A Taxonomy of Evaluation Approaches in Software Engineering

Alexander Chatzigeorgiou, Theodore Chaikalis, Georgia Paschalidou, Nikolaos Vesyropoulos, Christos K. Georgiadis, Emmanouil Stiakakis

Department of Applied Informatics,
University of Macedonia, Thessaloniki, Greece

ABSTRACT

As in any academic discipline, the evaluation of proposed methodologies and techniques is of vital importance for assessing the validity of novel ideas or findings in Software Engineering. Over the years, a large number of evaluation approaches have been employed, some of them drawn from other domains and other particularly developed for the needs of software engineering related research. In this paper we present the results of a survey of evaluation techniques that have been utilized in research papers that appeared in three leading software engineering journal and propose a taxonomy of evaluation approaches which might be helpful towards the organization of knowledge regarding the different strategies for the validation of research outcomes. The applicability of the proposed taxonomy has been evaluated by classifying the articles retrieved from ICSE'2012.

Categories and Subject Descriptors

• Software and its engineering ~ Software verification and validation • Software and its engineering ~ Maintaining software

Keywords

Taxonomy; evaluation; classification; software engineering.

1. INTRODUCTION

A software practitioner would get confidence in a new software engineering technique or tool only if he is presented with sufficient evidence that the proposed approach works well in a real context and actually improves current practice with limited cost. An academician would outwardly reject any suggestion of a new methodology if it lacks proper evaluation to highlight strengths and limitations. As a result, it is a common truth among members of industry and academia that evaluation is one of the cornerstones of any novel research proposal. The same holds for any other discipline, but particularly in software it is believed that the "ease" of applying the proposed approaches by means of software tools on software artifacts can generate reliable evaluation results that can back up any proposed methodology or technique.

In the literature of software engineering there is an abundance of evaluation approaches of several kinds, often with very fine-grained differences among them, since the evaluation strategies are designed to satisfy the needs of each particular area of research. Motivated by the lack of a systematic classification scheme for evaluation techniques and the importance that peer reviewers pay to the need for thorough and extensive evaluation, we introduce in this paper a taxonomy of evaluation strategies employed in software engineering research.

Classification systems have been proposed for the discipline of Computing aiming at organizing subjects by area and facilitating proper indexing and retrieval of information. The dominant classification scheme has been devised by the Association for Computing Machinery and the latest version, published in 2012, has been developed as a poly-hierarchical ontology that can be utilized in semantic web applications [1]. Vessey et al. [19] identified limitations in existing classification systems which fail to address the combined needs of the Computer Science, Software Engineering and Information Systems disciplines. The authors proposed a richer, multi-faceted classification system to capture the interests of all three disciplines, but the five dimensions of the scheme do not include the employed evaluation approach. Literature reviews have also been performed to examine the state of software engineering research and categorized papers based on the addressed topics and research methods [6], but no special focus to the employed evaluation techniques was given.

The proposed taxonomy aims at identifying and classifying evaluation approaches employed in software engineering and is based on the study of articles that appeared in three leading journals in a timespan of one year. A major prerequisite for a successful classification [2], is the ability to ascertain the fundamental characteristics on which the classification is to be based. To this end, the proposed taxonomy has established a set of well-defined and unambiguous axes according to which evaluation approaches can be discriminated.

A classification scheme is of great value in any scientific discipline since it allows the development of theories and methodologies which are delimited to certain categories of an entire area of research. Regarding the classification of evaluation approaches in software engineering the following benefits can be identified:

- any researcher developing a novel approach can contrast his own evaluation strategy to other approaches and consult papers with a similar topic regarding their (classified) evaluation approach.
- the classification of evaluation approaches might provide the ground for systematic specification of principles that should guide each particular evaluation category.
- the explicit identification of the employed evaluation approach within an article might be beneficial for the audience of a novel

BCI'15, September 02-04, 2015, Craiova, Romania.

DOI: http://dx.doi.org/10.1145/2801081.2801084

theory, method or tool to easily grasp the validation strategy and its relation to other approaches.

- provided that a classification is enhanced and improved by the community, it might be useful for better bibliographic organization or indexing (i.e. one might be able to look up for articles employing a specific evaluation approach).

A review and categorization of evaluation techniques of modelling methods has been introduced by Siau & Rossi [16]. Besides the fact that research efforts in software engineering have a much broader variety of evaluation methods, which are not covered by this study, evaluation approaches are classified only on a first level and without reference to explicit criteria for the classification. However, the three categories of identified evaluation techniques, namely feature comparison, theoretical and conceptual investigation, and empirical evaluation are similar to the first-level categories proposed in our taxonomy.

Moreover, in the classic paper by Mary Shaw [15] presenting a characterization for software engineering research, a coarse-grained classification of papers based on the type of employed research validation has been introduced. Validation techniques have been classified as Analysis, Experience, Example, Evaluation, Persuasion and Blatant assertion. However, the criteria for identifying the validation type are rather vague and no further statistics for each evaluation type nor representative examples have been provided.

The rest of the paper is organized as follows: Section 2 presents general concepts associated with the activity of classification. The context of the proposed study is outlined in Section 3. The methodology that has been followed is presented in Section 4, while key terms are defined in Section 5. The proposed taxonomy is introduced in Section 6. A discussion on the relative frequency of each category is included in Section 7, while in Section 8 we present validation results. Threats to validity are identified and discussed in Section 9. Finally, we conclude in Section 10.

2. TAXONOMIES: CLASSIFICATION CONCEPTS & PRINCIPLES

A taxonomy, considering also the Greek origin of the term (taxis: arrangement and nomos: law, method) aims at organizing a collection of objects in a hierarchical manner to provide a conceptual framework for discussion and analysis. The primary goal of a taxonomy is the classification of a set of examined items or concepts based on a set of pre-defined criteria and based on the identification of similarities among items. 'Classification' can be defined as the arrangement of concepts into groups based on observable or inferred properties [17]. Once a classification system has been adopted and acknowledged by a community of practitioners or researchers, its use allows the members of the community to generalize, communicate, apply, and compare their findings [17], [19]. The practice of classifying living things and plants dates back to the Greek philosophers and since then taxonomies have been important for biodiversity-based sciences. The science of developing systematic taxonomies has its origins in biology, anthropology, zoology, and botanology and even in the 17th century there was an abundance of botanical classification systems.

Besides the introduction of taxonomies in various scientific disciplines, research has also been performed regarding the proper classification strategies. The difficulty of correct classification emerges persistently in scientific work, but it is widely accepted that the adoption of proper principles of classification may minimize the points of disagreement. This means that it is important to identify the purpose of the classification and the criteria by which the researchers group objects and concepts into some system [17], [19].

Taxonomies are unavoidably dependent on the particular population of items on which they are built, whereas the goal of a taxonomy is to enable the classification of any object or concept that may be encountered and to accommodate even cases that have not been identified or proposed in the past. There are two alternatives that can be followed in order to construct a taxonomy. In a top-down approach the overall scheme is based on an a-priori logical understanding of the corresponding field and is used to categorize analyzed cases. For example, Glass et al. [6], in a study of software engineering research areas, defined in advance the categories into which objects (papers) would be classified. The alternative approach would have been to use a bottom-up classification driven by the papers themselves as they have been examined. The two approaches can be combined in an interactive manner where neither principles nor the actual data necessarily dominate in the construction of a taxonomy [17].

Errors that can be encountered when organizing taxonomic knowledge in the form of ontologies have been reported by Gómez-Pérez [7]. Such errors include, for example, partition errors (e.g. assuming that dogs and cats form a subclass partition of the set of mammals, it would be an error to define a class of animals as a subclass of both dogs and cats) or circularity errors, when a class is defined as a specialization or generalization of itself. However, such errors cannot occur in the proposed taxonomy since it is based on a directed tree, where each node can have only a single parent and no circuits are allowed by definition. Other types of errors, such as semantic inconsistency errors or incomplete concept classification are related to incorrect decisions during the classification and are discussed under threats to validity.

3. CONTEXT OF THE STUDY

The survey on which the proposed taxonomy is based has been performed by three faculty members and three PhD candidates of the Department of Applied Informatics at the University of Macedonia, Thessaloniki, Greece. Articles have been retrieved from three major journals in the field of Software Engineering (TSE: IEEE Transactions on Software Engineering, TOSEM: ACM Transactions on Software Engineering and Methodology, and JSS: Elsevier's Journal of Systems and Software). Therefore, we have distributed the survey to 3 groups (formed by a faculty member and a PhD candidate) that have worked independently on the analysis of the articles. During weekly meetings all articles from each group were revisited until consensus on the categorization of the employed evaluation techniques was reached.

For each article, we have collected and documented the bibliographic data (title, authors, journal, issue), the free keywords provided by the authors, its classification according to the 2012 ACM Computing Classification System [1], the employed evaluation approach(es), the number of pages devoted to the evaluation, as well as the total paper length. For all the examined journals, the articles that appeared in the corresponding 2012 volume have been employed (for TOSEM the time spans from December 2011 to November 2012, to cover all four issues that appear yearly). In total, 81 articles from TSE, 24 articles from

TOSEM, and 207 articles from JSS have been examined. Information on all analyzed articles can be found in the accompanying web page (http://se.uom.gr/taxeva) that contains all details of the study.

In order to focus on evaluation techniques employed strictly in the field of Software Engineering, articles that clearly did not belong in this domain, such as papers focusing on hardware related issues or on cryptographic algorithms, have been excluded. Moreover, there is a large body of papers that do not introduce any approach, method or tool such as Systematic Literature Reviews, surveys, mapping studies or studies that investigate phenomena, trends or relations among variables. For example, a study investigating the relation between maintainability and design faults or metric values based on historical data, does not propose any technique or methodology that needs to be evaluated. After this filtering process, 58 articles from TSE, 22 articles from TOSEM and 53 articles from JSS have been fed to the survey, a total of 133 articles.

4. TAXONOMY-BUILDING METHODOLOGY

Classification schemes offer a significant advantage by breaking the continuous 'real' world into discrete and collective categories suited for further analysis [13]. However, researchers share no common ground on how taxonomies should be derived: some schools of thought regard the taxonomy building process as a deductive activity based on intuition or previously existing knowledge and theory [18]. In this case categories are conceptualized and named before specific objects are placed into the taxonomy, which by many authors is called a typology in this case. Other approaches rely on a more empirical method, where the identification and naming of categories is performed in an inductive manner, after the analysis of examined data. The criteria which have been used for the proposed taxonomy of evaluation approaches in software engineering result in monothetic categories. This means that each category of the taxonomy is defined in terms of criteria that are both necessary and sufficient in order to decide membership, which is analogous to the Aristotelian definition of a class. The employed process for building the taxonomy is based on the methodological suggestions by Nickerson et al. [11] and Steininger et al. [18], which in turn adopt the principles laid out by Bailey [2]. An overview of the employed methodology is shown in Fig. 1.

The first step consists in the selection of entities under investigation in order to retrieve the suitable criteria, their manifestations and the corresponding categories. In our case this step corresponds to the retrieval of papers from the selected journals. The next step deals with the definition of the appropriate unit of analysis [18] that will dictate the classification. We have employed as unit of analysis the notion of 'theme' which captures an entire aspect of each examined entity, in our case the employed evaluation approach(es). This step results in a collection of raw evaluation techniques present in each paper. For example, one paper might employ a measure of CPU time and another a measure of memory consumption in order to demonstrate an improved performance over previous approaches. Next, a grouping of the identified raw characteristics is performed by iteratively reducing and qualitatively bundling similar concepts, until broader categories emerge [18]. With respect to the previous example, speed and memory measurements can be grouped under a more generic 'performance analysis' term.

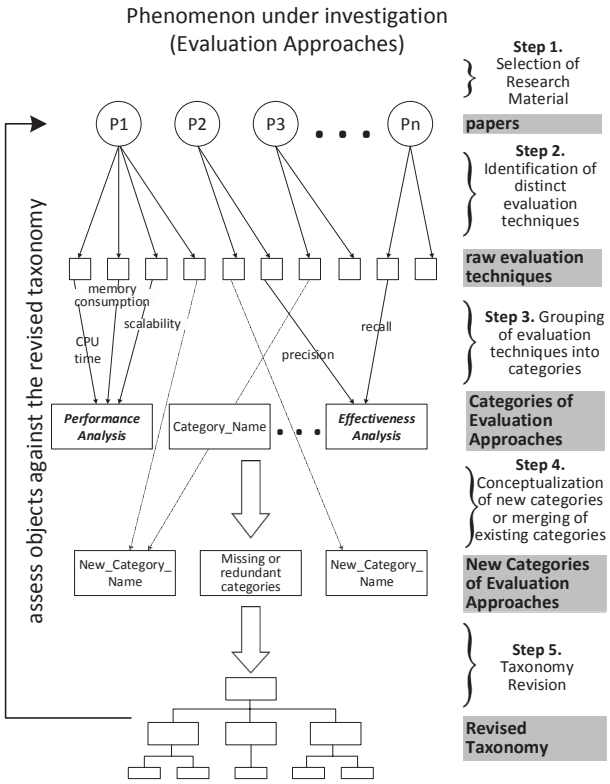

Fig. 1. Taxonomy-Building Methodology (adapted from Steininger et al. [18])

The result of this process is an initial version of the sought taxonomy. From this point on, an iterative investigation should be carried out, where the goal in each iteration is to look for missing categories (which would express the concepts of a specific subgroup of entities in a better way) or redundant classifications which should be removed. Each iteration might end up in a modified version of the taxonomy and the taxonomy is then re-evaluated against all selected entities [11].

5. TERMINOLOGY

The taxonomy of evaluation approaches that will be presented is based on the distinct characteristics of each evaluation approach. Therefore, it is vital to provide a precise definition of the employed terms so as to enable an unambiguous interpretation of each research work. The following terms appear in one or more nodes of the taxonomy:

Performance: We employ the most typical definition of performance, originating from computer architecture, according to which performance refers to the amount of work that a system/computer/program can perform in a given time or for given resources [9]. Whenever performance is mentioned in the taxonomy, the involved evaluation techniques measure either execution time or the extent of resource utilization.

Effectiveness: By effectiveness we refer to the extent by which a proposed technique/methodology accomplishes the desired goal. For example, a testing approach is effective if it reveals a large number of bugs. We have avoided the use of other similar terms, such as efficiency (which is often regarded as a synonym to performance), accuracy (which is often employed in the context

of precision/recall analysis) or efficacy (which is rarely used). According to this definition, most research papers presenting a novel approach attempt to demonstrate improved effectiveness and as a result this term appears in many of the following evaluation categories.

Benchmark: A benchmark is a standard, ideally widely acknowledged data set (consisting of tasks, collections of data items, software etc.) that has been designed with the purpose of being representative of problems/cases that will occur frequently in real domains. Obviously, the advantage of using a benchmark is that comparative analysis of performance and/or effectiveness is possible [10]; techniques are applied to the same set of data and the results of each approach can be measured and assessed against others.

6. PROPOSED TAXONOMY

The proposed taxonomy of evaluation approaches is graphically depicted in Fig 2. Leaf nodes (i.e. nodes that cannot be further decomposed) represent actual evaluation approaches adopted by one or more of the examined papers. In general, we have attempted to build a classification that leads to unambiguous and mutually exclusive subgroups, which when considered all together provide a complete coverage for the evaluation techniques present in the examined articles.

The top axis according to which articles have been classified (nodes below the root "Evaluation Taxonomy") refers to the key distinctive characteristics of the employed evaluation strategy. From this perspective, evaluation approaches can be classified either as a) "Comparison to similar approaches", b) "Formal Proof" or c) evaluation employing a sort of experimentation on one or more case studies ("Case Studies"). No article has been found whose evaluation technique could not be classified into one of these three top-level categories. Moreover, there are cases where in a single article two or more evaluation techniques co-exist. The goal was to create a taxonomy that is not too fine-grained in order to make the classification scheme as general as possible.

Next, we discuss individual evaluation approaches that appear as leaf nodes in the taxonomy. However, we provide also a description of intermediate nodes to clarify the classification criteria. The description of the taxonomy shown in Fig. 2 is performed by traversing the tree in a breadth-first manner.

E1 Comparison to Similar Approaches

In this category fall the evaluations that entail a more or less systematic comparison to similar approaches that have treated the same field of research in the past. The goal is in all cases to make clear the advantages and disadvantages over previous work, either by qualitative or quantitative means and usually to highlight the added value of the proposed technique. It can be further analyzed to the following subcategories:

E1.1 Qualitative Comparison (Listing of pros/cons)

Evaluations of this type attempt to compare the proposed technique with previous approaches by listing (i.e. in a verbal fashion) the pros and cons. Although this is usually a process that is performed in the presentation of the related work (which we do not regard as evaluation in this study), several papers in the literature of software engineering devote a significant part of their evaluation in qualitatively contrasting several aspects of the proposed approach to previous work.

A number of papers complement this type of evaluation with other, more quantitative types of evaluation as comparison to previous work is a rather usual way to present the state-of-the-art. However, there are cases, where mainly due to the particularities of the field this type of qualitative evaluation might be the only possible approach.

E1.2 Quantitative Comparison

Although a qualitative comparison to a previously presented approach certainly provides insight into the main methodological differences or drawbacks, many researchers believe that quantitative comparison is more accurate and objective as an evaluation approach. Therefore, the second subcategory of the "Comparison to similar approaches" consists in the comparison of the proposed technique/method/tool in terms of quantitative aspects of the achieved result (E1.2). Works in this category can be further classified based on whether benchmarks are used or not.

E.1.2.1 Non-Benchmark based

In this category, papers evaluate the proposed approaches by comparing them to other alternatives and attempt to extract quantitative measures based on test benches/case studies selected or developed ad hoc. Numerous parameters can be assessed in the context of the evaluation of a particular technique or method. However, driven by the aspects that have been analyzed in the examined papers, we have ended up in two general quality properties which appear to be of interest, namely performance and effectiveness. Quite often, both of these properties are evaluated.

E.1.2.1.1 Performance Analysis (ad hoc samples)

Although software engineering is not primarily aiming at assessing the time and space complexity of software systems, quite often, to demonstrate the feasibility and scalability of a novel technique it is required to evaluate performance measures. Parts of the evaluation in papers of this category aim at illustrating the reduced amount of time, memory or power consumption, required by the corresponding approach. In the usual case, the proposed implementation is executed on systems or data sets selected ad hoc and the properties of interest are measured during or after the execution.

E.1.2.1.2 Effectiveness (ad hoc samples)

Papers in this category perform an evaluation by comparing how well the proposed technique works against similar approaches employing as test benches either custom made case studies (e.g. software systems or sets of data) or examples selected from public repositories. Approaches are compared in terms of their effectiveness and numerous measures are used for this purpose. This evaluation type, as it is reasonable to expect, includes a large number of papers in our study.

Probably the most systematic use of a test bench is related to the measurement of the effectiveness of a technique in terms of precision and recall. These measures have been originally employed in information retrieval [12] to classify the accuracy in retrieving relevant documents given a search term. Out of the 18 articles in this category, 7 employ precision and recall for their evaluation.

Evaluation approaches in this subcategory, as well as in other branches of the taxonomy, can be further subdivided based on whether the evaluation has been driven by explicit research questions which have been stated by the authors.

E.1.2.1.2.1/ E.1.2.1.2.2 Use of Research Questions (Effectiveness analysis on ad hoc samples)

The articles that explicitly contain research questions according to the defined goals [3] form a distinct group with specific and easily identifiable characteristics in the presentation of their evaluation approach. These research questions drive the entire evaluation strategy. Results are collected, analyzed and interpreted in order to derive answers to the posed questions. On the other hand, methods or techniques are often compared with previous approaches without relying on explicitly stated research questions.

E1.2.2 Benchmark based

As already mentioned, benchmarks constitute a relatively objective approach for comparing different methods [10]. Thus, articles that employ benchmarks can systematically assess both the effectiveness of the proposed techniques/methodologies and the performance of the underlying computations or algorithms. This is the criterion according to which the following two sub-categories are separated.

E1.2.2.1 Performance Analysis

Benchmarks have been originally created in computing as a means of assessing performance characteristics of computer hardware, such as the number of floating point operations per second for a CPU. Gradually, the use of benchmarks has been extended for the comparison, usually in terms of execution time, of software systems. However, almost in all cases, this type of evaluation is complemented by other strategies, in order to assess the effectiveness of the proposed technique/methodology as well.

E1.2.2.2 Effectiveness Analysis

The other goal of using benchmarks is to compare how well different approaches accomplish a certain task. In other words the evaluation aims at assessing the effectiveness of each technique/methodology and as previously, this can be performed by any measure that is suitable for a particular property of interest. Papers in this category can be further refined depending on whether their evaluation relies on research questions or not (E1.2.2.2.1/ E1.2.2.2.2).

E2 Formal Proof

The application domain of several methodologies and techniques calls for a formal treatment of the evaluation process or parts of it. By formal treatment, the use of a mathematically-based approach for proving theorems, properties, invariants or the correctness of a system is meant. In a formal proof all logical inferences have been checked, ideally all the way back to the fundamental axioms, all logical steps are supplied and no appeal to intuition is made [8]. Naturally, not all of software engineering research can benefit from the application of formal methods, but whenever appropriate the known benefits of consistency-checking, automatic defect identification, and high-level of rigor can be enjoyed.

Although it would be difficult, and probably beyond the scope of this paper, to investigate means for distinguishing between different approaches in formal proving, we have attempted a first-level classification of articles in this category. The criterion is related to the completeness of the formal proof, i.e. the extent by which the mathematical reasoning validates the entire approach that is being proposed or ensures the fulfillment of certain properties.

E2.1 Properties Fulfillment

To confront the lack of formality in several areas of software engineering, researchers have often proposed sets of mathematical properties that provide a supportive underlying theory. A notable example is the mathematical properties for software metrics, such as complexity, coupling and cohesion, defined by Briand et al. [5]. These properties provide a framework against which newly proposed measures can be validated. In this category, part of the evaluation is devoted to proving that either the proposed methodology or the system resulting from the application of the proposed approach fulfills certain properties and this is carried out in a formal fashion.

E2.2 Theorem Proving

There are domains in Computer Science, such as model checking, where the completeness and soundness of a proposed algorithm or approach can be proved employing mathematical methods such as induction and contradiction or by direct proofs that logically combine axioms, definitions and previous proofs. Evaluation in such papers is often complemented by other techniques, such as experimental evaluation to assess the effectiveness, performance or user friendliness of the accompanying tools.

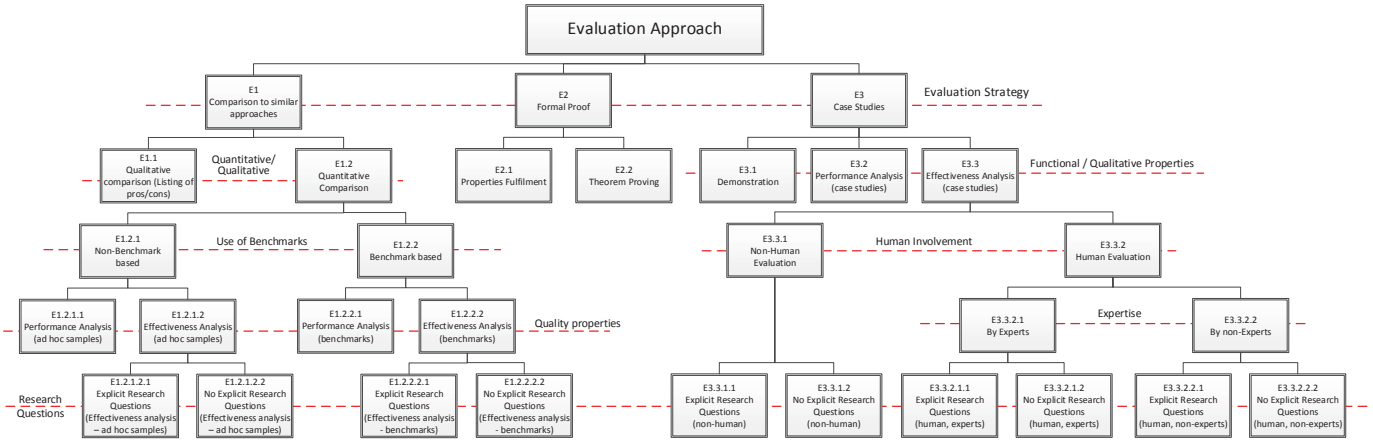

Fig. 2. Taxonomy of evaluation approaches

E3 Case Studies

The vast majority of research articles evaluate the proposed techniques and methods on one or more case studies, although the term "case study" is not always being used. The corresponding evaluation section can be found under the names, Case Studies, Case Study Evaluation, Empirical Evaluation, Empirical Results, Empirical Studies etc. In several cases, the corresponding section is termed Experiments or Experimental Results, whereas according to the strict definition of an experiment which assumes the manipulation of one or more variables, the presented evaluation does not constitute an experiment [20]. This evaluation strategy consists in the application of the proposed tool, algorithm, technique or method on artificially constructed or most often selected case studies, which can be either data sets, software systems, or any other artifact of the software development process. During this process, results are obtained and discussed to demonstrate the feasibility, performance or effectiveness of the approach.

E3.1 Demonstration

The evaluation section of several papers in the literature of software engineering often includes (and in many cases consists only in) a demonstration that the proposed approach is feasible, by applying it on a particular case study and discussing the implications, degree of complexity, points of interest or threats to validity. No quantitative measures or external evaluators are being used and the discussion of the case study is carried out from the perspective of the authors. According to our findings, this type of evaluation constitutes, in combination with other approaches, the second most frequent strategy. This is probably due to the fact that the demonstration of an approach on examples provides the greatest flexibility to the authors to illustrate the benefits of the proposed technique as it is not bounded by specific measures or comparisons. Thirty three papers (24.8%) in our study include demonstration as part of their evaluation. However, only in 14 papers, demonstration has been the only way of evaluating the proposed approach.

E3.2 Performance Analysis (Case Studies)

As mentioned in category E1, the need to demonstrate the applicability and scalability of the proposed approach calls for appropriate measurements of execution time or required resources. Obviously, it would be ideal to measure performance on specifically designed and acknowledged benchmarks and to compare performance with other similar approaches. However, in several research areas appropriate benchmarks have not been proposed, while in other cases there are no prior or similar techniques to the one that is being introduced in a paper. In such cases, performance analysis on case studies is carried out and results (time, memory or power consumption) are critically analyzed and discussed by the authors, inevitably introducing a certain level of subjectivity. It should be noted, that in many articles that include performance analysis based on case studies the focus is on scalability. In other words, performance is illustrated for varying values of selected parameters that relate to the problem size.

E3.3 Effectiveness Analysis (case studies)

The most common reason for performing evaluation on case studies is to show the effectiveness of the examined approaches, as experimentation on selected examples allows the collection of various quantitative data that enable authors to highlight the advantages of each technique. The first axis according to which evaluation approaches can be discriminated is whether humans are involved in the evaluation process.

E3.3.1 Non-Human Evaluation

In this type of evaluation the collected results do not require the subjective interpretation of independent reviewers, experts or not. Evaluation approaches in this subcategory, as well as in the subcategories where human involvement is necessary, can be further subdivided based on whether the evaluation has been driven by explicit research questions which have been stated by the authors (E3.3.1.1/ E3.3.1.2).

E3.3.2 Human Evaluation

Several methods and techniques by nature require human expertise in order to evaluate their quality properties. There are numerous examples such as the assessment of usability, the examination of whether automatically applied source code changes are in agreement with human intuition and whether models and diagrams satisfy rules of thumb, often of an aesthetic nature, implicitly considered by humans. Evaluation approaches that rely on humans can be further classified based on whether the evaluators are experts or not.

E3.3.2.1 Human Evaluation by Experts

Stakeholders in the software development process with significant experience (designers, architects, developers, project managers, and testers) are often called upon to provide feedback on technical aspects which cannot be assessed by other means. The notion of expert is usually related to their experience both in terms of time and the involvement in projects/products of the same domain. The definition of expert is often stretched to include students with significant development/research experience. Once again, evaluation approaches in this category can be carried out by stating explicit research questions or not (E3.3.2.1.1/ E3.3.2.1.2).

E3.3.2.2 Human Evaluation by non-Experts

Various aspects of software engineering related research can be evaluated by subjects that have limited experience in the corresponding domain since the assessment might seek the opinion of target users or developers who are not familiar with similar methods, resembling the average software practitioner. The feedback retrieved from these subjects, form quantitative data which can either be used to answer investigated research questions or analyzed without following previously stated hypotheses (E3.3.2.2.1/ E3.3.2.2.2).

7. POPULARITY OF EVALUATION APPROACHES

To provide an overview of the 'popularity' of each evaluation technique without loss of the taxonomy structure, we employed the modified bar chart shown in Fig. 3. The structure of the taxonomy is reflected on the chart by nesting evaluation types. Each bar corresponds to a node of the taxonomy (intermediate or leaf) and its length is proportional to the frequency of the corresponding category.

The frequency (or number of papers) of the evaluation types corresponding to intermediate nodes of the taxonomy (i.e. non-leaf nodes) cannot be simply obtained as the sum of their subordinate leaf node frequencies (or number of papers) as there are papers employing more than one evaluation approaches.

Therefore, the number of papers for each intermediate (int) node x is calculated as follows:

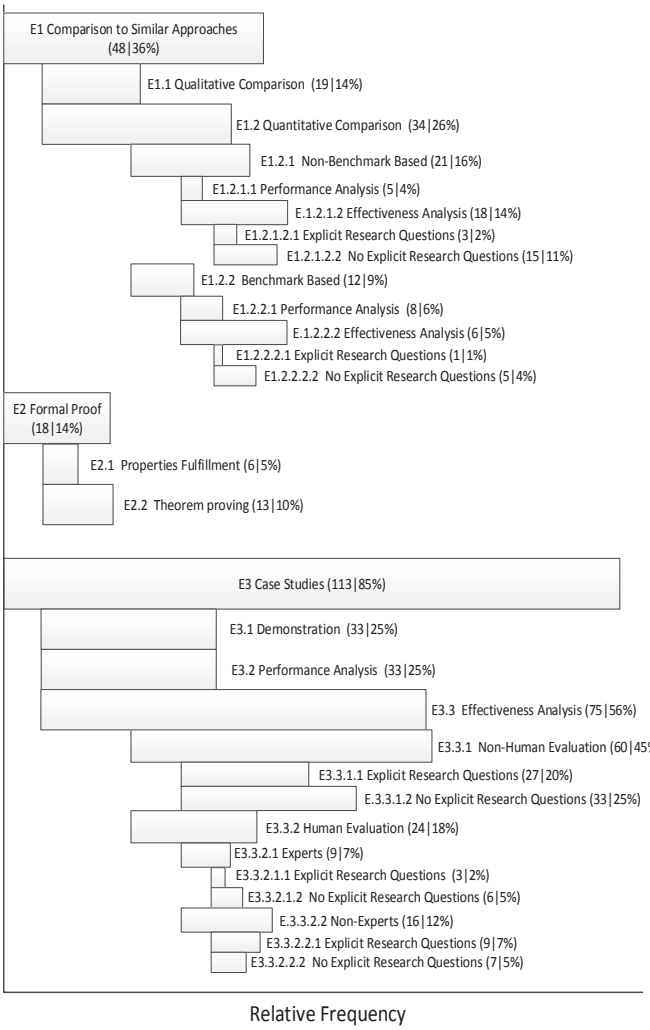

Fig. 3. Relative frequency of each evaluation technique

$$|\ papers(\text{int node } x)\ | = \left| \bigcup_{i \in \text{subordinate leaf nodes of } x} \{papers(i)\} \right| \quad (1)$$

As it can be readily observed, evaluation based on case studies is the dominant form of evaluation among the examined articles. Due to the inherent nature of software engineering research, this is a reasonable finding since case studies are usually software systems in which a technique, methodology or approach can be easily applied. Since the primary goal in an evaluation is to convince that the proposed approach accomplishes the intended objectives, among all subcategories of "Case Studies", effectiveness analysis has the lion's share.

8. VALIDATION OF TAXONOMY

By definition, it is difficult to assess whether taxonomies are valid, since their construction relies on the subjective interpretation of categories. To investigate the validity of the taxonomy that has been derived from the analysis of articles in three software engineering journals we have attempted to actually apply the taxonomy on articles which have not been considered during its development. In particular, we have classified the papers from the Main Track of the 34th International Conference on Software Engineering (ICSE'2012). In total, 87 articles have been considered for the validation. The classification of articles has been performed independently by two reviewers. For each article the following pieces of information have been recorded:

a) Whether the paper actually introduces any technique or method which necessitates its evaluation. Sixteen papers have been excluded based on this criterion.

b) Whether the paper could be mapped to any of the derived classification categories. Without any obstacles, all papers have been successfully mapped to one or more evaluation strategies.

c) The corresponding category code as well as the number of categories to which a mapping was found.

It should be mentioned that the mapping of evaluation strategies appearing in ICSE articles to the categories of the taxonomy shown in Fig. 2 has been effortless and no ambiguous cases have been encountered. Moreover, no discrepancies between the classifications performed by the two reviewers have been observed. We believe that this is primarily due to the existence of precise and distinct axes that served as guides for the selection of the appropriate categories. At the end of this mapping process, articles have been assigned to all leaf nodes of the taxonomy.

Regarding the distribution of evaluation approaches over the seventeen identified categories, we have observed similarities between proportions for the ICSE articles and those that formed the basis for building the taxonomy, which implies that the philosophy of evaluation in the examined journal articles does not differ significantly from that in the ICSE articles. Both distributions are shown in Fig. 4.

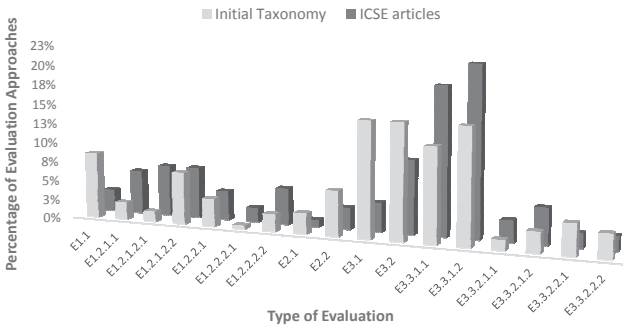

Fig. 4. Distribution of evaluation approaches over identified categories

For ten out of the seventeen categories the difference between the corresponding percentages is less than 3%. Major differences can be observed for the categories E3.1 (Demonstration) and E1.1 (Qualitative comparison). In other words, the ICSE articles combine less frequently verbal comparison with previous techniques and evaluation through demonstration with other evaluation strategies, possibly due to space limitations. If these two paper categories are excluded, then a chi-square test of independence, reveals that the null hypothesis can be accepted ($p=0.120$), i.e. the distributions are not significantly different.

9. THREATS TO VALIDITY

Building a classification scheme based on the analysis of selected items to be categorized is by definition a process that can raise concerns and objections depending on the particular viewpoint. For the classification of evaluation types in software engineering papers, the following threats have been identified.

The design of the taxonomy itself poses threats to construct validity concerning the relation between theory and observation

[20], since categories required for further classification might have been missed. For example, for the top level of the taxonomy, one could have identified other, more fine-grained strategies for the evaluation of proposed approaches. Overlooking concepts that exist in a domain is listed as an "incomplete classification error" in the study by Gómez-Pérez [7]. However, as shown in Fig. 2, the taxonomy has been organized along well-defined and discrete axes that serve as guides for the identification of appropriate subcategories. For some of the axes there is no ambiguity at all (for example, in the case of qualitative vs. quantitative comparison), while in other axes (such as the properties that are being investigated in the evaluation), the selection of subcategories has been driven by the content of the research articles that have been studied. Furthermore, each choice regarding the taxonomy (axes, subcategories, labeling, and levels) has been made only when consensus among all six authors had been reached.

A relevant, but slightly different threat concerns the external validity, which is the ability to generalize our findings. In the case of a taxonomy, generalization is related to the applicability of the classification scheme to other papers beyond those that have been included in the study. The selection of a different set of articles might have revealed other axes for the classification or different nodes in the taxonomy. However, this threat is mitigated by the inclusion of papers published in three leading software engineering journals and for a timespan of twelve months, partially ensuring that research efforts from representative areas in software engineering are reflected. Moreover, the application of the taxonomy on the papers from one additional scientific source revealed that all articles could be mapped effortlessly to the categories of the taxonomy.

Finally, another threat is related to the subjective interpretation of the evaluation strategies that each paper adopts. In other words, papers might have been misclassified leading to different frequencies per evaluation type. Such incorrect classifications are termed as "semantic inconsistency errors" [7]. To confront this subjectivity, the classification of each individual article and the collected information for each paper have been reviewed by all six authors. Moreover, the classification of the articles is strongly affected by the use of the specified axes, thus limiting the possibility for erroneous categorizations.

10. CONCLUSIONS

As in many scientific disciplines, approaches which are proposed in Software Engineering have to be rigorously evaluated in order to highlight their strengths and implications. As a result, in the software engineering literature there is a vast amount of different evaluation techniques designed and executed to serve the needs of each particular research effort. Motivated by the attention that the peer reviewing communities pay to the need for extensive and thorough evaluation and the lack of an appropriate classification scheme, we have attempted to introduce a taxonomy of evaluation approaches. Based on the analysis of articles in three leading software engineering journals and for a timespan of twelve months, we have identified criteria according to which evaluation techniques can be categorized. The study was based on 312 selected papers, of which 133 have been analyzed after the application of exclusion criteria. We have identified 17 evaluation types that any approach can adopt either individually or in combination with other types and 8 axes according to which evaluation approaches can be classified.

11. REFERENCES

[1] "ACM Computing Classification System." Association for Computing Machinery, 2012

[2] K.D. Bailey, Typologies and taxonomies: an introduction to classification techniques, Sage, Thousand Oaks, CA, 1994.

[3] V. R. Basili, G. Caldiera and H. D. Rombach, "Goal Question Metric Paradigm," Encyclopedia of Software Engineering, John Wiley & Sons, pp. 528-532, 1994.

[4] N. Breslow, "A generalized Kruskal-Wallis test for comparing K samples subject to unequal patterns of censorship", Biometrika, vol. 57, no. 3, pp. 579-594, 1970.

[5] L.C. Briand, S. Morasca, and V.R. Basili, "Property-Based Software Engineering Measurement," IEEE Trans. Software Eng., vol. 22, no. 1, pp. 68-86, Jan. 1996.

[6] R.L. Glass, I. Vessey, V. Ramesh, "Research in software engineering: an analysis of the literature," Information and Software Technology, vol. 44, no. 8, pp. 491–506, 2002.

[7] A. Gómez-Pérez, "Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases," Proc. Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'99), Alberta, Canada, pp. 1-18, October 1999.

[8] T. C. Hales, "Formal Proof," Notices of the ACM, vol. 55, no. 11, pp. 1370-1380, December 2008.

[9] J. L. Henessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann, 1990.

[10] R. W. Hockney, The Science of Computer Benchmarking, Society for Industrial & Applied Mathematics, 1996.

[11] R. Nickerson, J. Muntermann, U. Varshney, and H. Isaac, "Taxonomy development in information systems: developing a taxonomy of mobile applications," in European Conference in Information Systems, 2009

[12] D. L. Olson and D. Delen, Advanced Data Mining Techniques, Springer, Berlin, 2008.

[13] P. Rich, "The Organizational Taxonomy: Definition and Design," Acad. Manage. Rev., vol. 17, no. 4, p. 758, Oct. 1992

[14] N.E. Savin, "A test of the Monte Carlo hypothesis: Comment," Economic Inquiry, vol. 15 no. 4, pp. 613-617, 1977.

[15] M. Shaw, "What makes good research in software engineering?," Int. J. Softw. Tools Technol. Transf., vol. 4, no. 1, pp. 1–7, Oct. 2002.

[16] K. Siau and M. Rossi, "Evaluation techniques for systems analysis and design modelling methods - a review and comparative analysis," Information Systems Journal, vol. 21, issue 3, pp. 249-268, May 2011.

[17] R. R. Sokal, "Classification: Purposes, Principles, Progress, Prospects," Science, Vol. 185, No. 4157, pp. 1115-1123, Sep. 1974.

[18] D. Steininger, M. Trenz, and D. Veit, "Building Taxonomies in IS and Management – A Systematic Approach Based on Content Analysis," Wirtsch. Proc. 2013, Jan. 2013

[19] I. Vessey, V. Ramesh, R. L. Glass, "A unified classification system for research in the computing disciplines," Information and Software Technology, vol. 4, no. 47, pp. 245–255, 2005.

[20] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén, Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, 2000.