ESF
EIROPAS SOCIĀLAIS FONDS

EIROPAS SAVIENĪBA

IEGULDĪJUMS TAVĀ NĀKOTNĒ

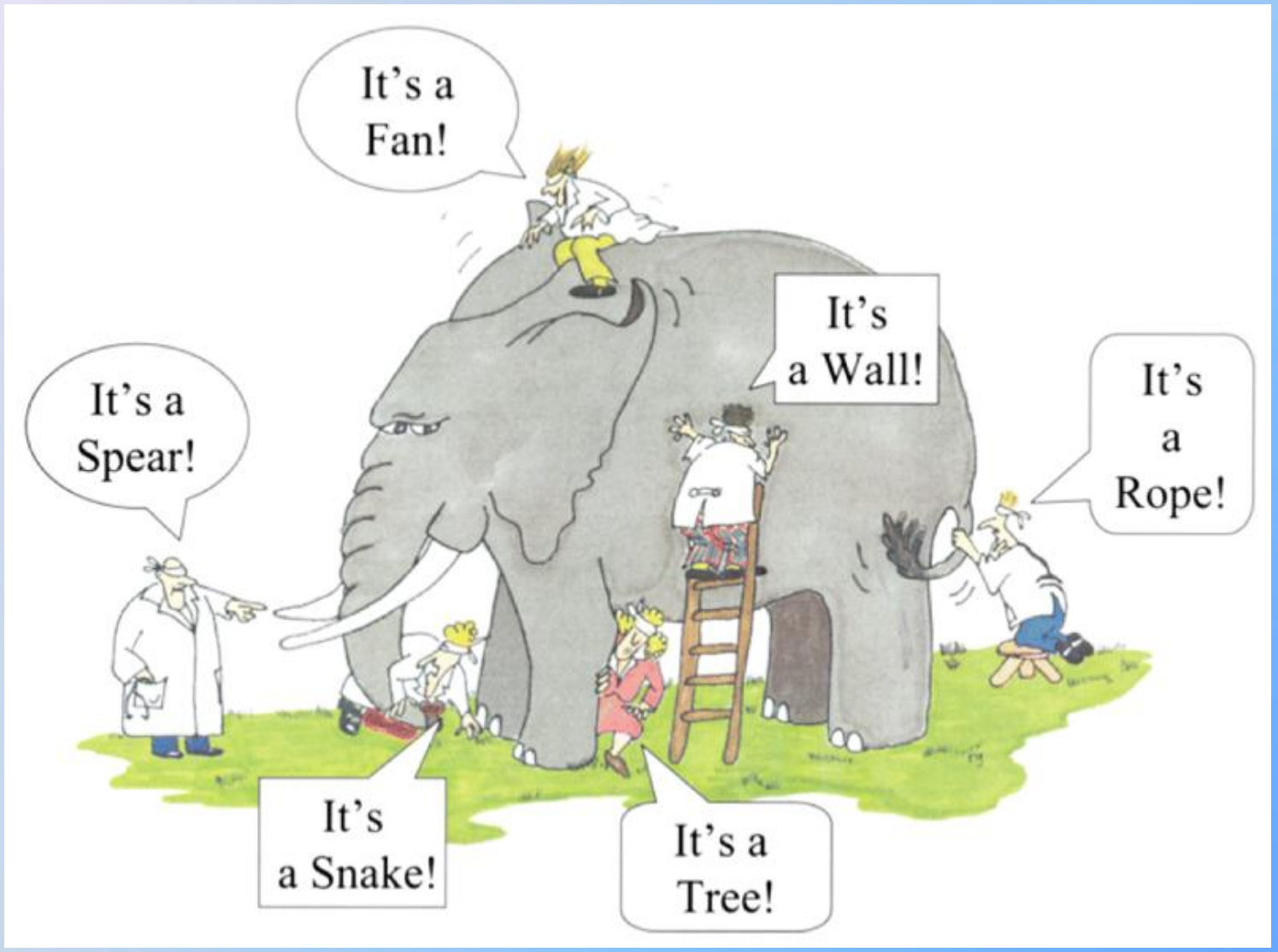
Transformation of the Software Testing Glossary into a Browsable Concept Map

Guntis Arnicans, Uldis Straujums
University of Latvia

International Conference on Engineering Education, Instructional Technology, Assessment, and E-learning (EIAE 12)

The work is partly supported by a European Social Fund Project No. 2009/0216/1DP/1.1.1.2.0/09 /APIA/VIAA/044 and by the Latvian National Research Program Nr. 2 „Development of Innovative Multifunctional Materials, Signal Processing and Information Technologies for Competitive Science Intensive Products" within the project Nr. 5 „New Information Technologies Based on Ontologies and Model Transformations"

Blind men and an elephant



Source: http://www.nature.com

Glossary of a domain

Software Testing Glossary

Last updated: Thursday, 24-May-2012 05:03:00 PDT

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A (return to top of page)

Acceptance Testing: Testing conducted to enable a user/customer to determine whether to accept a software product. Normally performed to validate the software meets a set of agreed acceptance criteria.

Accessibility Testing: Verifying a product is accessible to the people having disabilities (deaf, blind, mentally disabled etc.).

Ad Hoc Testing: A testing phase where the tester tries to 'break' the system by randomly trying the system's functionality. Can include negative testing as well. See also Monkey Testing.

Agile Testing: Testing practice for projects using agile methodologies, treating development as the customer of testing and emphasizing a test-first design paradigm. See also Test Driven Development.

Application Binary Interface (ABI): A specification defining requirements for portability of applications in binary forms across defferent system platforms and environments.

Application Programming Interface (API): A formalized set of software calls and routines that can be referenced by an application program in order to access supporting system or network services.

Automated Software Quality (ASQ): The use of software tools, such as automated testing tools, to improve software quality.

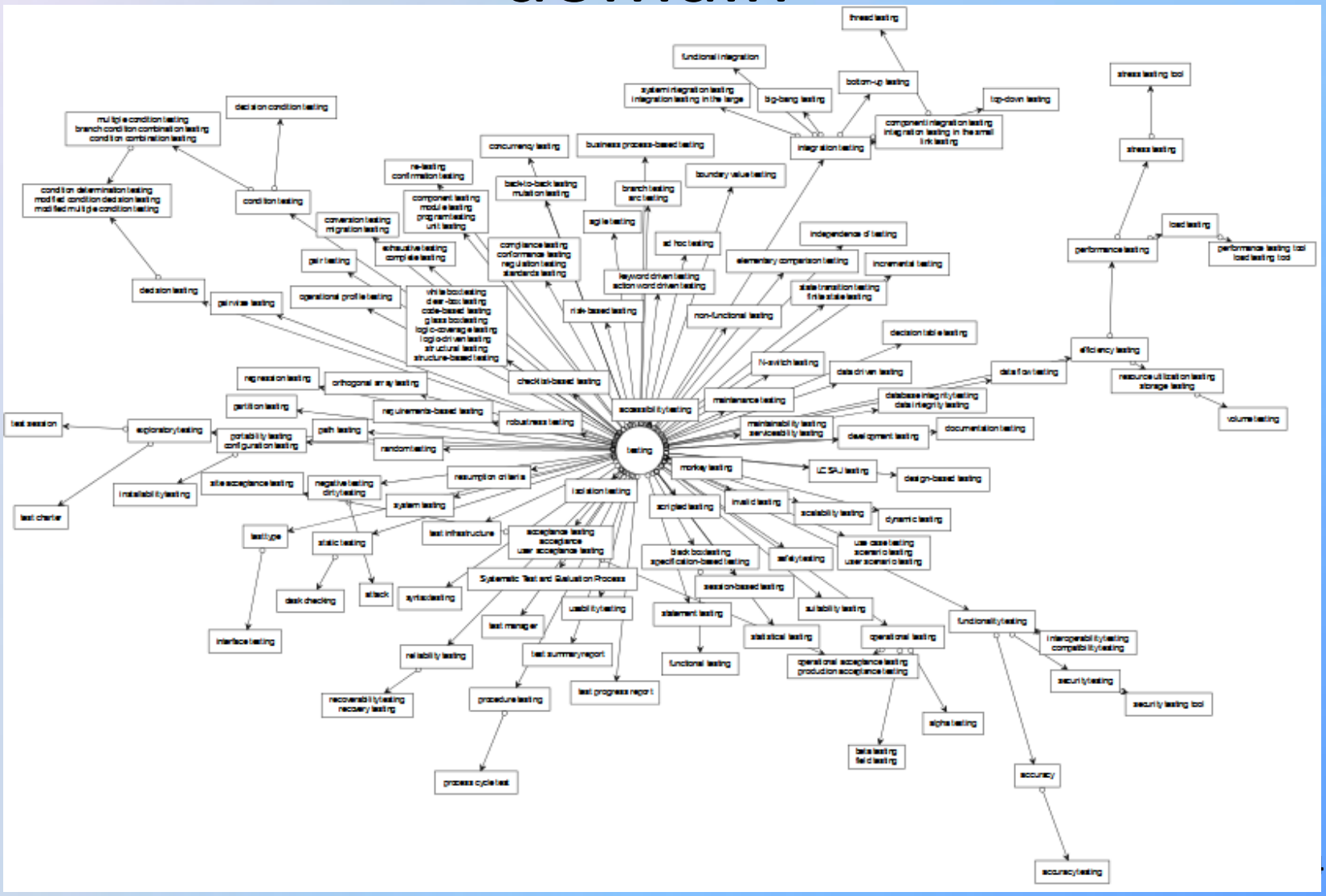
Automated Testing:

- Testing employing software tools which execute tests without manual intervention. Can be applied in GUI, performance, API, etc. testing.
- The use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.

B (return to top of page)

Backus-Naur Form: A metalanguage used to formally describe the syntax of a language.

Lightweight concept map of a domain



From glossary to concept map or ontology

Software Testing Glossary

Last updated: Thursday, 24-May-2012 05:03:00 PDT

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A (return to top of page)

Acceptance Testing: Testing conducted to enable a user/customer to determine whether to accept a software product. Normally performed to validate the software meets a set of agreed acceptance criteria.

Accessibility Testing: Verifying a product is accessible to the people having disabilities (deaf, blind, mentally disabled etc.).

Ad Hoc Testing: A testing phase where the tester tries to 'break' the system by randomly trying the system's functionality. Can include negative testing as well. See also Monkey Testing.

Agile Testing: Testing practice for projects using agile methodologies, treating development as the customer of testing and emphasizing a test-first design paradigm. See also Test Driven Development

Application Binary Interface (ABI): A specification defining requirement
defferent system platforms and environments.

Application Programming Interface (API): A formalized set of software
program in order to access supporting system or network services.

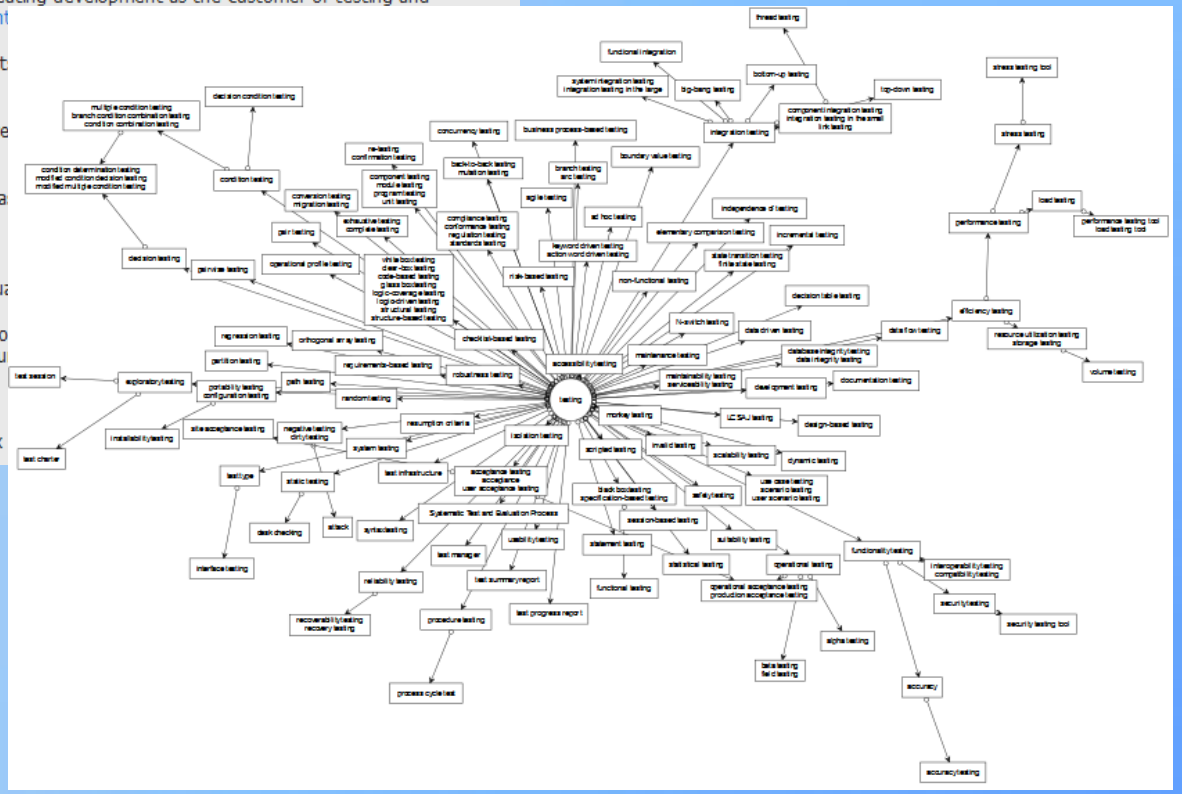
Automated Software Quality (ASQ): The use of software tools, such a

Automated Testing:

- Testing employing software tools which execute tests without manua
  etc. testing.
- The use of software to control the execution of tests, the compariso
  up of test preconditions, and other test control and test reporting fu

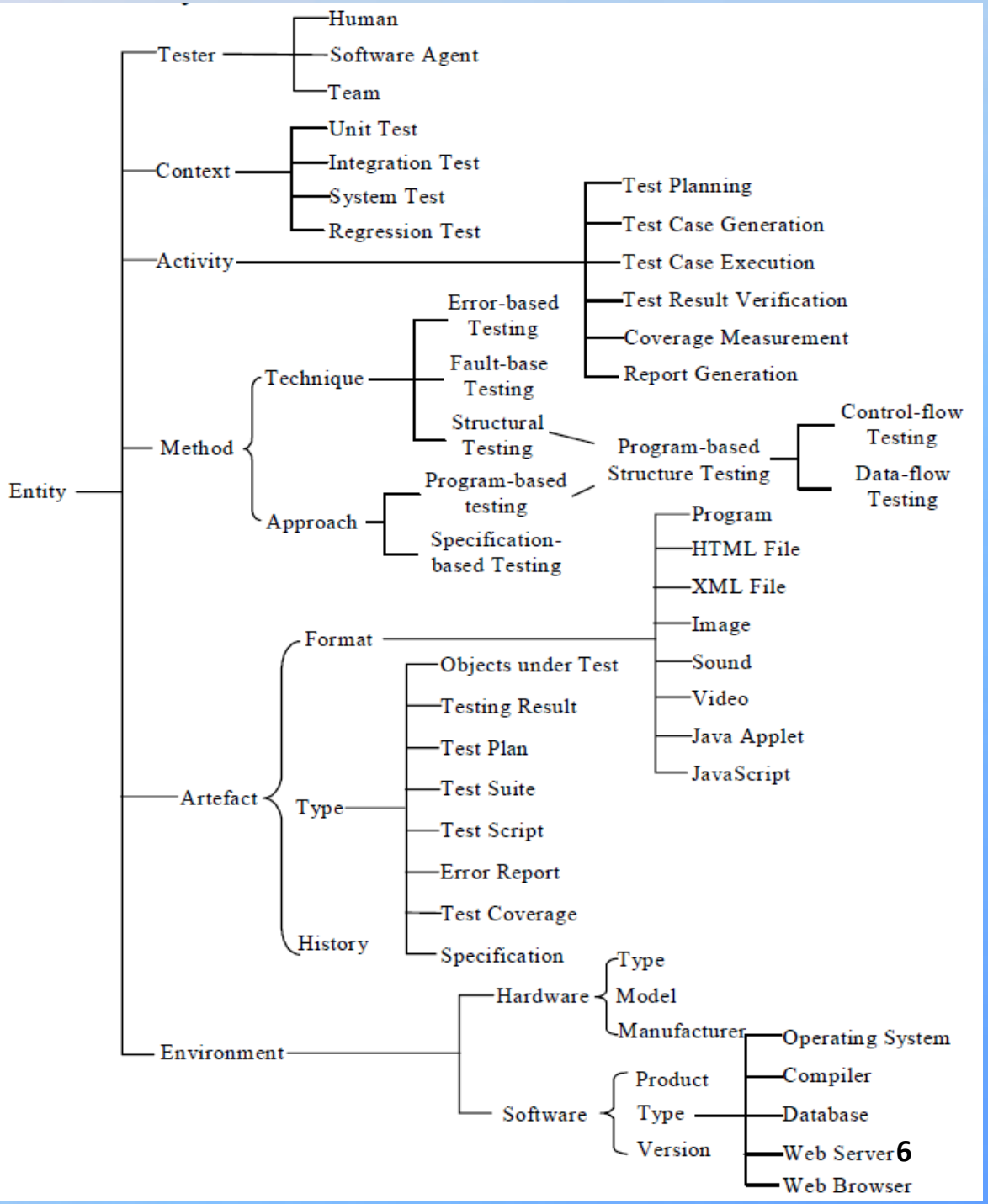
B (return to top of page)

Backus-Naur Form: A metalanguage used to formally describe the syntax



Domain ontology developed by experts

- H. Zhu and Q. Huo, 2005
- Ontology for an agent-based software environment to test web-based applications
- About 100 concepts



Related works (1/2)

Proposal of parallel construction of domain ontology and construction of complete domain terminology.

L. Bozzato, M. Ferrari, and A. Trombetta. ***Building a domain ontology from glossaries: a general methodology***. In A. Gangemi, J. Keizer, V. Presutti, and H. Stoermer, editors, Semantic Web Applications and Perspectives, SWAP 2008, volume 426 of CEUR Proceedings, 2008.

Related works (2/2)

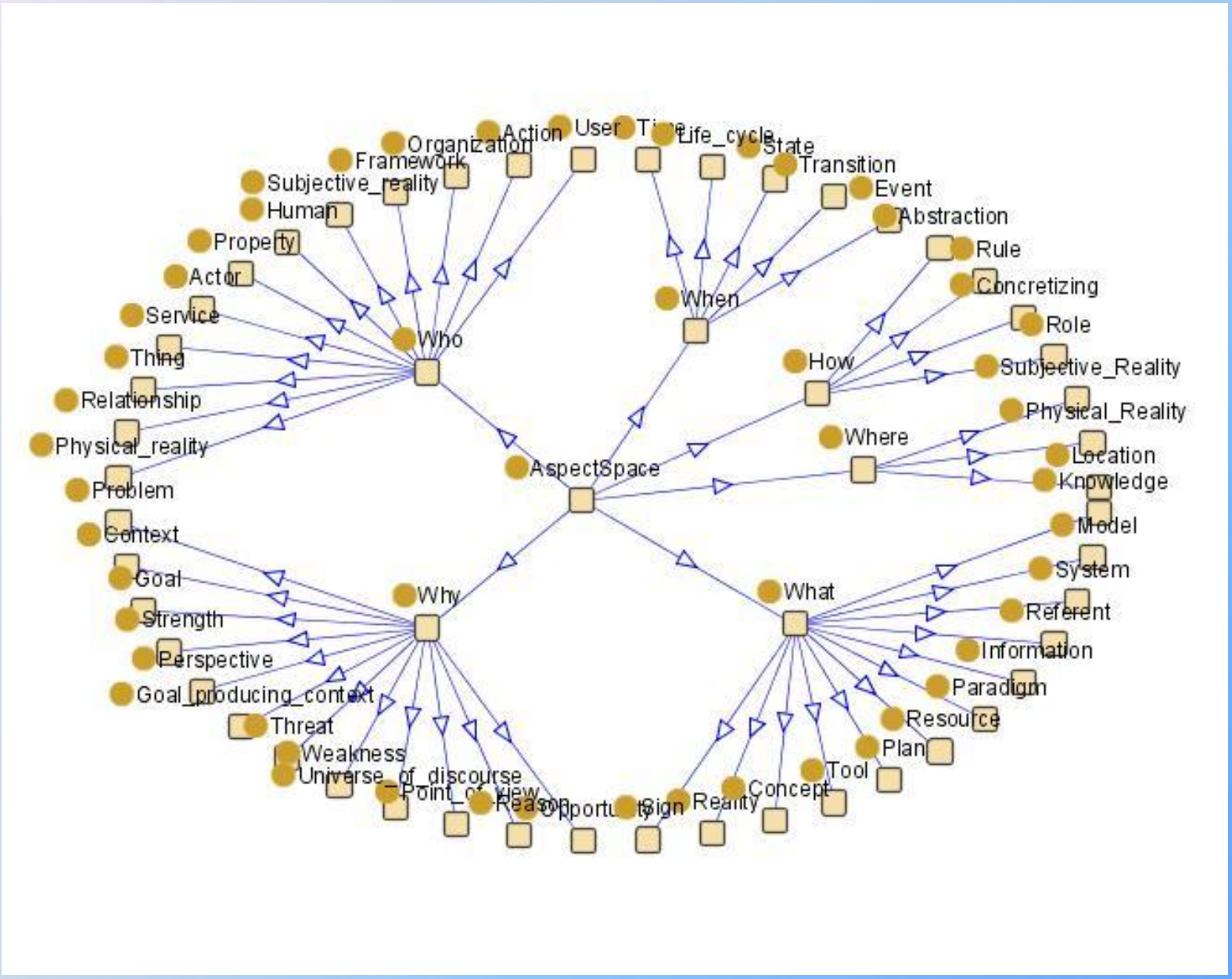
Obtaining of the ontology OntoGLOSE from the "*IEEE Standard Glossary of Software Engineering Terminology*".

Creating in some phases uses semi-automatic steps and uses semi-automatic linguistic analysis.

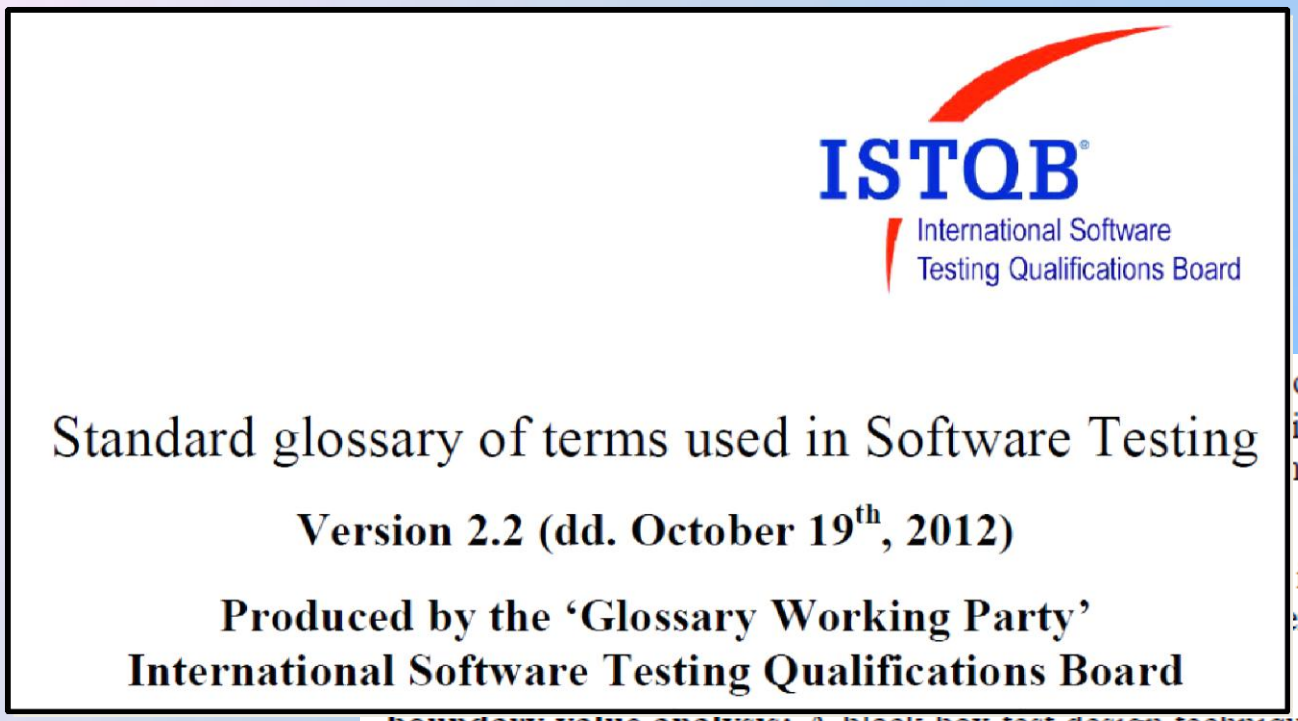
(No details of the automatization and results available)

Hilera José R., Pages Carmen, Martinez J. Javier, Gutierrez J. Antonio, De-Marcos Luis, ***An Evolutive Process to Convert Glossaries into Ontologies***, Information technology and libraries, vol. 29, no4(**2010**), 195-204.

ONTO6 Meta-Ontology top level simplified visualization



Initial document - glossary



Glossary contains 800 terms

on testing where the lowest level
itate the testing of higher level
nent at the top of the hierarchy is

is on the edge of an equivalence
r side of an edge, for example the

boundary value analysis: A black box test design technique in which test cases are designed based on boundary values. See also *boundary value.*

boundary value coverage: The percentage of boundary values that have been exercised by a test suite.

boundary value testing: See *boundary value analysis*.

branch: A basic block that can be selected for execution based on a program construct in which one of two or more alternative program paths is available, e.g. case, jump, go to, if-then-else.

Standard glossary of terms used in Software Testing

bottom-up testing: An incremental approach to integration testing where the lowest level components are tested first, and then used to facilitate the testing of higher level components. This process is repeated until the component at the top of the hierarchy is tested. See also *integration testing*.

boundary value: An input value or output value which is on the edge of an equivalence partition or at the smallest incremental distance on either side of an edge, for example the minimum or maximum value of a range.

boundary value analysis: A black box test design technique in which test cases are designed based on boundary values. See also *boundary value*.

boundary value coverage: The percentage of boundary values that have been exercised by a test suite.

boundary value testing: See *boundary value analysis*.

branch: A basic block that can be selected for execution based on a program construct in which one of two or more alternative program paths is available, e.g. case, jump, go to, if-then-else.

- The glossary contains 800 entries
- For comparison, "IEEE Standard Glossary of Software Engineering Terminology" (1990) contains approximately 1300 entries

Structure of the glossary

black box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.

specification-based testing: See *black box testing*.

functional testing: Testing based on an analysis of the specification of the functionality of a component or system. See also *black box testing*.

configuration control board (CCB): A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. [IEEE 610]

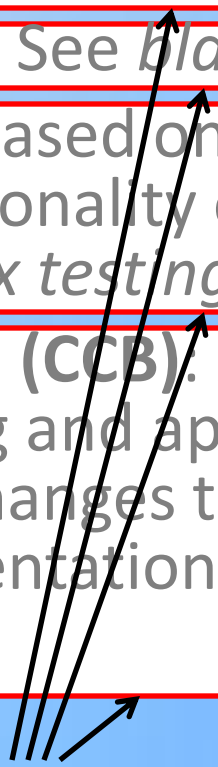
Structure of the glossary

black box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.

specification-based testing: See *black box testing*.

functional testing: Testing based on an analysis of the specification of the functionality of a component or system. See also *black box testing*.

configuration control board (CCB): A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. [IEEE 610]



Entries

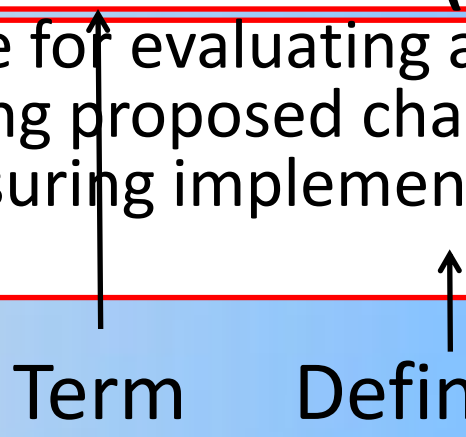
Structure of the glossary

black box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.

specification-based testing: See *black box testing*.

functional testing: Testing based on an analysis of the specification of the functionality of a component or system. See also *black box testing*.

configuration control board (CCB): A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. [IEEE 610]



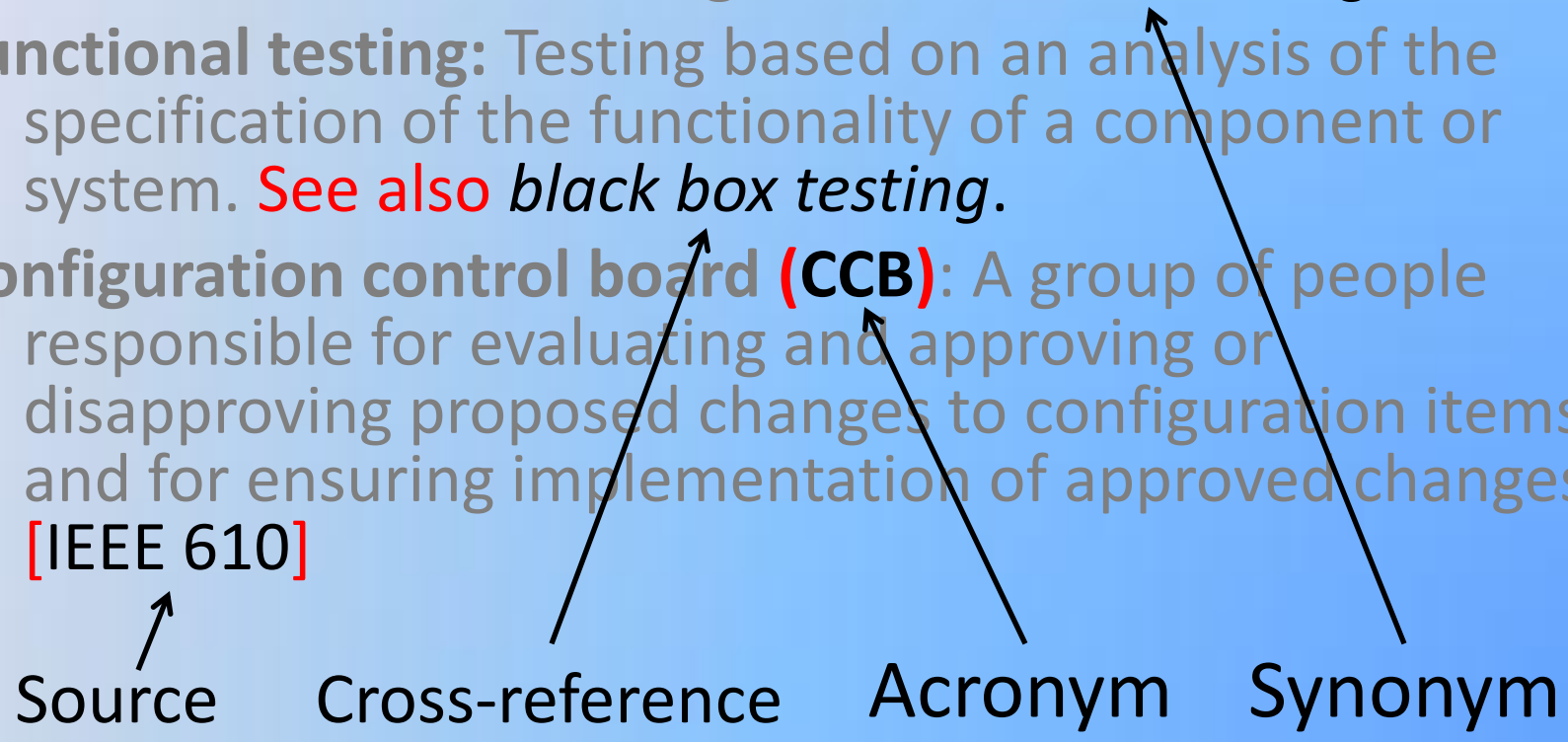
Structure of the glossary

black box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.

specification-based testing: See *black box testing*.

functional testing: Testing based on an analysis of the specification of the functionality of a component or system. See also *black box testing*.

configuration control board (CCB): A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. [IEEE 610]



Finding of significant aspects (words)

configuration control board (CCB): A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. [IEEE 610]

We can observe that:

1. The most semantically significant word of a term is at right hand side, usually it is the last word of term;
2. The most semantically significant word or words of definition are located at the beginning part of definition.

Glossary *entry* normalization

functional testing: Testing based on an analysis of the specification of the functionality of a component or system. See also *black box testing*.

functional testing **:** testing based analysis specification functionality component system

Indexing of words

- Assign an index to each instance of word
  - from right to left in term
  - from left to right in definition

functional(1) testing(0) : testing(0) based(1)
analysis(2) specification(3) functionality(4)
component(5) system(6)

Weighting of words

- Assign a weight to each instance of word
- Formula: $\mathbf{2^{-word\_index}}$

functional(2^{-1}) testing(2^{0}) : testing(2^{0})
based(2^{-1}) analysis(2^{-2}) specification(2^{-3})
functionality(2^{-4}) component(2^{-5}) system(2^{-6})

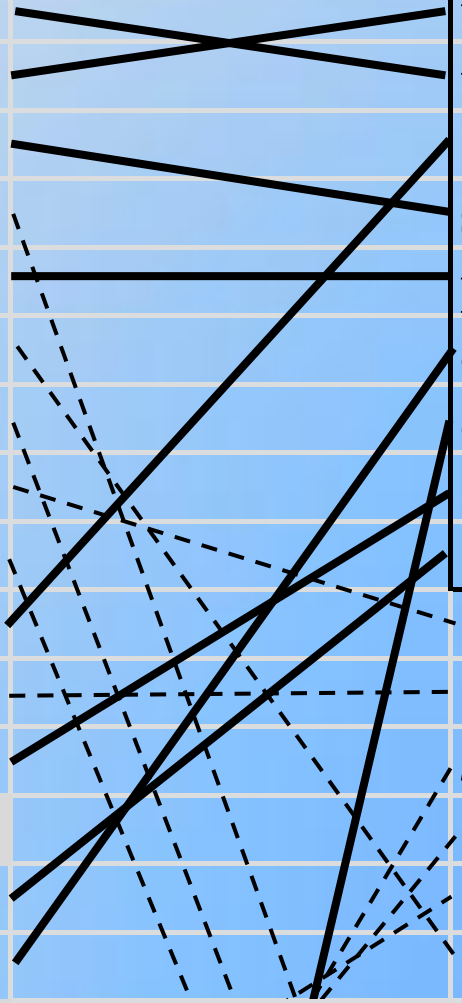
Total weight for word «testing» in the entry is
$\mathbf{2^{0} + 2^{0} = 2}$
Total weight for word «analysis» in the entry is
$\mathbf{2^{-2} = 0.25}$

Word weighting process result (1/2)

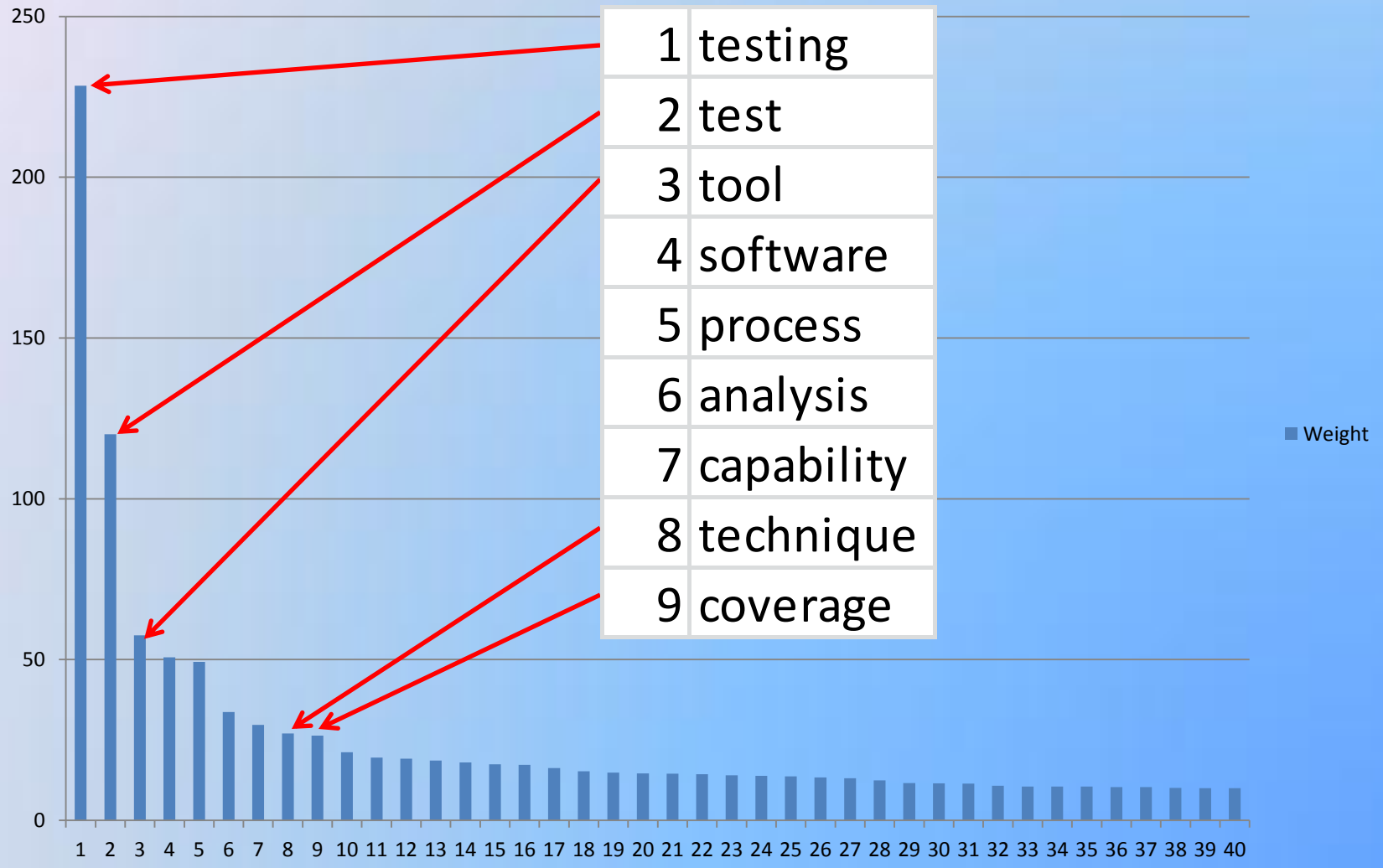
Rank	Count	Word	Word	Weight
1	512	**test**	**testing**	**228.49**
2	345	**testing**	**test**	**120.07**
3	180	**software**	*tool*	**57.54**
4	137	**system**	**software**	**50.67**
5	125	**process**	**process**	**49.26**
6	118	**component**	*analysis*	**33.69**
7	87	*product*	*capability*	**29.71**
8	77	*based*	*technique*	**27.03**
9	75	*design*	**coverage**	**26.35**
10	75	*tool*	*based*	21.17
11	68	**quality**	**quality**	19.53
12	67	*technique*	*set*	19.21
13	60	*execution*	*management*	18.61
14	60	**coverage**	**condition**	18.05
15	59	*analysis*	**component**	17.43
16	58	*data*	*model*	17.31
17	54	*requirements*	*percentage*	16.25
18	52	**condition**	*box*	15.25
19	52	*control*	**risk**	14.86
20	51	*development*	*document*	14.57
21	49	*management*	*black*	14.56
22	48	*level*	**system**	14.37
23	46	*set*	*report*	14.01
24	44	*model*	*product*	13.85
25	42	*activities*	*design*	13.68
26	42	**defect**	**review**	13.32
27	40	**project**	*approach*	13.07
28	40	**decision**	**integration**	12.42
29	40	**risk**	**case**	11.60
30	39	*user*	*development*	11.50
31	39	*determine*	**result**	11.43
32	39	*phase*	*criteria*	10.77
33	38	*specified*	*white*	10.56
34	34	*capability*	**statement**	10.54
35	34	**result**	**path**	10.53
36	34	**performance**	**specification**	10.39
37	33	**code**	*control*	10.35
38	33	**input**	*degree*	10.08
39	33	**specification**	*type*	10.03
40	33	*time*	*level*	10.00

Word weighting process result (2/2)

Rank	Count	Word		Word	Weight
1	512	**test**		**testing**	**228.49**
2	345	**testing**		**test**	**120.07**
3	180	**software**		*tool*	**57.54**
4	137	**system**		**software**	**50.67**
5	125	**process**		**process**	**49.26**
6	118	**component**		*analysis*	**33.69**
7	87	*product*		*capability*	**29.71**
8	77	*based*		*technique*	**27.03**
9	75	*design*		**coverage**	**26.35**
10	75	*tool*		*based*	21.17
11	68	**quality**		**quality**	19.53
12	67	*technique*		*set*	19.21
13	60	*execution*		*management*	18.61
14	60	**coverage**		**condition**	18.05
15	59	*analysis*		**component**	17.43



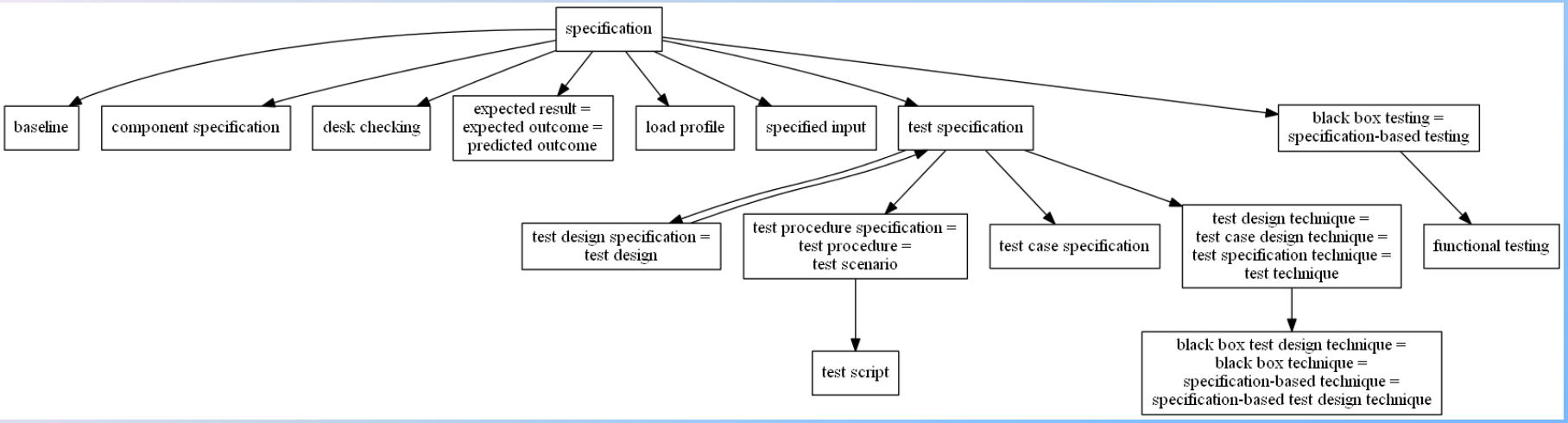
Word weight distribution



Creation of the aspect graphs

- An *aspect graph* is a set of nodes, which corresponds to terms, and edges (relations) among them.
- At first we find all entries that belong to a given aspect according to the aspect word.
- Then a graph is created
  - any two nodes are connected by edge if a relation between corresponding terms is discovered
  - graph is simplified by reducing nodes (merging of nodes that correspond to synonym terms) and by reducing edges (deleting excessive relations assuming that all relations are transitive)
- The details of algorithms are described in
  - **G.Arnicans**, **D.Romans**, and **U.Straujums**, "**Semi-automatic Generation of a Software Testing Lightweight Ontology from a Glossary Based on the ONTO6 Methodology**", Databases and Information Systems VII, Selected Papers from the Tenth International Baltic Conference, DB&IS 2012, IOS Press, 14 p., in press.

Results

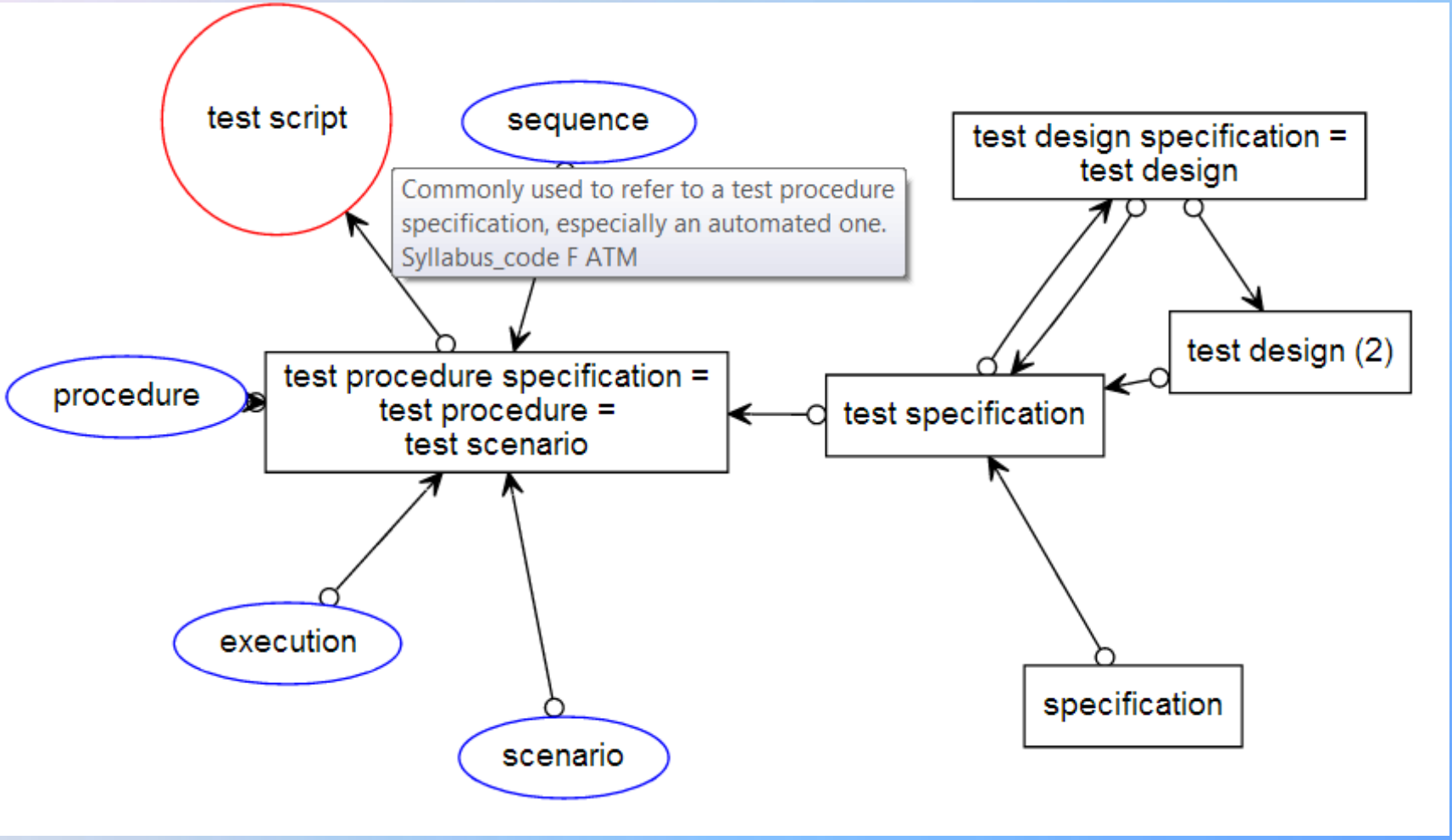


The weightiest 9 words *testing*, *test*, *tool*, *software*, *process*, *analysis*, *capability*, *technique*, *coverage* *contain 70% of all term-nodes*.

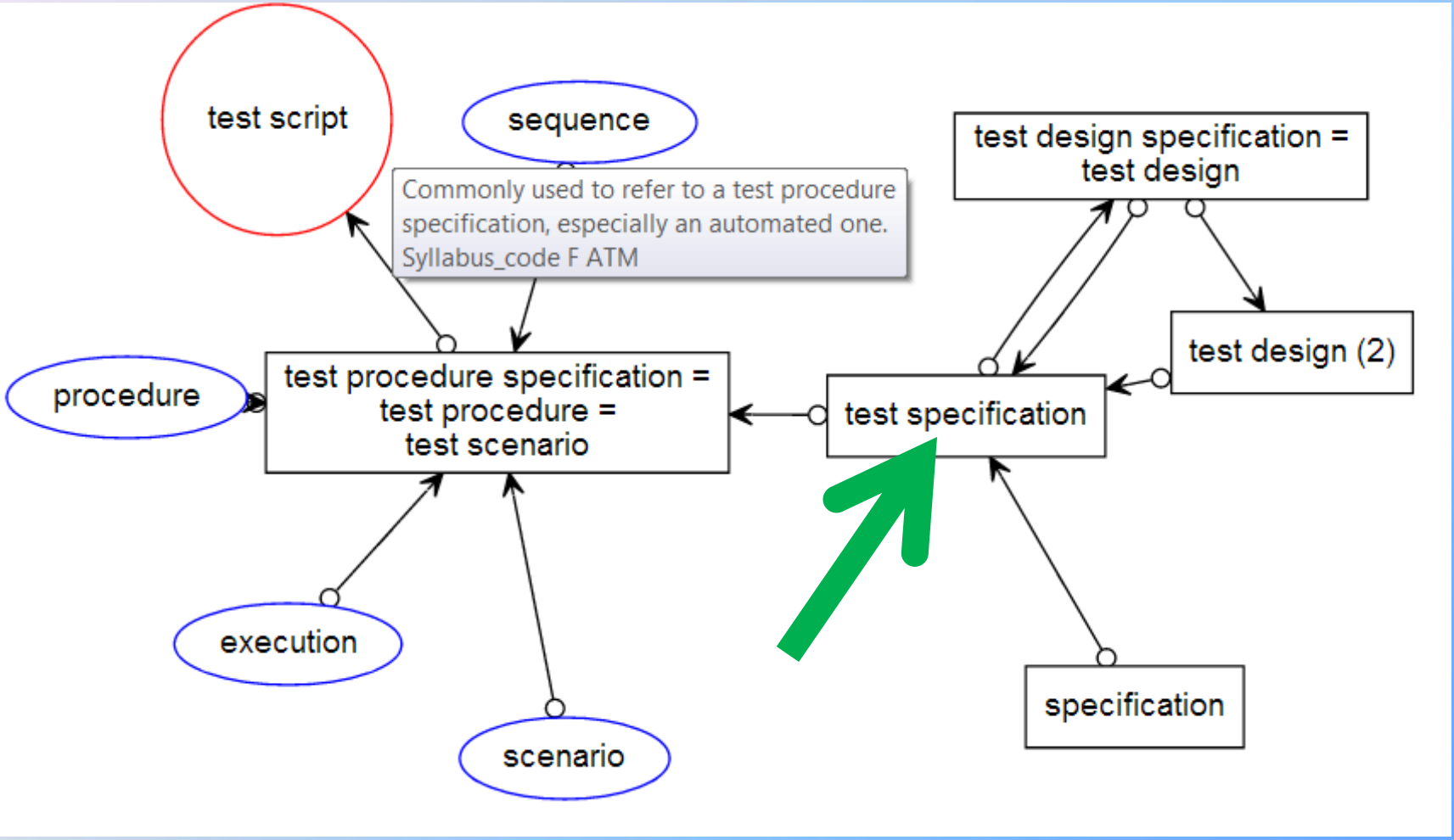
Total number of aspect graphs is **325** containing **608** unique term-nodes and **170** unique word-nodes.

These **9** aspects include 425 term-nodes (**70%**).

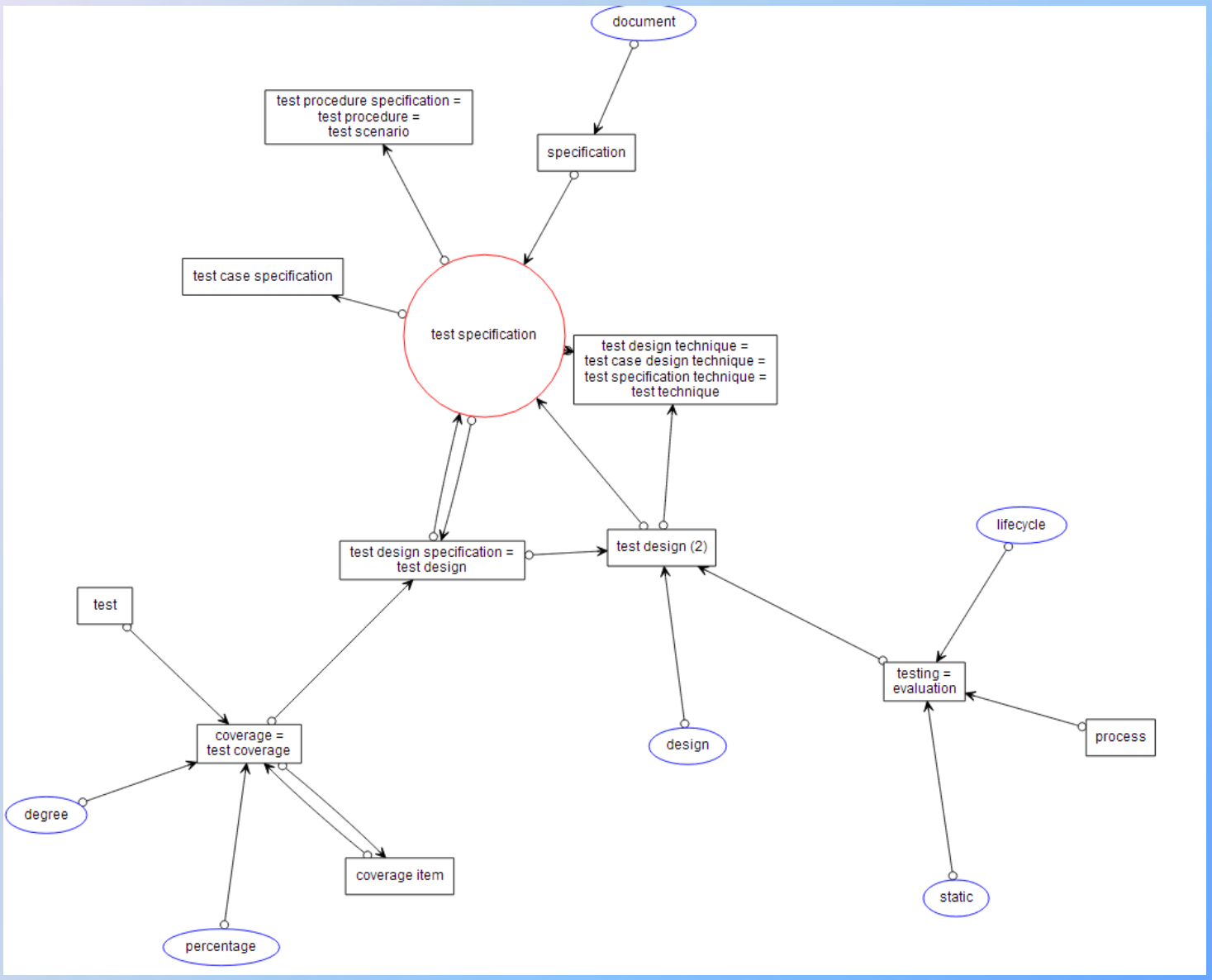
Example of concept map browsing



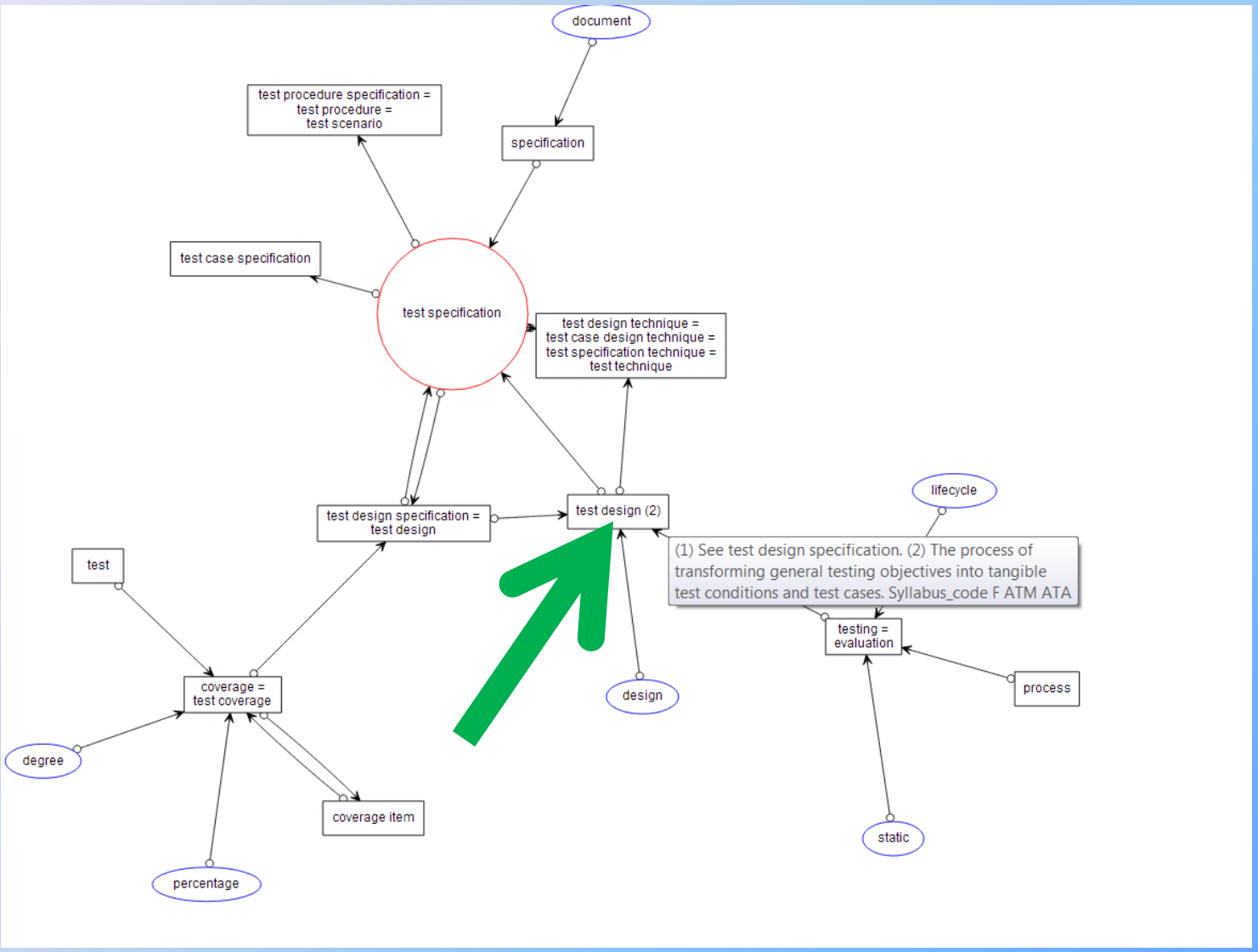
Example of concept map browsing



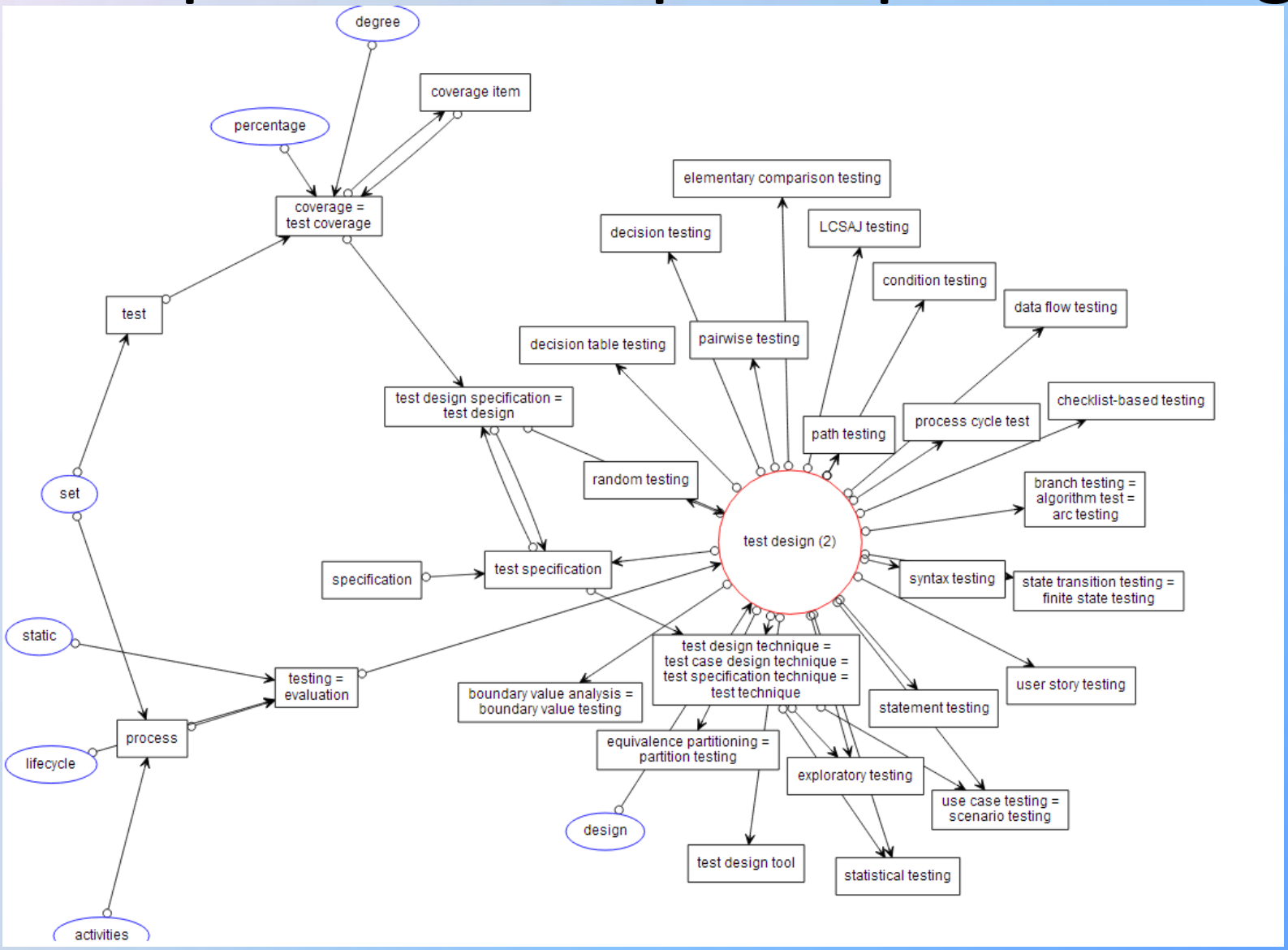
Example of concept map browsing



Example of concept map browsing



Example of concept map browsing



Conclusion and future work

- It is possible to semi-automatically generate a concept map from glossary
- We offer the principles and algorithms how to discover the significant concepts and to find simple relations between concepts in a glossary
- The initial concept map already allows to conceive the surroundings of a term in a browsable way
- We are going to continue work to deepen morphological analysis, to improve visualization of surroundings, to support concept map editing, etc.

Thank you very much for your attention

Complementary material to the paper

http://**science.df.lu.lv/as12/**