Databases and Information Systems IX
G. Arnicans et al. (Eds.)

doi:10.3233/978-1-61499-714-6-313

Building of Concept System to Improve Systematic Collection of Terminology

Vineta ARNICANE [1], Guntis ARNICANS and Juris BORZOVS
Faculty of Computing, University of Latvia, Riga, Latvia

Abstract. Authors propose an experimental tool for analysis and graphical representation of glossaries. The original heuristic algorithms and analysis methods incorporated into the tool GlossToolset appeared to be useful to improve the quality of the glossaries. The GlossToolset generates concept system with various representations. Authors analyze a glossary *ISTQB Standard Glossary of Terms Used in Software Testing*. There are instances of problems found in ISTQB glossary related to its consistency, completeness, and correctness described in the paper.

Keywords. glossary, concept map, quality, graphs, heuristic algorithms, software testing

1. Introduction

There are two major types of terminology work. The first one is an ad hoc work on terminology, which deals with a single term or a limited number of terms. The second one is a systematic collection of terminology, which deals with all the terms in a specific subject field or domain of activity. Glossaries for specific domains is one of the most important results of the systematic collection of terminology. Glossaries are alphabetical lists of the terms in a particular domain of knowledge with the definitions for those terms. Glossaries contain the explanations of numerous concepts of the certain field. It is important that the quality of glossaries be sufficiently high. The authors of this paper provide their view on the problem of glossaries' quality and propose some methods how to reveal the issues in glossaries and how to address them.

The authors use *Standard Glossary of Terms Used in Software Testing Version 3.01* (May 27, 2015) [2] (further in the text - *Glossary*). The Glossary is produced by the *Glossary Working Group (GWG)* of the *International Software Testing Qualifications Board (ISTQB)* [3]. The Glossary accumulates terms and their explanations from the most significant sources in the software testing field. The contribution was made from testing communities throughout the world. GWG has compiled the Glossary from the other related glossaries taking into account opinion of the industry, commerce and government bodies and organizations. The Glossary's scope is a little broader than software testing domain - the authors of the Glossary explain "*Some related non-testing terms are also included*

[1]Corresponding Author: Faculty of Computing, University of Latvia, 19 Raina Blvd., Riga LV 1586, Latvia; E-mail: vineta.arnicane@lu.lv.

[2]http://www.istqb.org/downloads/category/20-istqb-glossary.html

[3]http://www.istqb.org/

if they play a major role in testing, such as terms used in software quality assurance and software lifecycle models. However, most terms of other software engineering disciplines are not covered in this document, even if they are used in various ISTQB syllabi.".

At the time of writing GWG has worked for ten years. The first useful version V1.3 was issued on May 31, 2007. During next years, Glossary was substantially improved both in the range and in the quality. The Glossary contains 652 preferred terms that appear as an entry, with the total 170 synonyms indicated. GWG considers that glossary is almost complete and now concentrates on the improvement of its quality.

Standard `ISO 704:2009 Terminology work - Principles and methods` [1] states that modelling concept systems and establishing representations of concept systems through concept diagrams are among main activities systematic terminology work. We consider that the Glossary is among the best the domain-oriented glossaries in the world. However, GWG did not follow principle stated in the standard ISO 704:2009 — create concept system represented by graphical diagrams at first. Taking into account that the Glossary has very good quality, an attempt to create automatically lightweight ontology or concept map for software testing domain was made [2], [3]. After analysis of results, the authors of this paper concluded that the Glossary has inconsistency issues.

During the localization and translation of the Glossary in Latvian, the authors realized that it is problematical to localize the terms of the whole domain of testing. There are difficulties to keep consistency both within the domain of testing and with the terms of related domains, for instance, software engineering, quality assurance, management and mathematics domains. It is hard to track down the consistency and mutual relationships between terms when there is such a plenitude of terms.

There is little research done in the field of the quality of glossaries, revealing the deficiencies and the elimination of them. Thus, the authors had to use the heuristic method — to learn, discover, and solve problems by experimental and especially by trial-and-error methods. The authors have developed a software that can generate various graphical representations for terms (concepts) and their interrelationships in given glossary. This chapter extends the work presented in [4] and describes some new results.

This paper is organized as follows. Section 2 describes the structure of glossaries, standards regarding terminology work and the quality evaluation of glossaries. Section 3 briefly explains GlossToolset developed by authors for analysis of glossaries. In Section 4 we analyze few problems revealed in ISTQB Software Testing Glossary. Section 5 summarizes paper and suggests issues for future research on glossary analysis.

2. Glossary Quality and Its Assessment

Glossary is a list of terms in a special subject, field, or area of usage, with accompanying definitions. From a perspective of automatic text analysis, the glossary is a semi-structured text document that contains descriptions of domain concepts and links among them. Some links are defined explicitly using keywords or text formatting means.

2.1. Glossary Structure

All records of Glossary that contain terms and definitions are arranged alphabetically. These records have various names: *entry*, *lemma*, *gloss*. We use term *entry* in this paper.

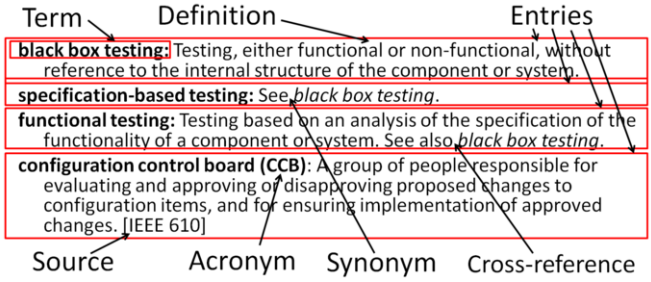

Figure 1. The structure of the ISTQB Glossary v.2.2. Source: [4]

defect management tool

See Also: incident management tool

Synonyms: bug tracking tool, defect tracking tool

A tool that facilitates the recording and status tracking of defects and changes. They often have workflow-oriented facilities to track and control the allocation, correction and re-testing of defects and provide reporting facilities.

Figure 2. An example of an entry from the ISTQB Glossary v.3.01.

Figure 1 shows important structure elements of the glossary such as *entry*, *term*, *definition*, *synonym*, *cross-reference*, *acronym*, and *source*. The other elements also exist, for instance, a *variant of definition* in case the term has many meanings, the context for which the definition is given.

Each glossary may have different rules/symbols that determine how the entry is composed, keywords that define links among terms, and formatting that can express another links or properties (see Figure 2). For instance, the glossary used for Figure 1 has an entry that contains term `specification-based testing` that is a synonym for the preferred term `black-box testing`, and has no information about synonyms for the preferred term. Otherwise, the glossary used for Figure 2 has terms that are preferred to other ones, in which case, the preferred term appears as an entry, with the synonyms indicated, but synonyms do not have their individual entry at all.

For doing the automatic analysis of a glossary, a common approach is to transform a glossary to a text format with convenient keywords or to a database that stores glossary elements as separate units.

Most of the glossaries usually have terms that correspond to a *noun*. In such case, entry is composited using pattern *X is a Y [that ...]*, i.e., a term named X (may be a phrase) is defined/explained by Y (may be a phrase). Usually, a description, how X differs from Y and other information that is useful to understand the meaning of X, is added. In linguistics, X is a *hyponym* and Y is a *hypernym*. A hyponym is a word or phrase whose semantic field is included within hypernym. In simpler terms, a hyponym shares a *type-of* relationship with its hypernym. In computer science, this relation is called *is-a* relationship. We prefer to use the later one. Few samples showed in the Figure 1 are "black-box testing *is-a* testing", "functional testing *is-a* testing", "configuration control board *is-a* group of people" or "defect management tool *is-a* tool" showed in the Figure 2.

Sometimes authors of glossary violate good practice of using the pattern *X is a Y*. For instance, entry in Figure 3 has no hypernym given in an explicit way. The reader

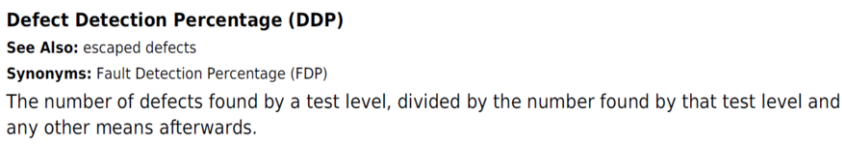

Figure 3. A hypernym is not given in an explicit way.

has to come to a conclusion himself that hypernym is *percentage* that is calculated as it is mentioned in the definition. In this case, the guessing is not difficult due to word *Percentage* used in the term. This observation is important – it is possible to establish *is-a* or *hyponym-hypernym* relationship from the term alone.

2.2. Standards

Few standards are created regarding terminology work. For instance, `ISO 704:2009 Terminology work - Principles and methods` [1] that is intended to standardize the essential elements of terminology work providing guiding principles; `ISO 1087-1:2000 Terminology work` [5] which the main purpose is to provide a systemic description of the concepts in the field of terminology and to clarify the use of the terms in this field.

Creators or maintainers of glossaries only partly are following standard recommendations. Standards are criticized, for instance, for *constructing a typology of concept relations* for terminology work [6]. Unfortunately, there are not better standards, and we have to try to exploit or adapt standards at hand, i.e., to use the most appropriate ideas, principles, and recommendations.

The standard `ISO 704: 2009` states: "The goal of terminology work ...is ... a clarification and standardization of concepts and terminology for communication between humans".

The main activities of terminology work are: 1) Identifying concepts and concept relations; 2) Analysing and modelling concept systems on the basis of identified concepts and concept relations; 3) Establishing representations of concept systems through concept diagrams; 4) Defining concepts; 5) Attributing designations (predominantly terms) to each concept in one or more languages; 6) Recording and presenting terminological data, principally in print and electronic media.

By evaluating available public glossaries authors of this paper have made a conclusion that many of the terminologists do not follow these recommendations. Authors of glossaries base on models created in their mind. The main problem is in a fact that models with many hundreds of concepts have inconsistencies, and different people have different models in their minds. As a result, the created ontology suffers from various inconsistencies. Let us pay attention to the second activity of terminology work "Analysing and modelling concept systems on the basis of identified concepts and concept relations" and the third activity "Establishing representations of concept systems through concept diagrams".

2.3. Quality Evaluation

Redman formulated a definition of data quality. Data quality is the degree to which data meet the specific needs of specific customers. Note that one customer may find data to

be of high quality for one usage of data, while another find the same data to be of low quality for another usage.

From the perspective of glossary user, the following quality attributes can be defined: structure of glossary, language correctness, glossary completeness regarding terms and relations, correctness of relations, unambiguity of definitions, understandability.

We focus on glossary completeness and correctness of relations, including consistency validation problem. Authors continue the further improvement of the tool that adopts the graph building algorithm from glossary entries [2], development of browsable concept map [3], and generating hierarchical diagrams for evaluating quality of glossary [4].

Since the Glossary is made without creating a concept system and its graphical representations before writing definitions, we propose to do reverse engineering – obtain the concept system for software testing domain from the Glossary. Thus, we can evaluate a quality of Glossary by analyzing the obtained concept system. The concept system can reveal the models that were in the minds of terminologists. We exploit our tool GlossToolset to generate various kinds of browsable concept maps with different notations and colors according to available information and analyzing goals.

3. GlossToolset

We have adopted our previously developed tools and improved them with better algorithms and functionality with a goal to use them for glossary analysis and quality evaluation. To refer to these tools easier, let us denote by *GlossToolset* a whole collection of our tools. The main goal of the GlossToolset is to generate various kinds of concept graphs or lightweight ontologies. Authors of [7] had a similar goal - to create an ontology from the `IEEE Standard Glossary of Software Engineering Terminology`, but there are not shown any pieces of evidence in the paper that all process is automatic.

Automatic concept map or ontology construction from document collections is an actual problem that is not fully solved yet. The review [8] surveys what is possible now and what are current research directions. Authors of [9], [10] and [11] solves similar problem as stated in this paper - how to extract semantic structure from glossaries automatically.

3.1. Parsing of Glossary

At the very beginning, one has to parse glossary and store into the database all terms, definitions, additional information and explicit relationships among terms. As a source of the glossary, one can take a text file obtained from initial glossary document or data (e.g., CSV files) received from glossary owner (for instance, the ISTQB Glossary is stored in the relational database).

The next step, according to works of Arnicans et al. [2] and [3], is to find *domain aspect names*. They introduce a new method for extracting the most significant words from a document in the form of a glossary. Then they assign a weight to each word in the glossary. The total weight of the word is a sum of the word weights in each entry. The weight of a word in a sentence depends on its position in the term and the definition. This

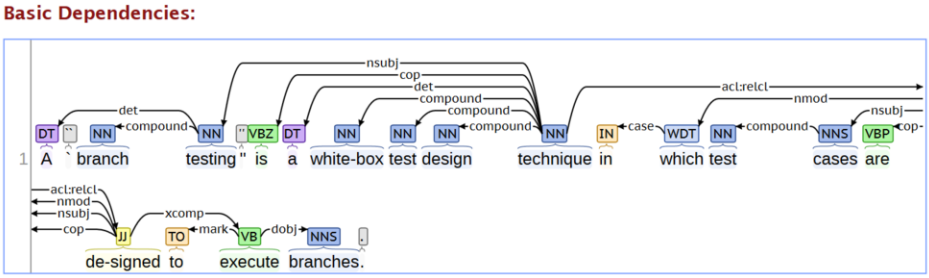

Figure 4. Basic Dependencies Tree obtained by CoreNLP from sentence taken from entry's definition. Source: [4]

new weighting method gives a better set of more *significant words* that characterizes the selected domain.

Authors of this paper improved weighting method by using natural language processing tool `Stanford CoreNLP` [12] that provides a set of natural language analysis tools. These tools can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract open-class relations between mentions, etc.

Our improved weighting method has the following steps (see details in [4]):

1. The GlossToolset transforms each entry into a sentence to get better results processing it with `Stanford CoreNLP`.
2. The CoreNLP processes each sentence, and all information is stored.
3. The GlossToolset analyze each obtained `Basic Dependencies tree` (see sample in Figure 4) and extract hypernyms (it is a `white-box test design technique` in the given sample).
4. The GlossToolset calculates total weight for each word in the glossary as a sum of weights of each word's instance. An instance of the word gets a weight calculated by formula 2^{-level} where the *level* is a level of word's instance in the `Basic Dependencies tree`.
5. The GlossToolset creates a list of most important words/concepts.

3.2. "explains" Graphs

The GlossToolset can create "explains" graphs and browsable concept map. The tool bases on principles described in [2] and [3].

First, the tool generates domain aspect graphs that can be considered as small concept maps. Authors of [13] also propose to create small concept maps to visualize domain. They take the important concept in focus and shows related concepts.

Second, all domain aspect graphs are merged into one large hypergraph. New concept maps are created on the fly by focusing any term; a subgraph from the hypergraph determine the structure of each concept map. The concept map helps to collect similar or related terms, immediately see definitions of terms and traverse through whole term graph (Figure 5). The relation can be defined as "concept X *explains* concept Y". For

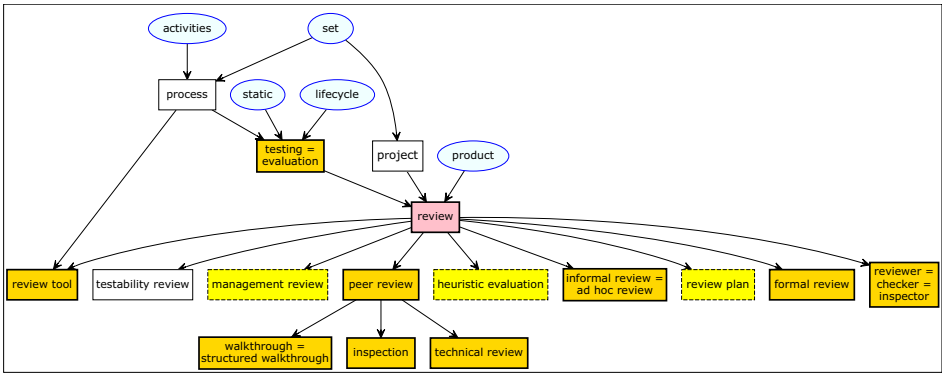

Figure 5. Concept map with relationship *explains* for the focused node review as a part of the hypergraph. Source: [4]

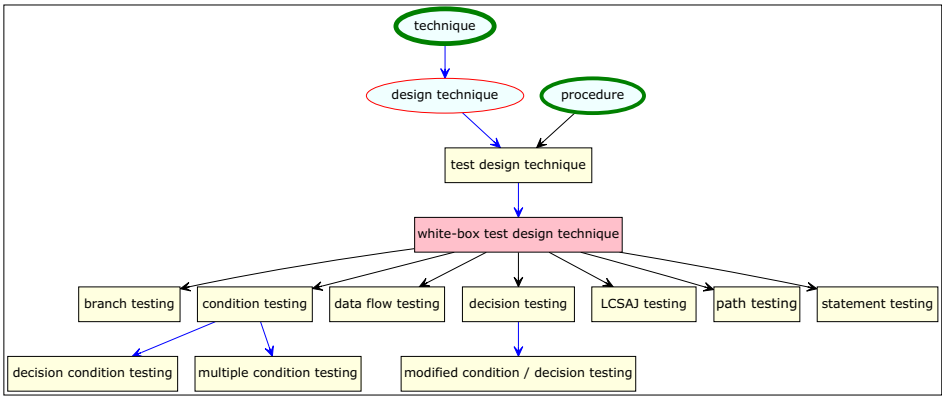

Figure 6. Concept map with relationship *is-a* for the focused node `white-box test design technique` as a part of the hypergraph. Source: [4]

instance, `peer review` explains `inspection`. ISTQB Syllabus level is shown using colors. Colors help to evaluate some quality aspects of syllabus too.

We have created graphs with different coloring principles, too. For instance, [14] used colors to classify terms in classes that follow Six Ws principle (Who, What, Where, When, Why, How).

3.3. "is-a" and "part-of" Graphs

"is-a" and "part-of" graphs show hierarchy among related concepts. Such kind of graphs is welcomed by `ISO 704:2009` standard. We introduce various sorts of relations according to the type of automatic concept recognition or creating of concepts and relations among them. For instance, new terms (in ellipses) that are no part of the Glossary are generated using our algorithms, terms in boxes are colored in order to expose their subdomain such as *pure testing*, *quality assurance*, *management*, *software engineering*, *mathematics* (see Figure 6 that shows only "is-a" relations).

Another graph sample is given in Figure 7. It is a fragment from the whole concept graph with relations "is-a" and "part-of" using UML notation for relations. To reduce

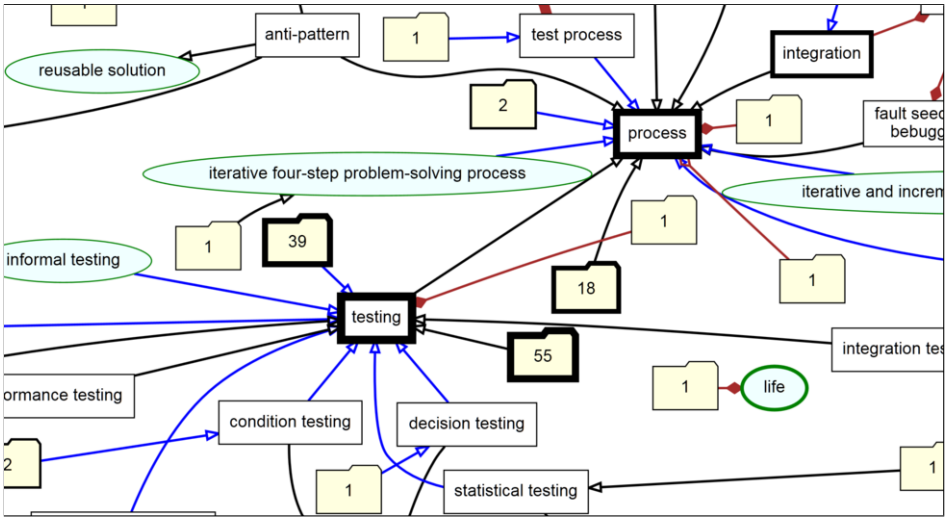

Figure 7. A fragment of the whole concept graph with relations "is-a" and "part-of" with collapsed concepts that are *leaves*.

the complexity of the graph, we collapse all subclass elements that are as leaves for their superclass element and display it as a folder with a number of collapsed elements (by clicking on the folder one can see all collapsed elements).

4. Analysis of ISTQB Software Testing Glossary

During our work in the localization of ISTQB Software Testing Glossary's terms in Latvian, we tried to preserve the level of Glossary's quality. In order to do that, we used GlossToolset, a set of our tools. As it later turned out, there are problems in the Glossary, too. In most cases there are inconsistencies, for instance, the same terms are used with different semantics or vice versa different terms are used for the same concept. Another problem is an inconsistent usage of hypernyms in definitions. Next subsections are devoted to a description of instances of problems found in Glossary using GlossToolset.

4.1. Inconsistent Usage of Hypernyms

Let us look at a group of terms related to concept `tool`. The graph is created by the GlossToolset, part of which is shown in Figure 8. Definitions of all terms included in the figure are given in Table 1. The terms from the Glossary are represented by boxes and significant additional concepts revealed by the GlossToolset are represented by ellipses.

There are only two terms `hyperlink test tool` and `test comparator` connected to `test tool` because their definitions contain phrase *test tool* as a hypernym.

The terms `security testing tool` and `stress testing tool` are located under node `testing tool` offered by the GlossToolset. If we look at definitions of these terms, it is noticeable, that all of them use only word *tool* as hypernym, not *test tool* or *testing tool*.

Definition of the term `test data preparation tool` is interesting because the hypernym in the definition is `type of test tool`. This is another approach that is

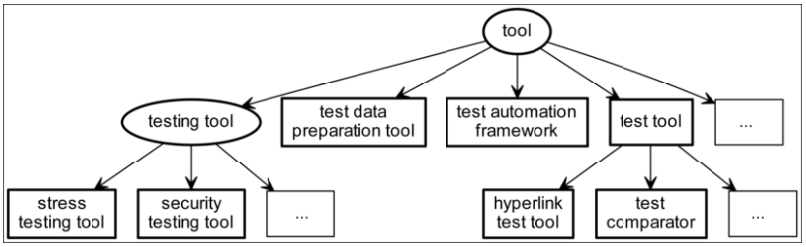

Figure 8. Part of the graph created for word *tool*. Source: [4]

Table 1. Definitions of some Glossary terms related to tools

Term	Definition
test tool	A software product that supports one or more test activities, such as planning and control, specification, building initial files and data, test execution and test analysis.
hyperlink test tool	A tool used to check that no broken hyperlinks are present on a web site.
test comparator	A test tool to perform automated test comparison of actual results with expected results.
security testing tool	A tool that provides support for testing security characteristics and vulnerabilities.
stress testing tool	A tool that supports stress testing.
test data preparation tool	A type of test tool that enables data to be selected from existing databases or created, generated, manipulated and edited for use in testing.
test automation framework	A tool that provides an environment for test automation. It usually includes a test harness and test libraries.

used in Glossary in order to describe tool. However, it requires asking, why, for instance, `test comparator` is a `test tool` and is not a `type of test tool`.

We conclude that the quality and comprehensibility of the Glossary could be better if the hypernyms of terms would be used in the same manner in the same context.

Sometimes definitions are formulated in such a way that the true hypernyms cannot be found automatically by the GlossToolset, but can be revealed by a domain expert. For instance, the definition of the term `test automation framework` explicitly shows semantics that the GlossToolset cannot recognize – this term belongs to the family of test tools.

GlossToolset generates graphs that demonstrate types of relationships among the terms by different colors and show the definitions of terms as tooltips for each node. This ability is very convenient for experts in order to notice consistency problems.

4.2. Inconsistent Usage of the Terms Test and Testing

According to the definitions provided by the Glossary showed in Table 2 the term `test` means a set of test cases while the term `testing` means process consisting of all activities that have to be done to test the software. At the same time, the definition of the term `test estimation` says that it is an "*approximation of a result related to various aspects of testing*". So, this term means the assessment not only of the set of test cases but also of process related aspects of the testing.

Table 2. Usage of the terms `test` and `testing` in the Glossary

Term	Definition
test	A set of one or more test cases.
testing	The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.
test estimation	The calculated approximation of a result related to various aspects of testing (e.g., effort spent, completion date, costs involved, number of test cases, etc.) which is usable even if input data may be incomplete, uncertain, or noisy.
test process	The fundamental test process comprises test planning and control, test analysis and design, test implementation and execution, evaluating exit criteria and reporting, and test closure activities.
test level	A group of test activities that are organized and managed together. A test level is linked to the responsibilities in a project. Examples of test levels are component test, integration test, system test and acceptance test.
component testing	The testing of individual software components.

There are more cases of confusing and inconsistent usage of terms `test` and `testing` in the Glossary. For instance, the term `test level` by its definition is *a group of activities*. Consequently, this term has to be titled as `testing level` instead of `test level` according to the definitions of the terms `testing` and `test`.

If we look at the end part of the definition of the term `test level` there are named examples of those levels — `component test`, `integration test`, `system test`, and `acceptance test`. There are no such terms in the Glossary. Glossary contains terms `component testing` whose definition is shown in Table 2, `integration testing`, `system testing`, and `acceptance testing`.

4.3. Different Terms with the Same Meaning

Let us look at the definitions of the terms `test process` and `testing` shown in Table 2. In point of fact both of them have the same meaning. It means that one of them can be supposed as a redundant term in the glossary, for instance, `test process`, or maybe they have to be declared as synonyms.

4.4. Different Meanings of the Same Term or Significant Word

Another case, when the automatic generation of concept map from glossary is very hard or impossible, is when terms have different meanings in different contexts. For instance, there are two different types of the significant word *framework* mentioned in the Glossary. In some definitions, it is used with the meaning as skeleton or outline of activities that should be done during some organizational process, for instance, `Capability Maturity Model Integration (CMMI)` or `TPI Next` or `SCRUM`, but sometimes *framework* means software tool, for instance, `test automation framework`, `unit test framework`. The definitions of `Capability Maturity Model Integration (CMMI)` and `unit test framework` as examples are shown in Table 3.

A similar situation is with terms `input`, `input value`, `specified input` whose definitions are shown in Table 4. The definition of `input` says, that `input` is a variable, and also `input value` has its definition as an instance of the `input`.

Table 3. Definitions of the terms `Capability Maturity Model Integration (CMMI)` and `unit test framework`

Term	Definition
Capability Maturity Model Integration (CMMI)	A framework that describes the key elements of an effective product development and maintenance process. The Capability Maturity Model Integration covers best-practices for planning, engineering and managing product development and maintenance.
unit test framework	A tool that provides an environment for unit or component testing in which a component can be tested in isolation or with suitable stubs and drivers. It also provides other support for the developer, such as debugging capabilities.

Table 4. Some definitions the Glossary terms related to concept `input`

Term	Definition
input	A variable (whether stored within a component or outside) that is read by a component.
input value	An instance of an input.
specified input	An input for which the specification predicts a result.

A `specified input` is a term used in detailed test cases which have described input values and specified expected output values or result. The term `specified input` does not mean variables as it can be supposed having the term `input` in the context. Such inconsistency leads to propose the term `specified input values` instead of `specified input`.

The Glosary contains also the terms `software lifecycle` and `test phase` provided in the Table 5. In definition of the term `software lifecycle` the term `test phase` is used with the aim to describe testing stage of lifecycle. At the same time from the term's `test phase` definition follows that `test phase` can be also a substage of testing stage of the software lifecycle.

4.5. Different Comprehension of Similar Terms

The definition of the term `test process` provided in the Table 2 says that it consists of different activities, for instance, `test planning`, `test control`, `test analysis`, `test design`, `test implementation`, `test execution`, `test closure`.

Let us look on the definitions shown in Table 6. The definition of the term `test planning` tells that `test planning` is *activity*. The definition of the term `test control` declares that `test control` is *test management task*. But the definitions of the terms `test analysis`, `test design`, `test implementation` and `test execution` asserts that `test analysis`, `test design`, and `test implementation` are *processes*. The definition of the term `test closure` on its turn states that `test closure` is *phase*.

At the same time the definition of the term `test tool` shown in Table 1 says that `test planning and control`, `specification`, `building initial files and data`, `test execution` and `test analysis` are *test activities*.

All mentioned definitions allow us to suggest that the words *process*, *task*, *activity* and *phase* are used as synonyms in the Glossary. Such situation can be misleading and confusing for the reader.

Table 5. Usage of the terms `software lifecycle` and some terms related to phases

Term	Definition
software lifecycle	The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software lifecycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase. Note these phases may overlap or be performed iteratively.
test phase	A distinct set of test activities collected into a manageable phase of a project, e.g., the execution activities of a test level.
requirements phase	The period of time in the software lifecycle during which the requirements for a software product are defined and documented.
test execution phase	The period of time in a software development lifecycle during which the components of a software product are executed, and the software product is evaluated to determine whether or not requirements have been satisfied.
reactive testing	Testing that dynamically responds to the system under test and test results being obtained. Typically reactive testing has a reduced planning cycle and the design and implementation test phases are not carried out until the test object is received.

Table 6. Terms describing the `test process` activities

Term	Definition
test planning	The activity of establishing or updating a test plan.
test control	A test management task that deals with developing and applying a set of corrective actions to get a test project on track when monitoring shows a deviation from what was planned.
test analysis	The process of analyzing the test basis and defining test objectives.
test design	The process of transforming general test objectives into tangible test conditions and test cases.
test implementation	The process of developing and prioritizing test procedures, creating test data and, optionally, preparing test harnesses and writing automated test scripts.
test execution	The process of running a test on the component or system under test, producing actual result(s).
test closure	During the test closure phase of a test process data is collected from completed activities to consolidate experience, testware, facts and numbers. The test closure phase consists of finalizing and archiving the testware and evaluating the test process, including preparation of a test evaluation report.

4.6. Contradictoriness of Term Definitions

The Glossary also has some contradictions in definitions of terms. For instance, the definition of the term `test phase` shown in Table 5 tells that `test phase` is a *set of activities collected into a manageable phase of a project* and as example provided the execution activities of a test level.

The reader can conclude that there can be `test execution phase` and it could be a set of activities. Nevertheless this is wrong conclusion because the Glossary contains the term `test execution phase` (also shown in Table 5) which by definition is *the period of time*.

There are terms `design and implementation test phases` mentioned in the definition of the term `reactive testing` (shown in the Table 5) but their definitions

also are not included in the Glossary. The reader of Glossary cannot be confident whether the `test design phase` and `test impelementation phase` are *the set* of activities or *the period of time* while these activities are realized.

4.7. Missing or Redundant Terms

There are more terms missing in the Glossary. Let us to look on the definition of the term `software lifecycle` provided in the Table 5 once more. It enumerates all phases of the lifecycle — *concept phase*, *requirements phase*, *design phase*, *implementation phase*, *test phase*, *installation and checkout phase*, *operation and maintenance phase* and *retirement phase*. The Glossary contains the term `requirements phase`, but it does not contain as terms all other mentioned lifecycle phases. It allows to think that either `requirement phase` is redundant term in Glossary or, on the other hand, may be these other terms are missing in the Glossary. Similarly as it is described in the subsection 4.4 in the case of the term `test phase` the lack of these terms also has led to contradictions in the Glossary.

5. Conclusions and Future Work

The GlossToolset developed by authors of the paper emerged as useful and handy tool when it is necessary to analyze the glossaries. There can be different aims of analysis, for instance, improvement of glossary's quality or attempt to develop the ontology of the concepts held by a glossary. The GlossToolset can graphically show the different relationships between glossary's terms, for instance, relation hyponym-hypernym or which terms or significant words (concepts) are used to explain or define given term. The graph generated by the GlossToolset substantially facilitates recognition of different type inconsistencies that are in glossary thus helping to eliminate them and improve the quality of glossary.

There are also problems in glossaries that are not recognizable using the GlossToolset or another tool. In such case, a contribution of an expert is necessary. For instance, following problems require expert decision 1) true hypernym is verbally hidden in the definition of the term; 2) the same term of the concept is used in different meanings; 3) there are two or more terms for the same concept in the glossary.

Complimentary material to the paper, containing top domain aspects, browsable concept maps, etc., is available on our expanding site[4].

Acknowledgments

This paper partially has been supported by the European Regional Development Fund Project No. 2DP/2.1.1.3.1/11/APIA/VIAA/010.

[4]Improvement of Systematic Collection of Terminology, http://science.df.lu.lv/aab16

References

[1] ISO 704:2009, Terminology work — Principles and methods. Geneva: International Standards Organization.

[2] G. Arnicans, D. Romans, and U. Straujums. Semi-automatic generation of a software testing lightweight ontology from a glossary based on the ONTO6 methodology. In et al. Caplinskas, Albertas, editor, *Frontiers in Artificial Intelligence and Applications. Databases and Information Systems VII*, volume 249, pages 263–276. IOS Press, 2013.

[3] G. Arnicans and U. Straujums. Innovations and advances in computing, informatics, systems sciences, networking and engineering. chapter Transformation of the Software Testing Glossary into a Browsable Concept Map, pages 349–356. Springer International Publishing, Cham, 2015.

[4] V. Arnicane, G. Arnicans, and J. Borzovs. Heuristic method to improve systematic collection of terminology. In G. Arnicans, V. Arnicane, J. Borzovs, and L. Niedrite, editors, *Databases and Information Systems: 12th International Baltic Conference, DB&IS 2016, Riga, Latvia, July 4-6, 2016, Proceedings*, pages 337–351. Springer International Publishing, Cham, 2016.

[5] ISO 1087-1:2000, Terminology work - Vocabulary - Part 1: Theory and application. Geneva: International Standards Organization.

[6] A. Nuopponen. Tangled web of concept relations. concept relations for iso 1087-1 and iso 704. In *Terminology and Knowledge Engineering 2014*, pages 10–p, 2014.

[7] J. R. Hilera, C. Pagés, J. J. Martínez, J. A. Gutiérrez, and L. De-Marcos. An evolutive process to convert glossaries into ontologies. *Information technology and libraries*, 29(4):195–204, 2010.

[8] O. Medelyan, I. H. Witten, A. Divoli, and J. Broekstra. Automatic construction of lexicons, taxonomies, ontologies, and other knowledge structures. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(4):257–279, 2013.

[9] P. Velardi, S. Faralli, and R. Navigli. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707, 2013.

[10] R. Navigli and P. Velardi. Ontology enrichment through automatic semantic annotation of on-line glossaries. In *Managing Knowledge in a World of Networks*, pages 126–140. Springer, 2006.

[11] R. Navigli and P. Velardi. From glossaries to ontologies: Extracting semantic structure from textual definitions. *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pages 71–87, 2008.

[12] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[13] S. Deokattey and K. Bhanumurthy. Domain visualisation using concept maps: A case study. *DESIDOC Journal of Library & Information Technology*, 33(4):295–299, 2013.

[14] I. Kuļešovs, V. Arnicane, G. Arnicans, and J. Borzovs. Inventory of testing ideas and structuring of testing terms. *Baltic J. Modern Computing*, 1(3–4):210–227, 2013.