913, September 2011.

[8] T. Dyba, B. A Kitchenham, and M. Jorgensen. Evidence-based software engineering for practitioners. *IEEE Software*, 22(1):58– 65, February 2005.

[9] Shull F. Feldmann, R. and M. Shaw. Building decision support in an imperfect world. In *2006 International Symposium on Empirical Software Engineering, 2006. ISESE '06. Proceedings*, page 33–35, August 2006.

[10] D.S. Cruzes and T. Dyba. Recommended steps for thematic synthesis in software engineering. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 275 –284, September 2011.

[11] Emelie Engström, Per Runeson, and Mats Skoglund. A systematic review on regression test selection techniques. *Information and Software Technology*, 52(1):14–30, January 2010.

[12] S. Charters, D. Budgen, M. Turner, B. Kitchenham, O. P Brereton, and S. Linkman. Objectivity in research: Challenges from the evidence-based paradigm. In *Software Engineering Conference, 2009. ASWEC '09. Australian*, pages 73–80. IEEE, April 2009.

[13] Samuel B Bacharach. Organizational theories: Some criteria for evaluation. *Academy of Management Review*, 14(4):496–515, 1989.

[14] Paul Davidson Reynolds. *A Primer in Theory Construction*. Bobbs-Merrill Co, June 1971.

[15] Dag I. K. Sjøberg, Tore Dybå, Bente C. D. Anda, and Jo E. Hannay. Building theories in software engineering. In Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 312–336. Springer London, London, 2008.

[16] R. Wieringa, M. Daneva, and N. Condori-Fernandez. The structure of design theories, and an analysis of their use in software engineering experiments. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 295 –304, September 2011.

[17] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, April 1976.

[18] Miller James. Applying meta-analytical procedures to software engineering experiments. *Journal of Systems and Software*, 54(1):29–39, September 2000.

[19] Paulo Sérgio Medeiros dos Santos, Guilherme Horta Travassos, and Marvin V. Zelkowitz. Action research can swing the balance in experimental software engineering. In *Advances in Computers*, volume Volume 83, pages 205–276. Elsevier, 2011.

[20] Jean Gordon and Edward H. Shortliffe. A method for managing evidential reasoning in a hierarchical hypothesis space. *Artificial Intelligence*, 26(3):323–357, July 1985.

[21] Thomas Thelin, Per Runeson, and Björn Regnell. Usage-based reading—an experiment to guide reviewers with use cases. *Information and Software Technology*, 43(15):925–938, December 2001.

[22] T. Thelin, P. Runeson, C. Wohlin, T. Olsson, and C. Andersson. How much information is needed for usage-based reading? a series of experiments. In *Empirical Software Engineering, 2002. Proceedings. 2002 International Symposium n*, pages 127– 138. IEEE, 2002.

[23] T. Thelin, P. Runeson, and C. Wohlin. An experimental comparison of usage-based and checklist-based reading. *IEEE Transactions on Software Engineering*, 29(8):687– 704, August 2003.

[24] T. Thelin, C. Andersson, P. Runeson, and N. Dzamashvili-Fogelstrom. A replicated experiment of usage-based and checklist-based reading. In *10th International Symposium on Software Metrics, 2004. Proceedings*, pages 246– 256. IEEE, September 2004.

[25] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9–10):833–859, August 2008.

[26] Donald Sharpe. Of apples and oranges, file drawers and garbage: Why validity issues in meta-analysis will not go away. *Clinical Psychology Review*, 17(8):881–901, December 1997.

[27] Rikard Larsson. Case survey methodology: Quantitative analysis of patterns across case studies. *Academy of Management Journal*, 36(6):1515–1546, 1993.

[28] U.H Graneheim and B Lundman. Qualitative content analysis in nursing research: concepts, procedures and measures to achieve trustworthiness. *Nurse Education Today*, 24(2):105–112, February 2004.

[29] Tore Dybå and Torgeir Dingsøyr. Strength of evidence in systematic reviews in software engineering. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '08, page 178–187, New York, NY, USA, 2008. ACM.

[30] Grading quality of evidence and strength of recommendations. *BMJ : British Medical Journal*, 328(7454):1490, June 2004. PMID: 15205295 PMCID: PMC428525.

[31] Andreas Birk and Dietmar Pfahl. A systems perspective on software process improvement. In Markku Oivo and Seija Komi-Sirviö, editors, *Product Focused Software Process Improvement*, volume 2559 of *Lecture Notes in Computer Science*, pages 4–18. Springer Berlin / Heidelberg, 2002.

[32] Oscar Dieste, Enrique Fernandez, Ramon Garcia Martinez, and Natalia Juristo. Comparative analysis of meta-analysis methods: When to use which? In *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*, pages 36–45. IET, April 2011.