Towards a Controlled Vocabulary on Software Engineering Education

Sebastián Pizard[a]* and Diego Vallespir[a]

[a] *Engineering School - Universidad de la República (UdelaR), Julio Herrera y Reissig 565, Montevideo, Uruguay*

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

Software engineering is the discipline that develops all the aspects of the production of software. Although there are guidelines about what topics to include in a software engineering curricula, it is usually unclear which are the best methods to teach them. In any science discipline the construction of a classification schema is a common approach to understand a thematic area. This study examines previous publications in software engineering education to obtain a first controlled vocabulary (a more formal definition of a classification schema) in the field. Publications from 1988 to 2014 were collected and processed using automatic clustering techniques and the outcomes were analyzed manually. The result is an initial controlled vocabulary with a taxonomy form with 43 concepts that were identified as the most used in the research publications. We present the classification of the concepts in three facets: 'what to teach', 'how to teach' and 'where to teach' and the evolution of concepts over time.

Keywords: software engineering education controlled vocabulary, software engineering education topics, controlled vocabulary development

1. Introduction

Software engineering is a discipline within engineering that develops all the aspects related to the production of software, from the specification of software to its maintenance after it starts to be used (Sommerville 2010). More formally, IEEE (2010)defines software engineering as "the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software".

We believe there are certain characteristics that make software engineering education a particularly complex task.

- Brooks (1987) points out some aspects that are intrinsic to the nature of software make it a product of difficult construction and handling. According to Brooks, software has a certain superior complexity (due to its size) to other human constructions, this is due to a great extent to the fact that: "no two parts are alike (at least above the statement level)" (Brooks 1987, 11). This makes understanding and technical communications difficult, and also their management (for example estimating the necessary effort it takes to construct a software component given its specification). On the other hand, unlike other human constructions, software suffers a lot of pressure for its modification once it has been constructed, "in part it is because software can be changed more easily–it is pure thought-stuff, infinitely malleable." (Brooks 1987, 12). The intangibil-

*Corresponding author. Email: spizard@fing.edu.uy ity of software is another aspect to be taken into account. Although this characteristic makes it possible to create really creative solutions, it also makes it difficult to represent it and make agreements previous to its construction.

- The software engineer must be skillful and knowledgeable in a wide range of fields. Although it is clear he must have deep knowledge of the computer science that enables him to create and handle software that adapts to his professional needs, he must also master other areas such as: communication abilities, problem-solving skills, design methodologies, negotiation, interaction human-computer, leadership, ethics, among others.(Freeman, Wasserman and Fairley 1976; Lethbridge 2000; Bourque and Fairley 2014). A significant aspect in the management of human resources is the specific type of work performed by software engineers that some authors define as creative intellectual work and that poses several additional difficulties (DeMarco and Lister 2013).
- Studies that investigate the expectations of the industry reveal that software engineers are not always prepared for their professional activity as regards the set of abilities needed and the necessary depth (Radermacher, Walia and Knudson 2014). This may be due to the fact that certain practices and abilities of the area are difficult to teach in the academic environment; because, for example, of problems of scale (Wohlin 1997). Another important factor is the extreme speed in which technology changes; new programming tools appear almost daily and new methodologies appear several times a year (Jones 2010).

Recent studies look into the bibliography in search for empirical answers to these uncertainties. In order to do this they use the techniques of systematic reviews of literature and mapping studies (Kitchenham and Charters 2007). In some cases they attempt to collect articles on a specific area; certain authors gather initiatives related to the curriculum of software engineering (Qadir and Usman 2011), whereas others research which sources and countries published during 2010 on software engineering education and classify the articles according to a very restricted list of topics (Malik and Zafar 2012). In other cases, the research questions are much more specific; some authors study the use of practical experiences in software engineering education during 2013 (Marques, Quispe and Ochoa 2014), others investigate the use of open source projects in software engineering education (Nascimento et al. 2013).

The accumulation of evidence in the area of software engineering in general is an emerging activity that still poses certain difficulties, among which is the lack of consensus concerning terminology. Kitchenham et al. (2015) sum up the current situation: "Software engineering lacks strong taxonomies. The terms that we use are often imprecise, and software engineers are rather prone to create new terms to describe ideas that may well be closely related to existing ones. This can complicate searching since we need to consider all possible forms of terminology that might have been used in the titles and abstracts of papers."

An example of the construction of taxonomies for the improvement of research is the one provided by health psychologists (Michie et al. 2015). In a 3-year project with hundreds of collaborators from different countries they have developed and evaluated a taxonomy of techniques of behavior change (used, for example, to change the habits of drug or alcohol use). According to the researchers who led the work, the interventions with behavior change techniques were reported using different labels and many times the same label was used for different techniques. The lack of standard definitions means that each researcher who carries out systematic reviews must develop his own classification systems and in many cases this leads to a duplication of efforts and weakens the possibility of gathering evidence between reviews. As a result, they obtained a taxonomy achieved due to the consensus of the researchers that covers 93 behavior change techniques with clear labels, definitions and examples.

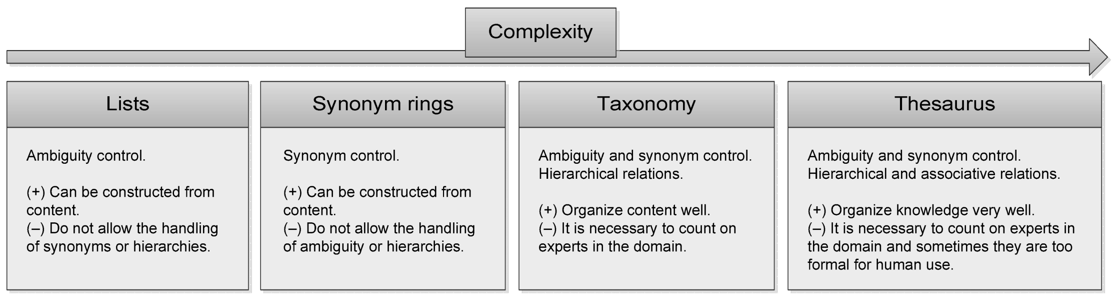
Complexity
Lists
Ambiguity control.
(+) Can be constructed from content.
(–) Do not allow the handling of synonyms or hierarchies.
Synonym rings
Synonym control.
(+) Can be constructed from content.
(–) Do not allow the handling of ambiguity or hierarchies.
Taxonomy
Ambiguity and synonym control. Hierarchical relations.
(+) Organize content well.
(–) It is necessary to count on experts in the domain.
Thesaurus
Ambiguity and synonym control. Hierarchical and associative relations.
(+) Organize knowledge very well.
(–) It is necessary to count on experts in the domain and sometimes they are too formal for human use.

Figure 1. Increase in the structural complexity in the controlled vocabularies.

The aim of our work is the creation of a controlled vocabulary on software engineering education. A controlled vocabulary is a set of previously selected terms subject to a certain control for its modification. These terms are used to describe documents or other types of content objects, thus making it possible to organize the material according to the elements that have been chosen to describe it (NISO 2005).The construction of a controlled vocabulary is another possible and complementary approach to understand a topical area. They make it possible to organize the knowledge considered in a thematic area by means of a unified list or categorization of the topics it includes.

This article is organized as follows. Section 2 introduces the topic of controlled vocabularies. In section 3 there is a summary of the method of work for the construction of the controlled vocabulary. Then, in section 4 the main results are presented and discussed. In section 5 the limitations of the work are described. Finally, in section 6 we present the conclusions and possible future work.

2. Controlled Vocabularies

The purpose of controlled vocabularies is to provide a way of organizing information. As noted by NISO (2005): "through the process of assigning terms selected from controlled vocabularies to describe documents and other types of content objects, the materials are organized according to the various elements that have been chosen to describe them". A content object is any element that has to be described for its inclusion in an information retrieval system, website, or another information source. Typical content objects are articles from scientific journals, technical reports and other types of documents. In practice, a controlled vocabulary consists of a list of terms that have been listed explicitly to represent the concepts that have been chosen to describe content objects. We should differentiate the notions of concept and term. While a concept is a thing, an idea or a shared understanding of something, a term is a label for a concept, in general the most common, a generic designation. All the terms of the controlled vocabulary must have an unambiguous non-redundant definition (NISO 2005). It is said that it is controlled because only the terms in the list should be used to refer to the concepts of the thematic area covered by the vocabulary. It is also controlled, since the modification of the list is subject to previously defined policies. The aim of a controlled vocabulary is to ensure consistency in the application of the language (Hedden 2010). An example of controlled vocabulary is the international standard *ISO/IEC/IEEE 24765:2010 - Systems and software engineering - Vocabulary*, which includes terms used in systems and software engineering (IEEE 2010). There are several types of controlled vocabularies that contemplate different purposes and different structural complexities . In Figure 1 the different types of controlled vocabularies and their characteristics are shown (NISO 2005).

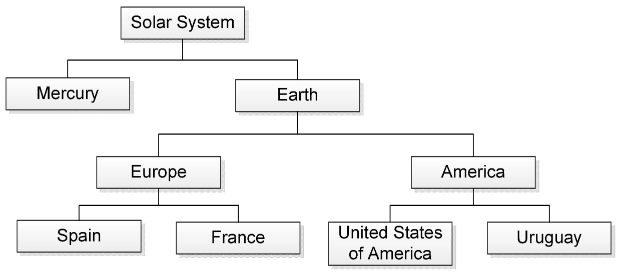

Figure 2. Example of Taxonomy.

In its simplest form a controlled vocabulary is made up only of a list of terms (each one associated to a different concept). The only important thing in the list is to eliminate ambiguity and in order to do so only one term is defined to refer to one concept. Different terms describe different concepts. Examples include geography (i.e. continents, countries, cities, etc.) or languages (e.g. English, Spanish, etc.).

Although the synonym rings are considered a type of controlled vocabulary, they are used differently from the rest. They are not used to classify content objects, but for their retrieval. A synonym ring is a set of terms considered equivalent. A synonym ring allows the users to access all the content objects that contain any of these terms. An example is the following set of terms: speech disorders, speech defects, disorders of speech, defective speech.

A taxonomy is a controlled vocabulary that is made up of preferred terms connected in a hierarchy or a poli-hierarchy. It is very important to identify the hierarchy relations between the terms and indicate them explicitly. Figure 2 presents an example of a controlled vocabulary with this type of structure.

Finally, a thesaurus is a controlled vocabulary organized according to a known order structured to make it possible to show clearly all the types of relations that need to be modeled. This means that apart from hierarchy relations it is possible to represent any other type of relations between terms (e.g. 'related to', 'use', 'within', etc.).

Three different approaches are suggested for the construction of a controlled vocabulary (NISO 2005).

- The Committee approach - This approach implies that experts in the area draw a list with the key terms indicating their relations. Lists of terms may be submitted by various experts or taken from various sources. The committee approach has two basic methods:
  - Top down First the wider terms are identified and the more concrete terms are selected according to the level of specificity chosen. The hierarchical structure is created during the work process.
  - Bottom up The same as in the previous case, the hierarchical structure is built during the work process, but it starts with the more specific terms first, and then with the wider ones.

  This approach presents as a great advantage that the construction of the vocabulary is done by the experts in the area, who will generally be the future users. An example of the application of this approach is the one presented above on the taxonomy of behavior change techniques (Michie et al. 2015). The authors performed a wide set of consensus exercises using the approach of the experts, first through Delphi methods (Linstone et al. 1975) for the collection of techniques, then applying top-down and bottom-up methods for the development of the hierarchy of the taxonomy (Cane et al. 2015) and finally carrying out activities together with the experts to evaluate the resulting taxonomy.

- The Empirical Approach - This approach contemplates basically two methods.
  - Deductive Method - The terms are extracted from a reference set of content objects, but there is no attempt to control or determine possible relations until a sufficient number of terms is collected. Then the terms are revised, the possible hierarchies are identified and the remaining terms are assigned to each one of them. The control of the vocabulary is done from there.
  - Inductive Method - New terms are selected for their possible inclusion in the vocabulary as they are found in a reference set of content objects. The control of the vocabulary is applied from the beginning as a continuous operation. If the controlled vocabulary has some kind of hierarchy, each term is assigned to the corresponding classes as soon as its inclusion is decided.
- Combination of Methods - In practice, more than one approach can be employed in one stage or another of the construction of a controlled vocabulary. For example, the hierarchies and other relations between terms that were established in an inductive way can be examined later from a deductive point of view.

Controlled vocabularies (especially those with thousands of terms) are easier to use if they are organized in a form different from the hierarchical one). The use of a facet analysis is similar to the construction of a controlled vocabulary in various subsets, in which every subset is in itself a controlled vocabulary that represents a different aspect of information (Hedden 2010). This allows you to assign multiple terms to classify a content object; for example, the following facets could be used to classify a book: format (paper, electronic, etc.), country where it was published (Uruguay, United States, etc.), topic (science fiction, historical novel, etc.). From a practical point of view, this type of analysis involves identifying the wider concept categories and selecting them as facets. Facet analysis is particularly useful for (NISO 2005):

- New areas of knowledge in which the knowledge of the domain is incomplete or the relations between the concepts are unknown.
- Interdisciplinary areas in which there is more than one perspective from which to consider a content object.
- Vocabularies for which multiple hierarchies are needed, but might be inadequate due to the difficulty to define the borders clearly.

Having a controlled vocabulary for the area of research in software engineering education would be extremely beneficial. A major consistency in the use of the language would present several advantages:

- Have more precise and unambiguous keywords, thus facilitating the indexation and retrieval of scientific articles in the search engines (Hedden 2010; Michie et al. 2015). This is currently extremely cumbersome due to the fact that controlled vocabularies do not have an adequate level of detail; for example in the taxonomy of terms of the IEEE (IEEE 2014) the term ‘software engineering education’ does not exist.
- Make comparisons (Hedden 2010). Once the ambiguity effect is eliminated it is possible to compare research that deals with the same subject, but using different terms.
- Allow a better use of the formal techniques of literature review, that is to say, systematic reviews of literature and mapping studies (Kitchenham et al. 2015). It would be possible to make better classifications based on the terms of the controlled vocabulary.
- Contribute in the construction of a body of knowledge on software engineering education.

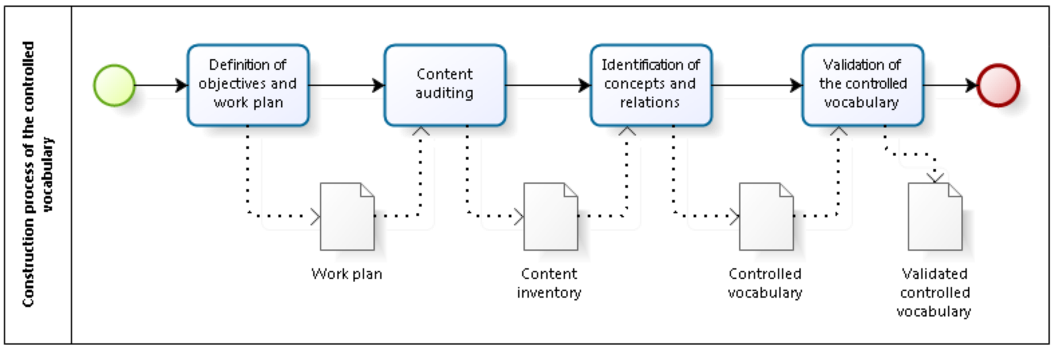
Construction process of the controlled vocabulary
Definition of objectives and work plan
Content auditing
Identification of concepts and relations
Validation of the controlled vocabulary
Work plan
Content inventory
Controlled vocabulary
Validated controlled vocabulary

Figure 3. Process for the construction of the controlled vocabulary, adapted from Z39.19-2005 (NISO 2005) and Hedden (2010).

3. Method for building the controlled vocabulary

Figure 3 shows the process used in this work for the creation of the controlled vocabulary; we base our work on the guide Z39.19-2005 provided by ANSI/NISO (NISO 2005) and the recommendations of Hedden (Hedden 2010). The process has 4 well defined stages. In the first stage the objectives of the controlled vocabulary to be constructed are defined and a plan to make it is established. The plan must include characteristics of the vocabulary (intended use, profile of the users and scope or content to consider), limitations or restrictions of the construction process and it must establish two very important aspects: the structure the controlled vocabulary will have and the approach for its construction. In the stage of vocabulary gathering all the vocabulary and possible knowledge on a thematic area to be covered by the controlled vocabulary are gathered. In order to do this, one or more research activities are carried out, like interviews or analysis of the material to be classified. In the third stage, Identification of concepts and relationships the list of possible concepts obtained in the previous phase is reviewed and the ones suitable for the controlled vocabulary are incorporated. Finally, in the stage of Validation of the controlled vocabulary the elements that indicate they are suitable for use are collected. In the rest of the section the execution details of this process for the construction of an initial controlled vocabulary on software engineering education are presented.

3.1. *Definition of the objectives and work plan*

Below we present the three main aspects of our work plan: characteristics of the vocabulary, proposed structure and selected approach.

The purpose of the controlled vocabulary is the classification of the research related to software engineering education. It must allow the classification of the existing research content (in general articles or sections from books), and its structure must support its evolution to accompany research to be carried out in the future. As far as the scope is concerned, initially the vocabulary will be used to classify scientific articles though its use in other types of research content objects is not dismissed.

The selection of the structure depends on the characteristics of the vocabulary but also of the resources available. Even given the difficulties present at the moment of defining hierarchical relations without the support of experts we choose to construct a first version of the controlled vocabulary with a taxonomy structure. In order to identify the hierarchical relations we will use facet analysis.

Since there are no previous controlled vocabularies on research in software engineering education, and it is not possible to rely on the effort of experts in the area, we believe it is convenient and feasible to use an empirical construction approach with a deductive method. This approach, as we explained in section 2, involves obtaining concepts from concept objects and their subsequent organization according to the chosen structure of the controlled vocabulary.

3.2. *Content auditing*

It is our intention that the concepts be good classifiers of the content published or to be published; that is why, the material published in the Conference on Software Engineering Education and Training (CSEET) - period 1988-2014 and in the Software Engineering Education and Training Track (SEET) of the International Conference on Software Engineering (ICSE) - period 2000-2014 is taken into account for the identification of concepts. These conferences have been selected since they are, in our opinion, the most recognized in the area and also because there are no specific scientific journals for the education of software engineering. All the years available are included for both, that is to say, all the years of the CSEET and all the years in which articles were published under the SEET.

It is clear that the literature selected as the basis for this study is relatively limited. There are other knowledge resources on software engineering education, of which two of the most important are text books and internet sources such as message forums and informal groups. We believe that this work is an initial step towards the construction of a taxonomy on software engineering education whose main objective is the classification of the research in the area. The selection of text books or forum messages for their analysis introduces important biases since not all of them can be taken into consideration. On the contrary, since there are only two conferences on the thematic area, this work covers an analysis of all their content. It is for this reason that we focus our effort on studying the research material described above.

The automatic cluster analysis technique (or clustering) is used for the extraction of terms taking as a basis the abstracts of the articles.

Generally speaking, text document clustering methods attempt to segregate the documents into groups where each group represents some topic that is different than those topics represented by the other groups (Hammouda and Kamel 2004; Frakes and Baeza-Yates 1992). Automatic clustering is a non-supervised learning method (this means it does not require human intervention) in which a set of clusters is obtained from a set of documents or texts (called corpus). In general, automatic clustering techniques are based on the recognition of syntactic patterns using the frequency of similar terms in the texts to discover similarities.

As a clustering tool we selected Carrot^2 (Weiss and Osinski 2015). Carrot^2 is an open source tool for cluster processing. It was created to process results of searches in the internet, but its characteristics make it a good tool to organize collections of small documents in thematic categories (Weiss 2006).

The corpus to be used in the process has as a characteristic the heterogeneity of the dates of publication of the documents (1988 to 2014). As the clustering tool does not use synonyms, it does not accumulate in the same cluster articles that use different terms to refer to the same concept. This may become an important bias since there are certain fashions in the use of the language which change after a certain number of years. To minimize this bias, the articles are processed in subsets taking the closeness of the publication date as a grouping criterion. The election of the subsets is arbitrary although there is an attempt to make sets of similar sizes and number of years. This additional step has the purpose of collecting vocabulary in different periods of history and enables, in some way, to extend the literary guarantee through time. Tennis (2002) states, concerning this, that it is necessary to keep a historical perspective, consequently, the resulting classification scheme must provide access to old conceptions of the universe

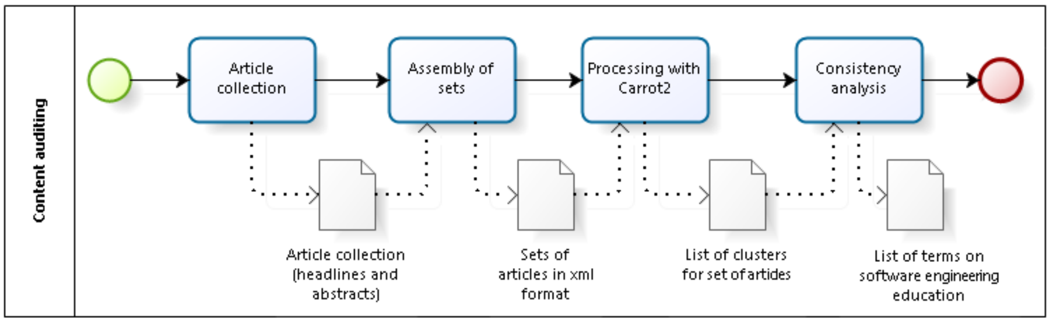
Content auditing
Article collection
Assembly of sets
Processing with Carrot2
Consistency analysis
Article collection (headlines and abstracts)
Sets of articles in xml format
List of clusters for set of articles
List of terms on software engineering education

Figure 4. Content auditing in detail.

of published material.

After obtaining the list of clusters for each period the consistency of each of the clusters found is studied manually. This involves reading the summaries of the articles of each cluster to determine if the concept is related to software engineering education or not. If it is, the term is added to the list of terms to be included in the controlled vocabulary. The steps taken in the content audit are illustrated in Figure 4.

The content audit by means of automatic clustering tool enables us to explore a wide corpus of research articles for the collection of concepts to be included in the controlled vocabulary. Although this technique saves a lot of effort, it has certain limitations that have to be taken into account. On the one hand, the clustering tool only considers the syntactic properties of the language used in the articles. In this way, a set of groups of documents is obtained that may or may not have semantic consistency. An analysis of the material conducted by experts should take into consideration, from the beginning, semantic properties, obtaining surely more relevant concepts and prioritizing the topical focus of the controlled vocabulary. On the other hand, the results of the automatic clustering analysis are biased by the algorithm used by the tool. In spite of these limitations we believe this type of work is very interesting to explore areas that have not been investigated in depth from the point of view of the organization of knowledge and controlled vocabularies.

3.3. *Identification of concepts and relations*

Finally, the list of terms that have been obtained is revised in order to identify concepts and establish preferred terms for each concept, as well as synonyms. In this first stage we do not look for hierarchical relations, although a first classification is done in the following facets that have been taken from the work of Nie and others (Nie, Ma and Nakamori 2007):

- Know-what (‘What to Teach’) - The concepts related to topics, skills or knowledge taught to the students are grouped here. Some examples are: ‘software design’, ‘communication and collaboration skills’, ‘software testing’ and ‘software project management’.
- Know-how (‘How to Teach’) - This facet considers concepts related to approaches, techniques or teaching methods. All the concepts that explain how to teach. Some examples are: ‘problem-based learning’ and ‘case studies’.
- Know-where (‘Where to Teach’) - The concepts related to the frame in which the concepts related to software engineering are taught are grouped here. This can include courses, workshops, secondary classes, etc. Some examples are: ‘software engineering course’, ‘master degree in software engineering’.

Table 1. Corpus of publications included in the construction of the controlled vocabulary.

Sets	Publications	Number of articles
2012-2014	CSEET years 2012 (27), 2013 (53) and 2014 (35). ICSE years 2012 (11), 2013 (13) and 2014 (19).	158
2009-2011	CSEET years 2009 (56), 2010 (29) and 2011 (78). ICSE year 2010 (7).	170
2006-2008	CSEET years 2006 (38), 2007 (48) and 2008 (36). ICSE years 2006 (13), 2007 (23) and 2008 (9).	157
2003-2005	CSEET years 2003 (50), 2004 (35) and 2005 (33). ICSE year 2005 (22).	139
2000-2002	CSEET years 2000 (46), 2001 (28) and 2002 (35). ICSE years 2000 (10) and 2001 (6).	125
1995-1999	CSEET years 1997 (21), 1998 (18) and 1999 (20). CSEE years 1995 (43) and 1996 (20).	122
1988-1994	CSEE years 1988 (18), 1989 (19), 1990 (17), 1991 (20), 1992 (34) and 1994 (44).	152
	Total number	1.023

These facets allow us to consider the publications from perspectives that order the concepts regardless of future hierarchical relations established between the concepts of the vocabulary.

3.4. *Validation of the controlled vocabulary*

The controlled vocabulary is at an initial version. We intend to expand some of the facets established as a result of this study before carrying out the formal validations. In spite of this, we proceed to explain the intended validation strategy.

Hedden recommends three complementary aspects for the validation of a taxonomy (Hedden 2010): the adaptation to the content to be classified (literary guarantee), the validation of the possible users of the taxonomy (guarantee of the users) and the needs of the organization that sponsors the development of the taxonomy (organizational guarantee). Given the nature of the construction process we understand that the vocabulary has some literary guarantee. In order to get the guarantee of users we will attempt to achieve a review of the controlled vocabulary by participants of international and regional conferences on software engineering education.

4. Results and discussion

Table 1 summarizes the numbers of publications used identifying their source and year of publication grouped in the study subsets. If any year is not indicated in the table for one of the publications it means that no published articles were found; particularly for the ICSE this does not mean that the conference was not held, but that there were no articles published in the education track - SEET.

A total of 1023 articles grouped in 7 different subsets were processed with the tool Carrot[2]. A list of clusters was obtained for each subset that were later reviewed manually. The manual review included reading all the headlines and summaries of the articles within each group (or cluster) to establish semantic consistency; namely the existence of one or more concepts on education in software engineering associated to the group. Thus 43 concepts were obtained that could all be grouped in the facets: ‘what to teach’, ‘how to teach’ and ‘where to teach’. For each of the concepts the different terms that are used in the publications to represent them were collected, and the most used one was designated as the preferred term, and the rest as synonyms. The initial controlled vocabulary with all its preferred terms is shown in Figure 5. Below we present and discuss the results we obtained.

4.1. *What to teach*

The concepts related to topics, skills or knowledge taught to the students are grouped in the facet 'what to teach'. Figure 6 shows the list of concepts of this facet (designated by their preferred term) and indicates for each concept the periods of time in which the cluster analysis detected it as one of the topics that grouped most publications.

The concepts associated to this facet consider: sub disciplines of software engineering ('software design', 'software maintenance', etc.), development approaches ('object-oriented programming and design', 'test-driven development', 'personal software process') and skills ('communication and collaboration skills', 'soft skills', 'technical skills'). It is possible to notice certain diversity that increases with time concerning the evolution. Previously the list included concepts related to sub disciplines (in particular 'software design', 'software maintenance', 'software process' and 'software project management') and development approaches, while currently concepts related to the skills to teach are also included. The concepts that have remained in more periods are the following: 'software design', 'software process', 'software process improvement', 'software project management' and 'software testing'.

We find it interesting to study the correspondence of the concepts obtained with some reference material of software engineering. We selected for this the last version of the SWEBOK guide (Bourque and Fairley 2014) and the result of the mapping is included in Table 2.

As can be seen, most of the concepts of the facet 'what to teach' have a corresponding concept in the SWEBOK (i.e., an knowledge area or subarea, a topic or subtopic or a concept within a subtopic). The topics that have no correspondence are: *'global software development'* that could be related to the topic *2.7.5 - Dealing with multicultural environments* but as the mapping is based on the identification of the same term or synonyms found cannot be considered as corresponding, very general concepts related to skills ('soft skills', 'technical skills'), very abstract concepts that later in the evolution of software engineering were covered in more concrete topics ('complex systems', 'real world problems') and some concepts related to a specific development approach ('personal software process'). Based on this, there is nothing to indicate that the SWEBOK cannot be used as a reference for a future expansion of the facet 'what to teach'. This expansion work requires a considerable effort, since it is necessary to analyze the whole text of the SWEBOK to carry out the activities of auditing the content and identifying concepts and relations (in a similar way to how they were explained in the points 3.2 and 3.3, although in this case the analysis is manual and it is not convenient to use automatic clustering tools). For this reason it is not presented as part of this work but as future work.

4.2. *How to teach*

The concepts related to techniques, approaches or ways of teaching are grouped in the facet 'how to teach'. Figure 7 shows the list of concepts grouped in this facet and their evolution over time.

This facet includes a great variety of topics that go from the collaboration between industry and academia to teaching materials. As regards its evolution, we notice that the number of concepts expands in the last periods to incorporate more teaching approaches

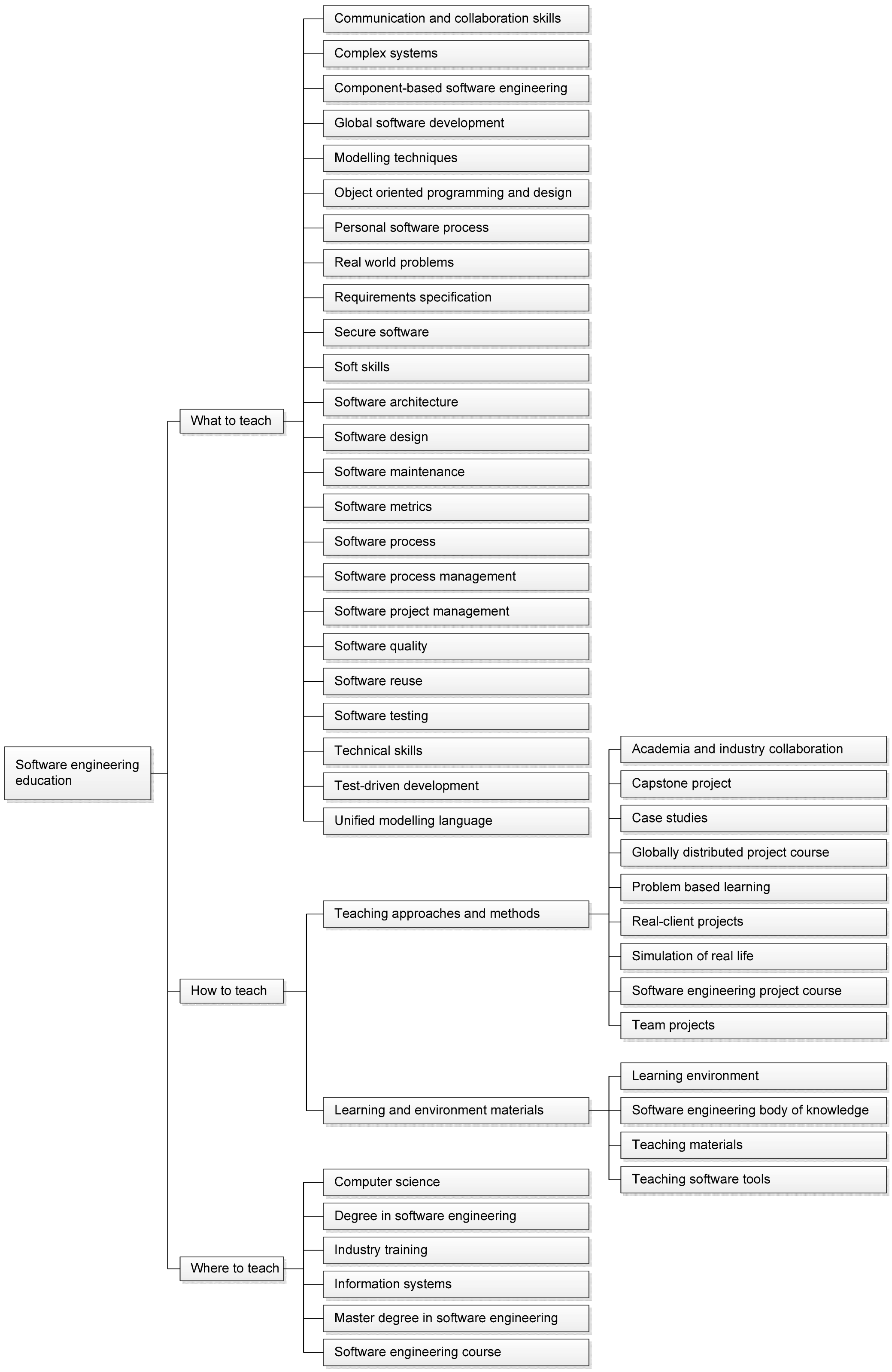
Software engineering education
What to teach
Communication and collaboration skills
Complex systems
Component-based software engineering
Global software development
Modelling techniques
Object oriented programming and design
Personal software process
Real world problems
Requirements specification
Secure software
Soft skills
Software architecture
Software design
Software maintenance
Software metrics
Software process
Software process management
Software project management
Software quality
Software reuse
Software testing
Technical skills
Test-driven development
Unified modelling language
How to teach
Teaching approaches and methods
Academia and industry collaboration
Capstone project
Case studies
Globally distributed project course
Problem based learning
Real-client projects
Simulation of real life
Software engineering project course
Team projects
Learning and environment materials
Learning environment
Software engineering body of knowledge
Teaching materials
Teaching software tools
Where to teach
Computer science
Degree in software engineering
Industry training
Information systems
Master degree in software engineering
Software engineering course

Figure 5. Initial software engineering education controlled vocabulary with all its preferred terms.

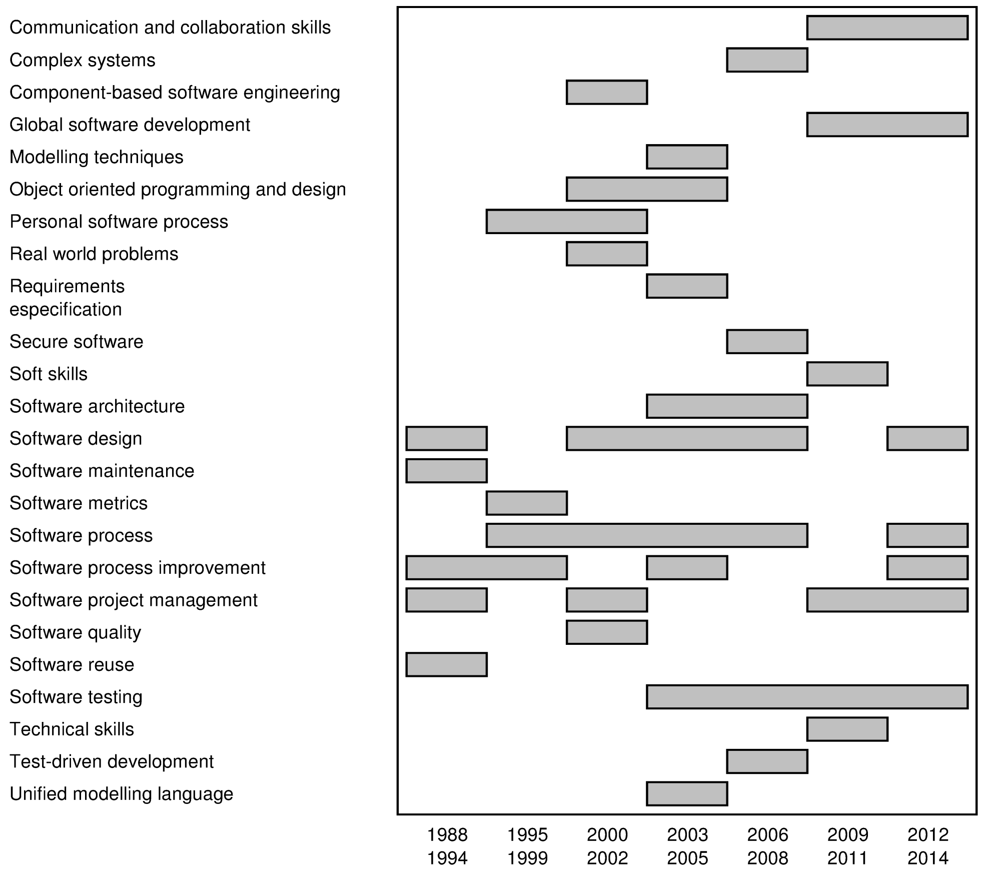

Figure 6. Evolution of the topics of the facet ‘what to teach’ over time.

related to different learning theories (‘case studies’, ‘problem-based learning’, ‘simulation of real-life’). Even so, there are two concepts that remain unchanged through almost all the periods, which are: ‘team projects’ and ‘software engineering project courses’. Both correspond to practical activities namely groups of students carrying out projects (in general development projects). The ‘software engineering project course’ involves a specific course made for that activity while ‘team projects’ might involve activities within another course or another subject (for example, using the name ‘team projects’ in a ‘software design course’).

It is difficult to find a basis for comparison for the facet ‘how to teach’. The closest might be the outline of teaching approaches presented by Nascimento and others (Nascimento et al. 2013), but it does not seem quite suitable since our facet does not refer only to teaching approaches. Realizing this, we decided to classify the concepts of this facet in two subsets: those associated to approaches or teaching methods and those connected to the teaching environment and materials (Figure 7 shows the concepts grouped according to this criterion). Table 3 presents the comparison of the subset of teaching approaches and the classification scheme of Nascimento.

Although there is not a good correspondence, it seems appropriate to divide the facet how-to-teach into two sub categories: ‘teaching approaches and methods’ and ‘learning environment and materials’; and in the future broaden the content of both of them in order to consider completely works such as that of Nascimento.

Table 2. Correspondence between the concepts of the facet ‘what to teach’ and the topics of the SWEBOK.

Concept of controlled vocabulary	**Topic or sub topic within the SWEBOK**
Communication and collaboration skills	11.3 - Communication skills
Complex systems	-
Component-based software engineering	2.7.5 - Component-based design (CBD)
Global software development	-
Modelling techniques	1.4.1 - Conceptual modelling [software requirements] 9 - Software engineering models and methods
Object oriented programming and design	2.7.3 - Object-oriented design 3.4.2 - Object-oriented runtime issues
Personal software process	-
Real world problems	-
Requirements specification	1.5 - Requirements specification
Secure software	13.17 - Secure software development and maintenance
Soft skills	-
Software architecture	2.3 - Software structure and architecture
Software design	2 - Software design
Software maintenance	5 - Software maintenance
Software metrics	1.7.5 - Measuring requirements 2.5.3 - Measures [software design] 3.2.3 - Construction measurement 4.4 - Test-related measures 5.2.4 - Software maintenance measurement 7.6 - Software engineering measurement 10.3.4 - Software quality measurement 15.3 - Measurement [engineering foundations]
Software process	8 - Software engineering process
Software process improvement	8.3 - Software process assessment and improvement
Software project management	7 - Software engineering management
Software quality	10 - Software quality
Software reuse	3.1.4 - Reuse
Software testing	4 - Software testing
Technical skills	-
Test-driven development	3.4.16 - Test-first programming
Unified modelling language	1.4.1 - Conceptual modelling [software requirements] 3.1.5 - Standards in construction 8.1 - Software process definition 9.2 - Types of models [software engineering models and methods]

Table 3. Correspondence between the concepts of the facet ‘how to teach’ related to teaching approaches and the classification scheme of Nascimento et al. (2013)

Concept of the controlled vocabulary	Category of the classification scheme of Nascimento
Academia and industry collaboration	-
Capstone project	-
Case studies	Case-based learning
Globally distributed project course	-
Problem-based learning	Problem-/Project-/Inquiry-based learning
Real-client project	-
Simulation of real-life (environment/problems)	-
Software engineering project course (project courses)	Peer/Group/Team learning
-	Active learning
-	Game-based learning
-	Studio-based learning

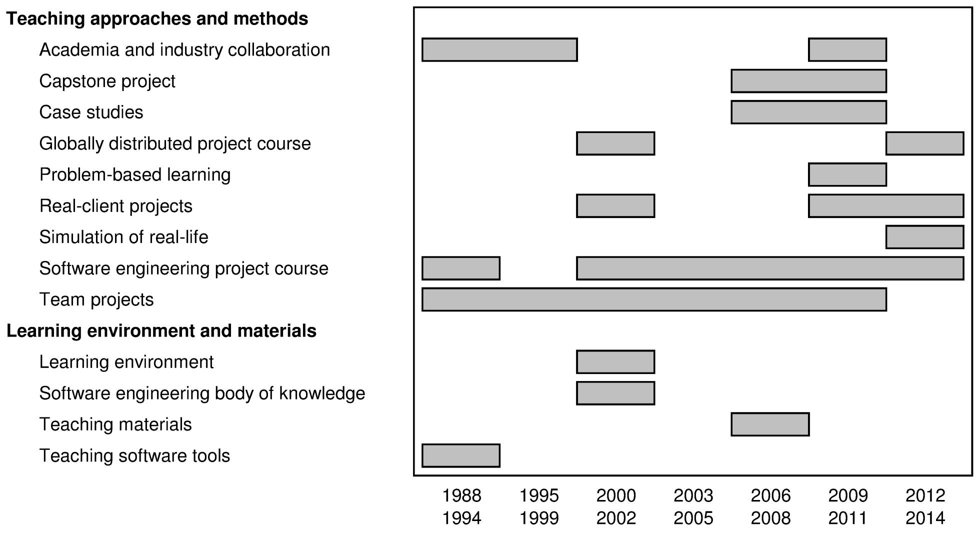

Figure 7. Evolution of the topics of the facet 'how to teach' over time.

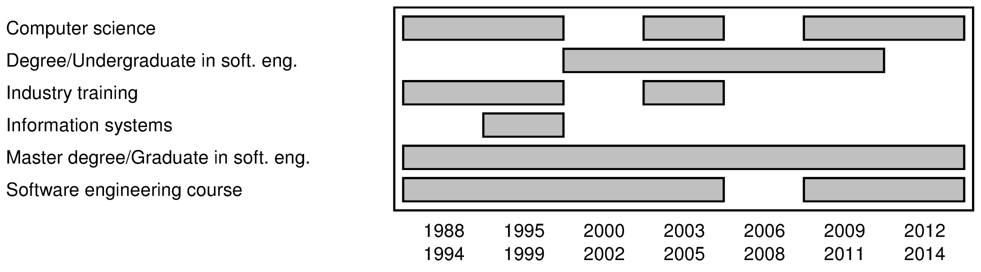

Figure 8. Evolution of the topics of the facet 'where to teach' over time.

4.3. *Where to teach*

The concepts related to the frame in which the concepts related to software engineering are taught are included in the facet 'where to teach'. This, in general could be a course, an academic career, training given to industrial personnel, etc. Figure 8 shows the list of concepts included in this facet and their evolution over time.

The following concepts are included in this facet:

- 'Computer Science': includes reports on the teaching of software engineering in the field of computer science, and this can refer to courses, departments, syllabuses or curricula of this discipline
- 'Degree/Undergraduate in Software Engineering': includes publications referred to the teaching of software engineering within the frame of a graduate course in that discipline.
- 'Industry Training': includes publications on the teaching of software in the field of industry, that is to say, workers or professionals of the industry. This in general involves specific training or tailor training.
- 'Information Systems': includes publications related to the teaching of software engineering in the field of information systems, and it can involve courses, departments, syllabuses or curricula of that discipline.
- 'Masters Degree/Graduate in Software Engineering': includes articles on the teaching of software engineering within a masters degree level in that discipline.
- 'Software Engineering Course': includes publications related to the teaching of software engineering or its sub topics in a software engineering course.

Regarding the evolution of these concepts, we can notice that more experiences of teaching software engineering in diverse contexts were published in the past, while at present more research on software engineering in specific contexts (degree, post-graduate degree or courses) is reported. This could show a greater formalization in the area and in the teaching of software engineering. It is also possible to see the importance research on software engineering education had directly in the industrial context ('industry training programs'), which nowadays has declined.

5. Limitations

The results of this study should be interpreted taking into account the following limitations: (a) only the headlines and summaries of the collected publications were studied, (b) the criterion used to group the articles to process them has been arbitrary, in spite of the attempt to obtain subsets of similar size and publication date, (c) the main aim of this work is to obtain an initial controlled vocabulary using the most used terms, and therefore, the result is not a comprehensive vocabulary; and (d) the body of the studied literature in this work is limited and it corresponds to a sample of research articles, since the objective of the created taxonomy is the classification of research of software engineering education. There are other sources used for the teaching of software engineering, like, for example text books and message forums in the web which have not been taken into consideration due to lack of resources and prioritizing.

6. Conclusions

This work presents the construction of a controlled vocabulary to provide support to the research on software engineering education. We show the results of applying automatic clustering techniques to the most relevant publications from 1988 to 2014 (1023 articles) to obtain a list of the most researched concepts from the study of the frequency of the terms in the abstracts.

The result is a controlled vocabulary that takes a taxonomic form with 43 concepts that were identified as the most used in the research publications in the period covered. The classification of the concepts is presented in three great facets: 'what to teach' (24 concepts), 'how to teach' (13 concepts) and 'where to teach' (6 concepts). The evolution of concepts over time is also shown over the years of publication.

To the best of our knowledge this vocabulary is the first one in the thematic area of software engineering education and we hope it will be of great value to the community. It can be used to create key words to be used when labeling articles, to understand a concept and go deeper consulting the suggested sources and it can also be used as the basis for future terminology standardization works in the area. We believe it is a vocabulary that must be considered initial or in the process of construction due to the number of concepts it includes.

The facet 'what to teach' groups concepts related to topics, skills or knowledge that are taught to the students. The concepts gathered consider: sub disciplines of software engineering ('software design', 'software maintenance', etc), development approaches ('object-oriented programming and design', 'test-driven development', 'personal software process'), and skills ('communication and collaboration skills', 'soft skills', 'technical skills'). We also detected an increase in the diversity of the topicsin the most recent years, mainly on the topics related to skills to be taught. We made a thorough comparison and concluded that there is nothing to indicate that the SWEBOK cannot be used as a reference to expand the facet.

The concepts related to approaches, techniques or teaching methods are grouped within the facet 'how to teach'. All the concepts that explain how to teach are included. As regards their evolution we notice that the number of concepts expands in the last years to incorporate more teaching approaches related to different teaching theories. We identify two concepts that remain through almost all the periods ('team projects' and 'software engineering project course'), both corresponding to practical activities in which groups of students work carrying out projects (in general development projects). Concerning this facet, this work includes a comparison with another previous work of Nascimento and others (Nascimento et al. 2013), and although the results obtained were not very good as far as coverage is concerned, it made it possible to notice the existence of two well differentiated sub categories: the first, 'teaching approaches and methods', and the second, 'learning environment and materials'

The facet 'where to teach' models the frame in which the research on software engineering education is reported. This includes courses or careers related to other disciplines (eg. 'computer sciences' or 'information systems') training to industry workers, as well as courses, careers or software engineering programs. The most relevant aspect is an increase in publications related to specific contexts of software engineering (eg. 'degree/undergraduate in software engineering', 'masters degree/graduate in software engineering' and 'software engineering course').

In future work we intend to continue working on the controlled vocabulary so as to expand it and validate it. As far as its expansion is concerned, it is possible to use the results of this work to achieve better results. In particular, it seems appropriate to use the content of the SWEBOK guide to expand facet 'what to teach'. On the other hand, facet 'how to teach' must be expanded to consider, at least, the most used teaching approaches. It is also important to make progress in the identification of more specific hierarchies between concepts.

The huge difference between what has been obtained and the previous literature on the concepts related to the aspect how to teach, seems to indicate the need to complement our work with analysis by experts, at least in categories without reference bibliography. On the other hand, recent studies on health psychology (Michie et al. 2015) present a very good precedent in the construction of a taxonomy by means of analysis by experts (or committee approach) the objective of which is to classify the research of a topical area. In this work the starting point were several existing taxonomies. We can assume then that our work can be seen as a preliminary step in the construction of a more complete taxonomy adequate to the use employing also the approach of the experts committee. Along this line, we believe that the results of our work provide a certain order to the terminology of the topical area and that it would be beneficial to count on the results of similar analyses so that later on ordering by experts could take place.

References

Bourque, P. and Fairley, R. E. 2014. *SWEBOK: Guide to the Software Engineering Body of Knowledge*, version 3.0 ed. Los Alamitos, CA: IEEE Computer Society Press, 2014. http://www.swebok.org/

Brooks, J., F. P. (1987). "No Silver Bullet Essence and Accidents of Software Engineering," *Computer*, 20(4), 10–19.

Cane, J., Richardson, M., Johnston, M., Ladha, R. and Michie, S. (2015). "From lists of behaviour change techniques (BCTs) to structured hierarchies: comparison of two methods of developing a hierarchy of BCTs." *British journal of health psychology*, 20(1), 130–150.

DeMarco, T., and Lister, T. (2013). *Peopleware: Productive Projects and Teams* (3rd ed.). Addison-Wesley Professional.

Frakes, W.B. and Baeza-Yates, R. 1992. *Information Retrieval: Data Structures and Algorithms.* Upper Saddle River, NJ, USA: Prentice Hall, Inc.

Freeman, P., Wasserman, A. I. and Fairley, R. E. (1976). “Essential Elements of Software Engineering Education.” *Proceedings of the 2Nd International Conference on Software Engineering*, 116–122.

Hammouda, K.M. and Kamel, M.S., Oct 2004. “Efficient phrase-based document indexing for Web document clustering,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 10, 1279–1296.

Hedden, H. 2010. *The Accidental Taxonomist.* Information Today Inc.

IEEE. 2010.“Systems and software engineering – vocabulary,” *ISO/IEC/IEEE 24765:2010(E)*, 1–418.

IEEE. 2014. “IEEE Taxonomy: A Subset Hierarchical Display of IEEE Thesaurus Terms.”

Jones, C. (2010). *Software Engineering Best Practices* New York, NY, USA: McGraw-Hill, Inc.

Kitchenham, B. and Charters, S. 2007. “Guidelines for performing Systematic Literature Reviews in Software Engineering,” Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001.

Kitchenham, B. A., Budgen, D. and Brereton, P. (2015). *Evidence-Based Software Engineering and Systematic Reviews* CRC Press.

Lethbridge, T. C. (2000). “What knowledge is important to a software professional?” *Computer*, (5), 44–50.

Linstone, H. A., and Turoff, M. (1975). *The Delphi method: techniques and applications* Addison-Wesley Pub. Co., Advanced Book Program.

Malik, B. and Zafar, S. 2012. “A Systematic Mapping Study on Software Engineering Education,” *Proceedings of World Academy of Science, Engineering and Technology*, 1982–1993.

Marques, M. R., Quispe, A. and Ochoa, S. F. 2014. “A systematic mapping study on practical approaches to teaching software engineering,” in *Frontiers in Education Conference (FIE)*, 1–8.

Michie S, Wood CE, Johnston M, Abraham C, Francis JJ, Hardeman W. (2015). “Behaviour change techniques: the development and evaluation of a taxonomic method for reporting and describing behaviour change interventions (a suite of five studies involving consensus methods, randomised controlled trials and analysis of qualitative data).” *Health Technol Assess* 19(99):1-188.

Nascimento, D., Cox, K., Almeida, T., Sampaio, W., Almeida Bittencourt, R., Souza, R., and Chavez, C. 2013. “Using Open Source Projects in software engineering education: A systematic mapping study,” in *Frontiers in Education Conference*, 1837–1843.

Nie, K. Ma, T. and Nakamori, Y. 2007. “Building a taxonomy for understanding knowledge management,” *Electronic Journal of Knowledge Management*, vol. 5, no. 4, 453–466.

NISO. 2005. “ANSI/NISO Z39.19-2005: Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies”.

Qadir, M. and Usman, M. 2011. “Software engineering curriculum: A systematic mapping study,” in *5th Malaysian Conference in Software Engineering (MySEC)*, 269–274.

Radermacher, A., Walia, G., and Knudson, D. (2014). “Investigating the Skill Gap Between Graduating Students and Industry Expectations.” Proceedings of the 36th International Conference on Software Engineering, 291–300.

Sommerville, I. 2010. *Software Engineering*, 9th ed. Harlow, England: Addison-Wesley.

Tennis,J. (2002) “Subject ontogeny: subject access through time and the dimensionality of classification.” *Advances in Knowledge Organization*, 54–59.

Weiss, D. 2006. “Descriptive Clustering as a Method for Exploring Text Collections,” Ph.D. dissertation, Poznań University of Technology, Poznań, Poland.

Weiss, D., Osinski, S. 2015. “Carrot2 - Open Source Search Results Clustering Engine.” http://project.carrot2.org/, v.3.10.4.

Wohlin, C. (1997). “Meeting the challenge of large-scale software development in an educational environment”, *Tenth Conference on Software Engineering Education and Training.*