Concept map for the software testing domain

Guntis Arnicans

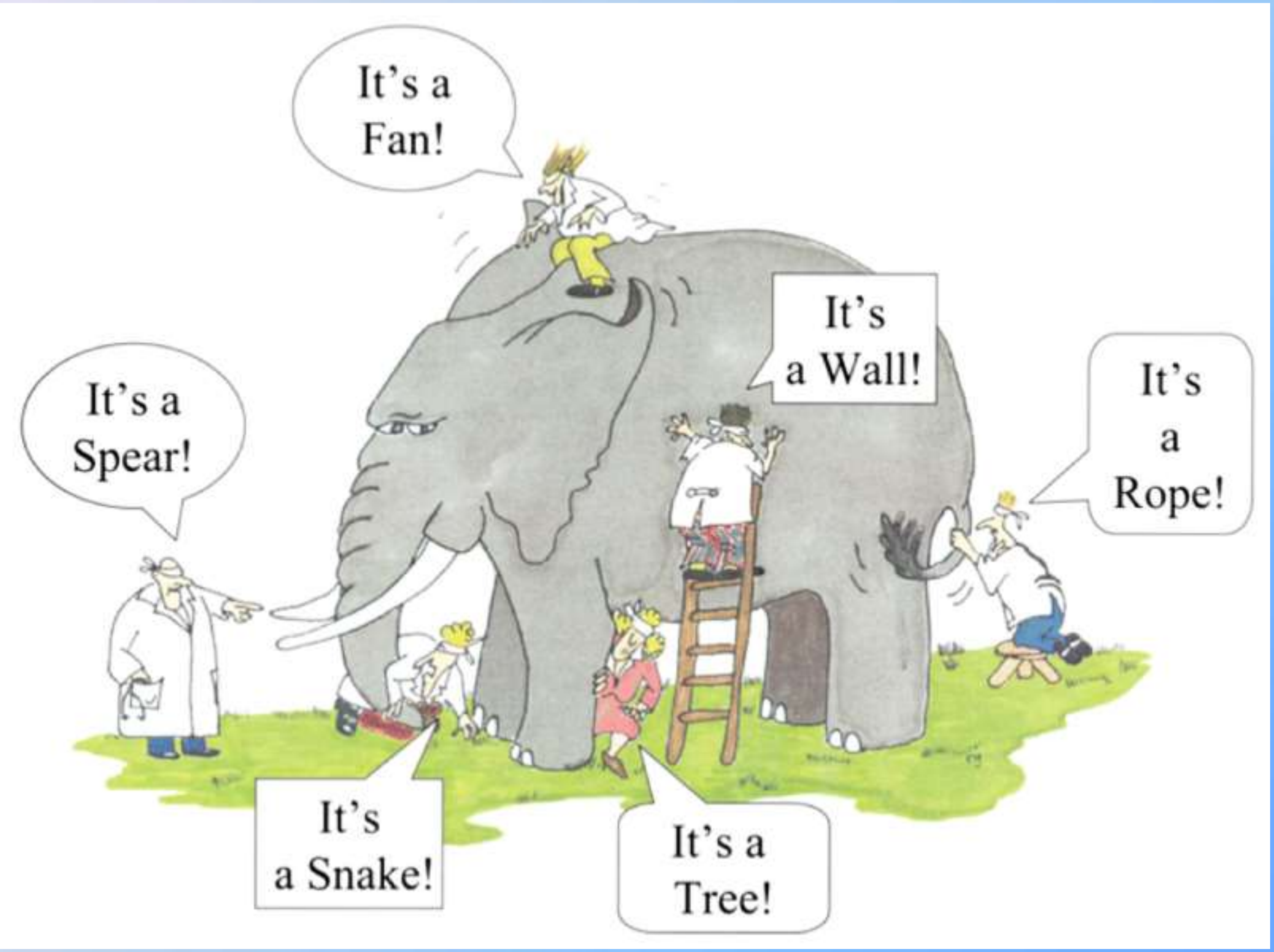
University of Latvia

Theory and Practice of Software Testing (TAPOST 2013),

14th International Conference “Testing and Social Networks”

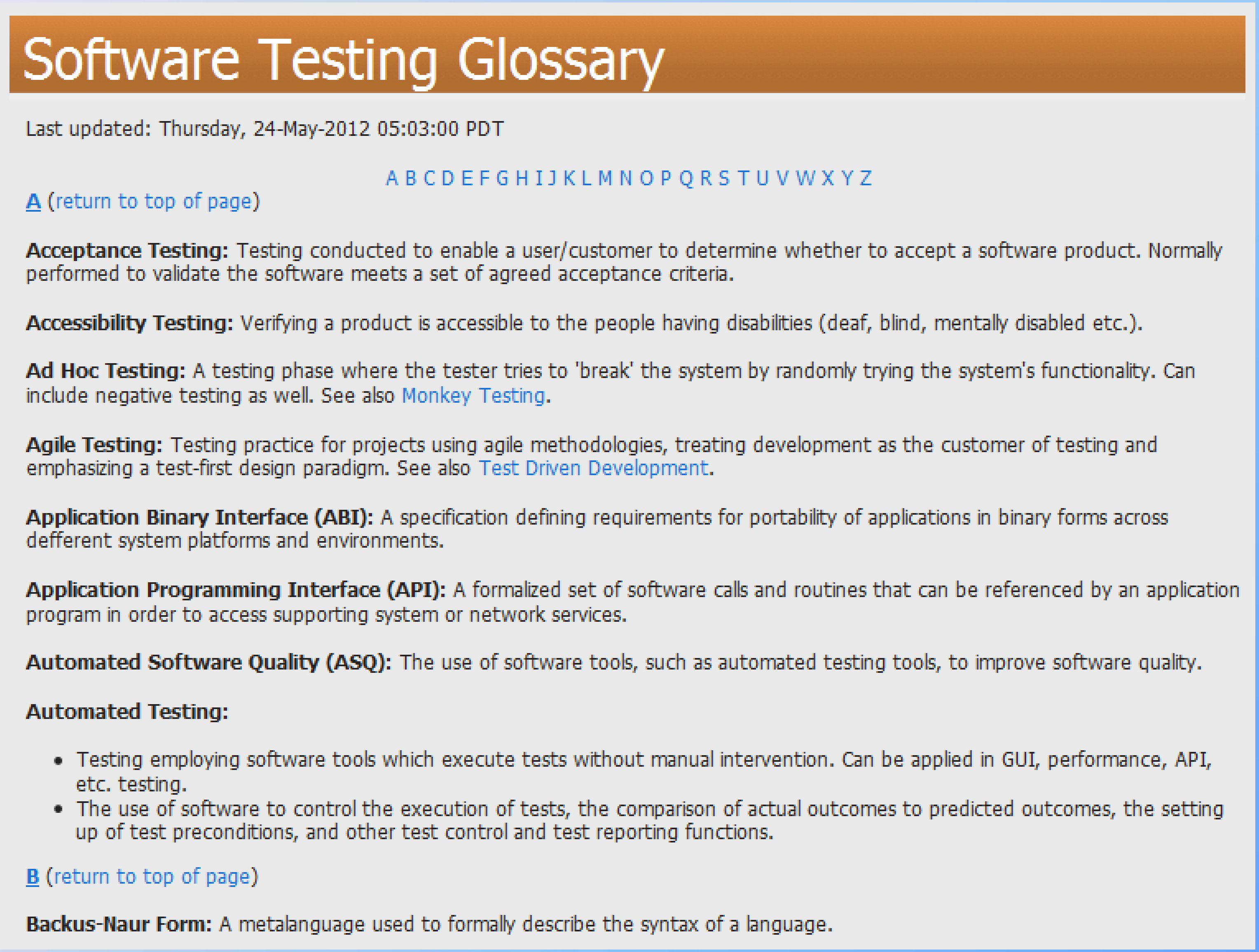
Riga, Latvia, May 28, 2013

Blind men and an elephant



Source: http://www.nature.com

Glossary – a view to a knowledge 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.

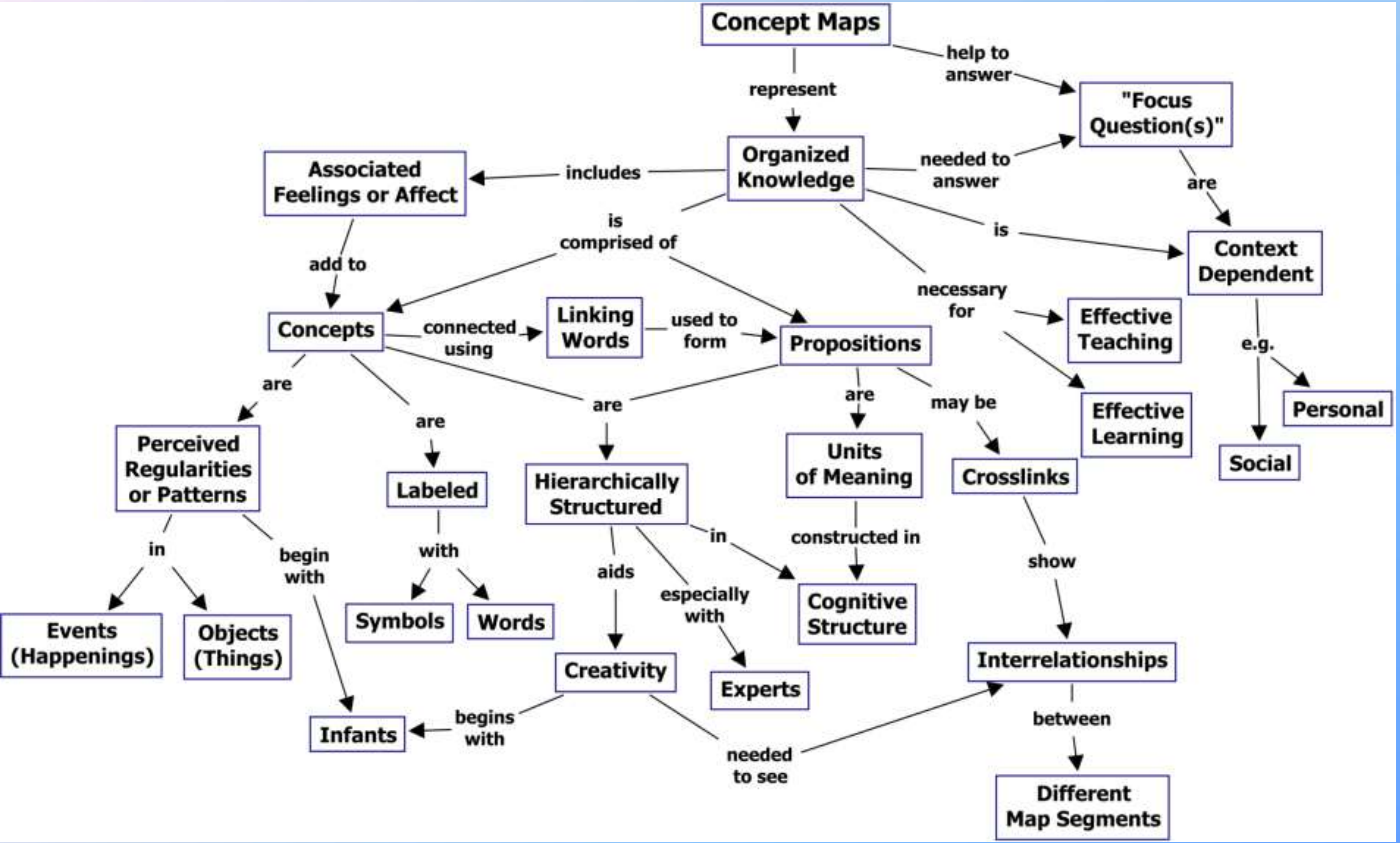
Automated Testing:

- 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.

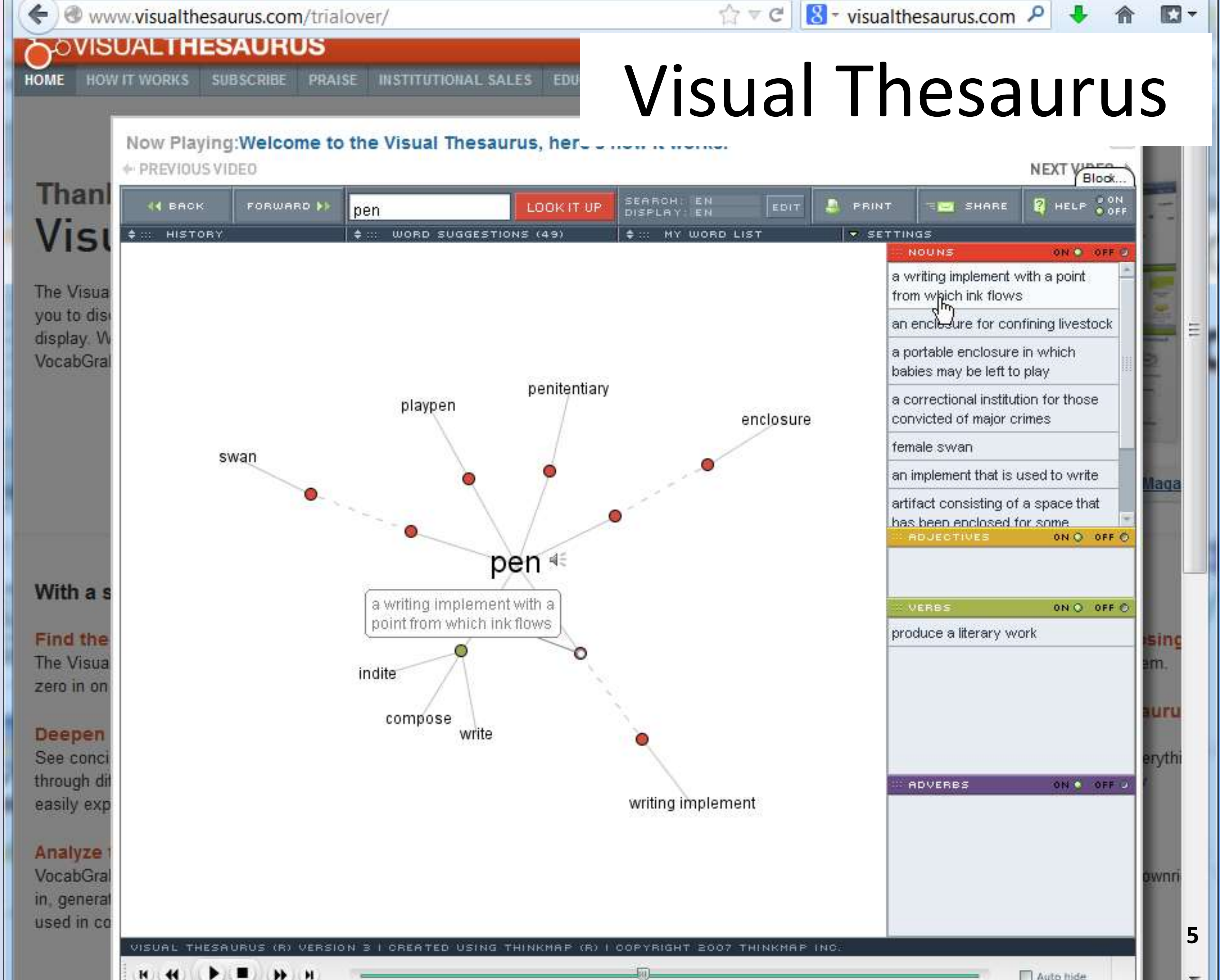
Concept map



Concept Maps
represent
help to answer
"Focus Question(s)"
Organized Knowledge
needed to answer
are
Associated Feelings or Affect
includes
is comprised of
is
Context Dependent
add to
necessary for
Effective Teaching
Concepts
connected using
Linking Words
used to form
Propositions
e.g.
Effective Learning
Personal
are
are
are
are
may be
Perceived Regularities or Patterns
Labeled
Hierarchically Structured
Units of Meaning
Crosslinks
Social
in
begin with
with
aids
in
constructed in
show
especially with
Events (Happenings)
Objects (Things)
Symbols
Words
Cognitive Structure
Creativity
Experts
Interrelationships
Infants
begins with
between
needed to see
Different Map Segments

Source: http://cmap.ihmc.us/publications/researchpapers/theorycmaps/theoryunderlyingconceptmaps.htm

Visual Thesaurus



NOUNS

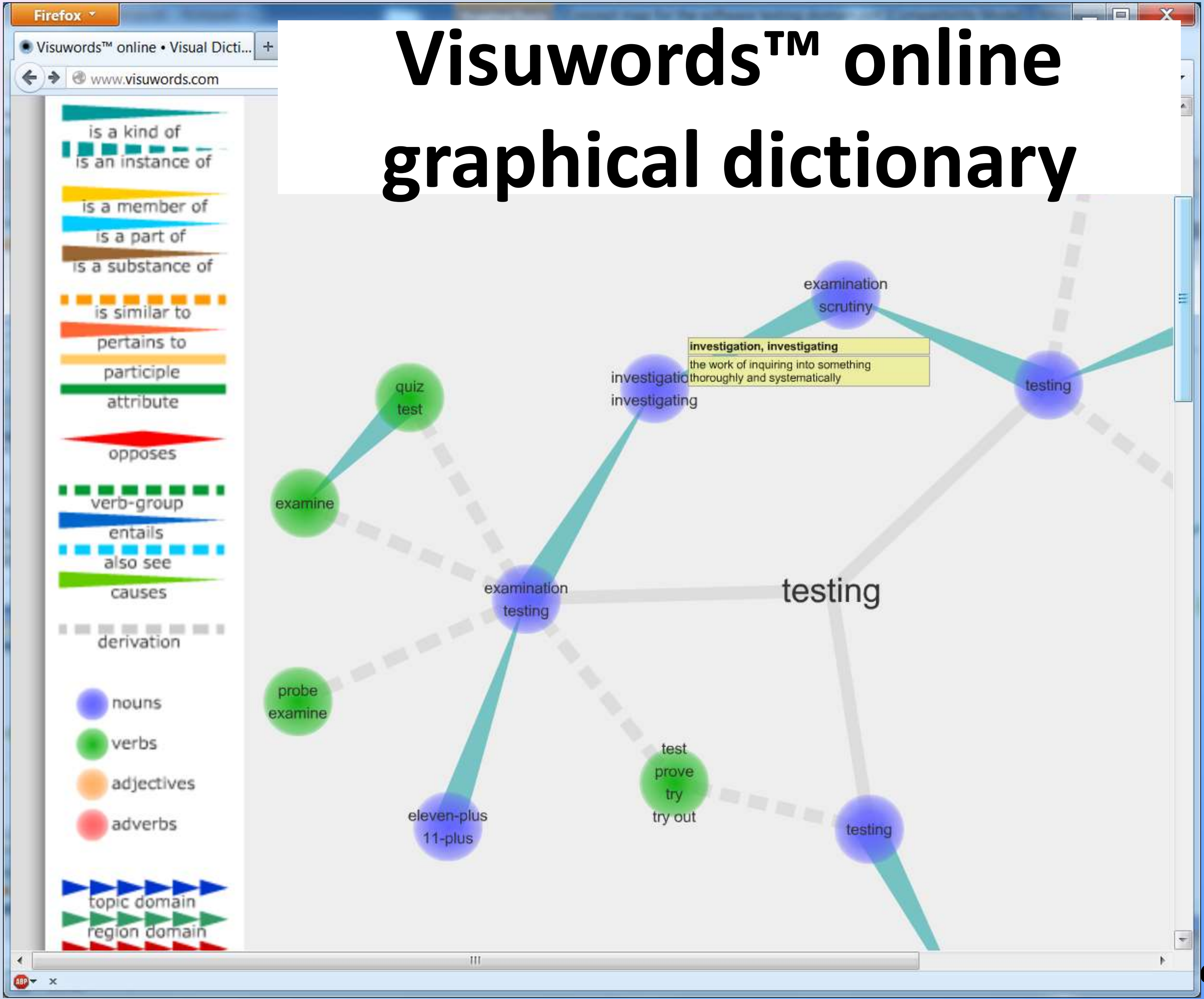
- a writing implement with a point from which ink flows
- an enclosure for confining livestock
- a portable enclosure in which babies may be left to play
- a correctional institution for those convicted of major crimes
- female swan
- an implement that is used to write
- artifact consisting of a space that has been enclosed for some

VERBS

- produce a literary work

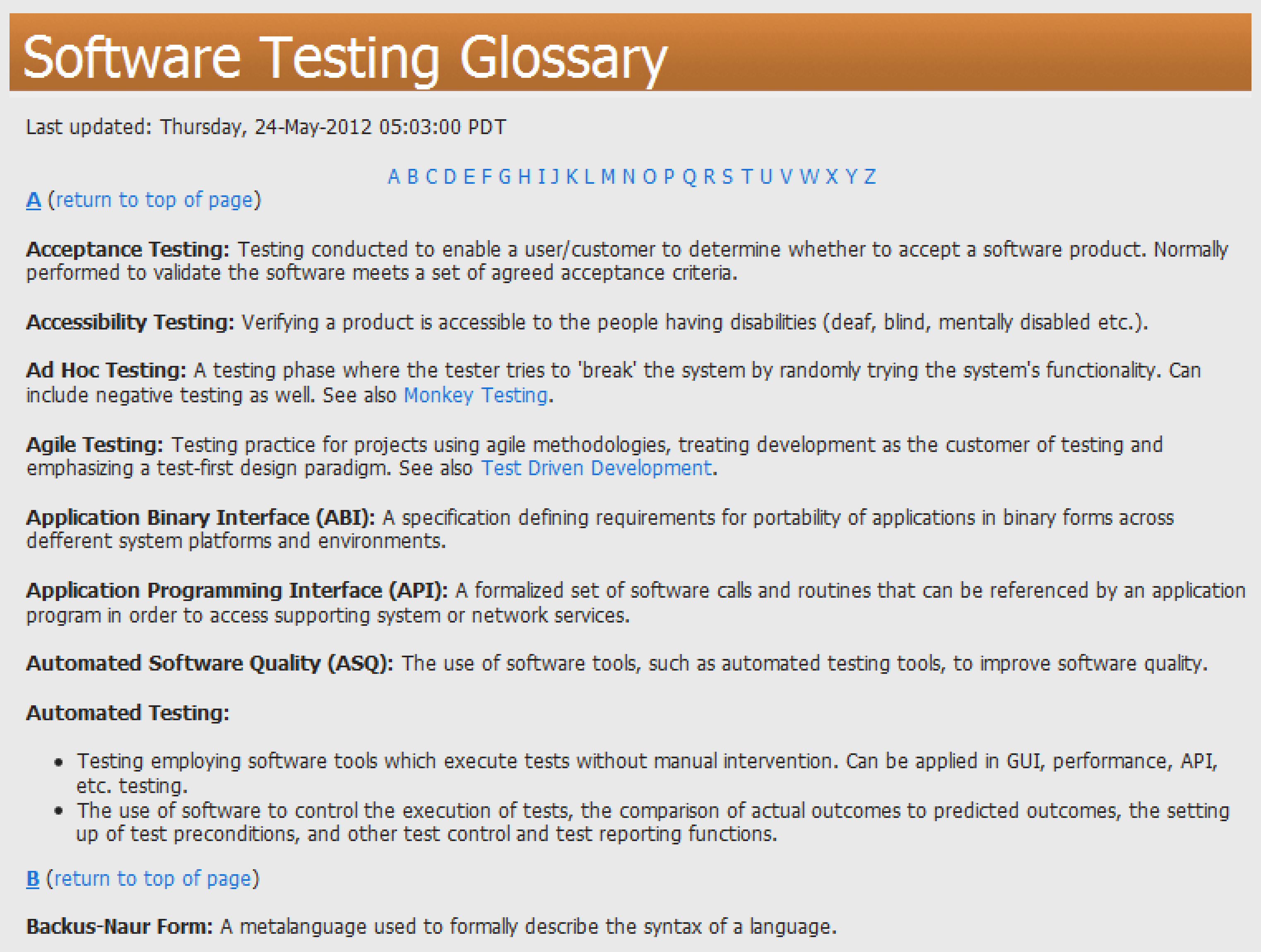
a writing implement with a point from which ink flows

Visuwords™ online graphical dictionary



investigation, investigating
the work of inquiring into something thoroughly and systematically

Glossary – links between terms



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:

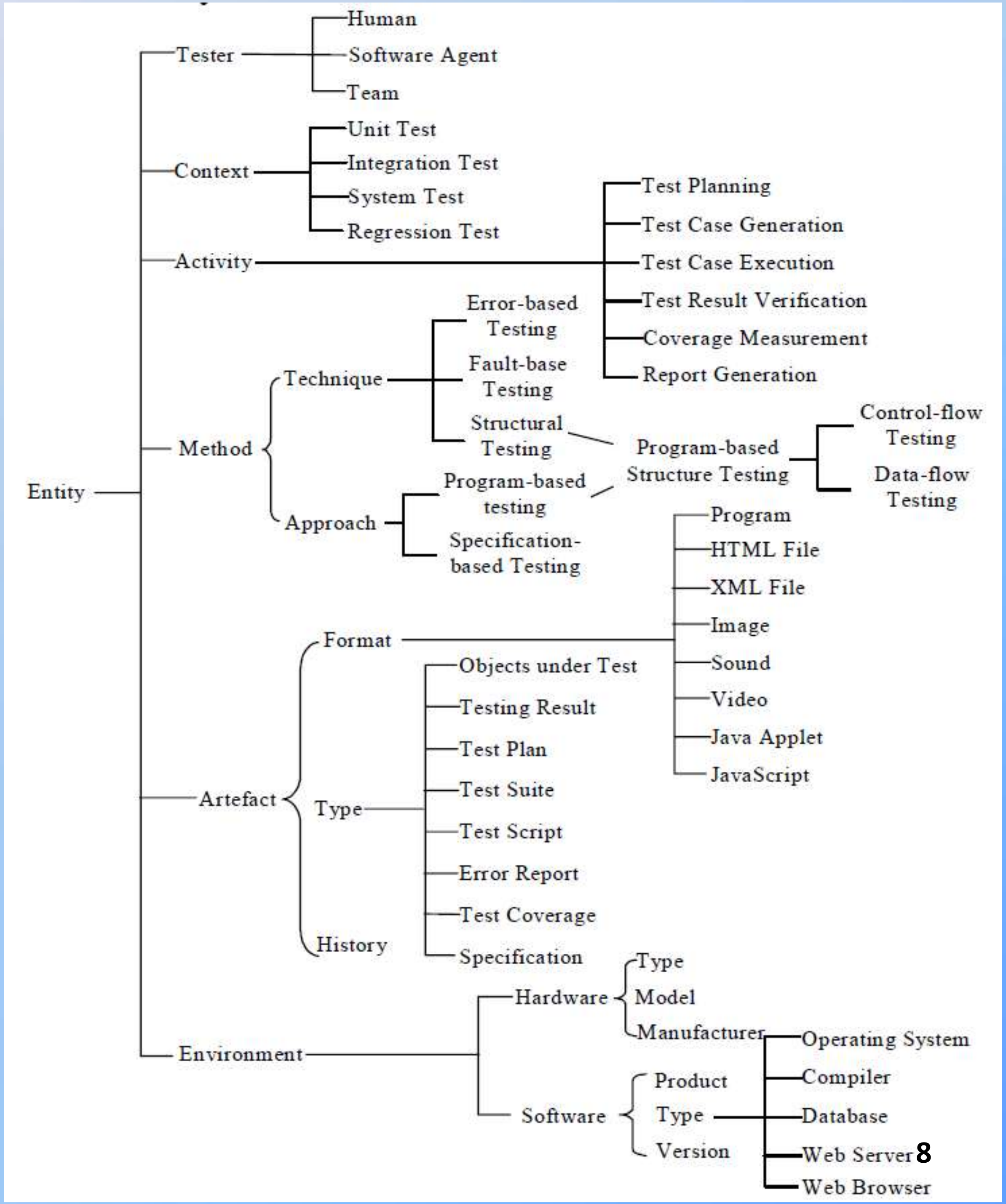
- Testing employing software tools which execute tests without manual intervention. Can be applied in GUI, performance, API, etc. testing.
- The use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other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

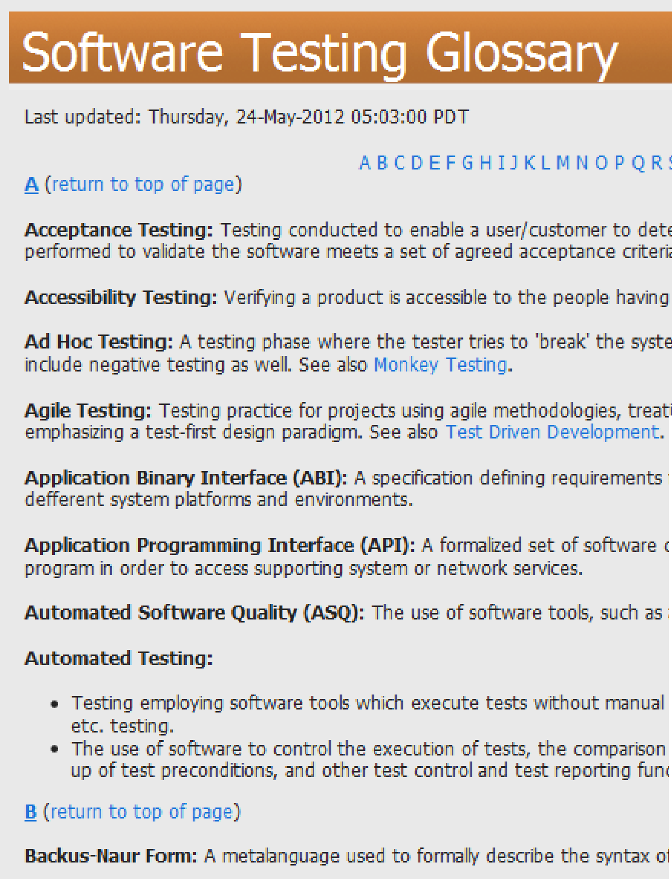
Domain ontology developed by experts

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



- Entity
  - Tester
    - Human
    - Software Agent
    - Team
  - Context
    - Unit Test
    - Integration Test
    - System Test
    - Regression Test
  - Activity
    - Test Planning
    - Test Case Generation
    - Test Case Execution
    - Test Result Verification
    - Coverage Measurement
    - Report Generation
  - Method
    - Technique
      - Error-based Testing
      - Fault-base Testing
      - Structural Testing
    - Approach
      - Program-based testing
      - Specification-based Testing
    - Program-based Structure Testing
      - Control-flow Testing
      - Data-flow Testing
  - Artefact
    - Format
      - Program
      - HTML File
      - XML File
      - Image
      - Sound
      - Video
      - Java Applet
      - JavaScript
    - Type
      - Objects under Test
      - Testing Result
      - Test Plan
      - Test Suite
      - Test Script
      - Error Report
      - Test Coverage
      - Specification
    - History
  - Environment
    - Hardware
      - Type
      - Model
      - Manufacturer
    - Software
      - Product
      - Type
        - Operating System
        - Compiler
        - Database
        - Web Server
        - Web Browser
      - Version

Transformation from glossary to concept map



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
defferent system platforms and environments.

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

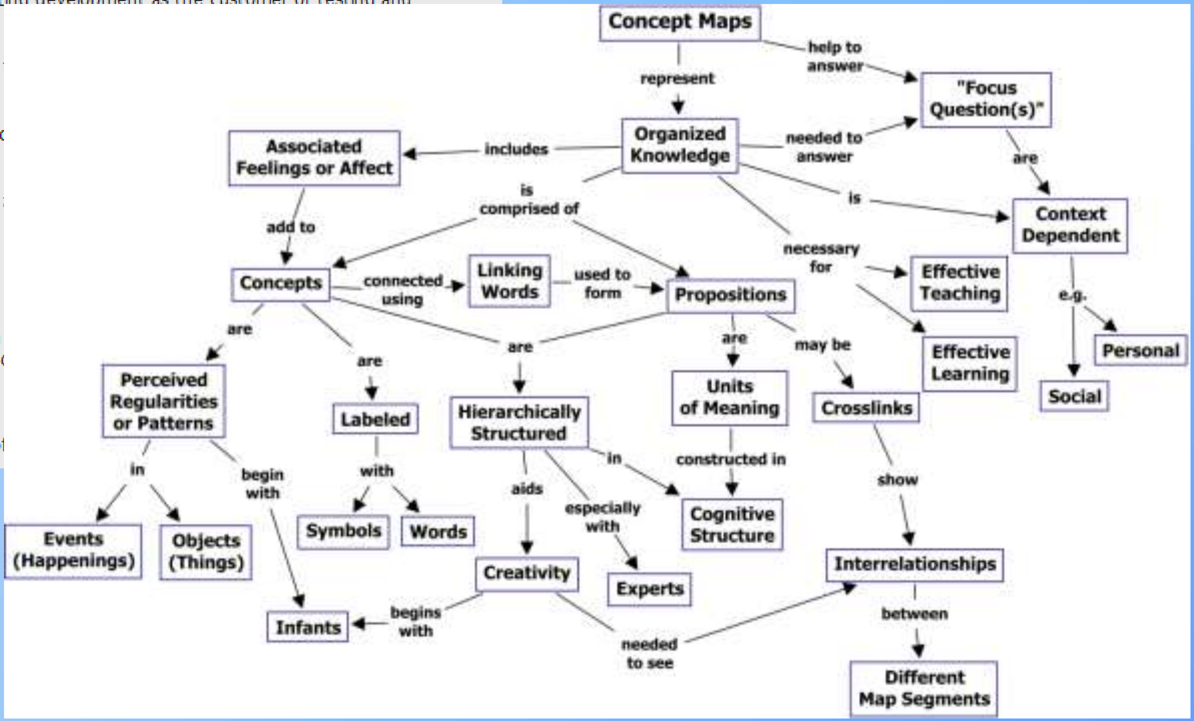
Automated Software Quality (ASQ): The use of software tools, such as

Automated Testing:

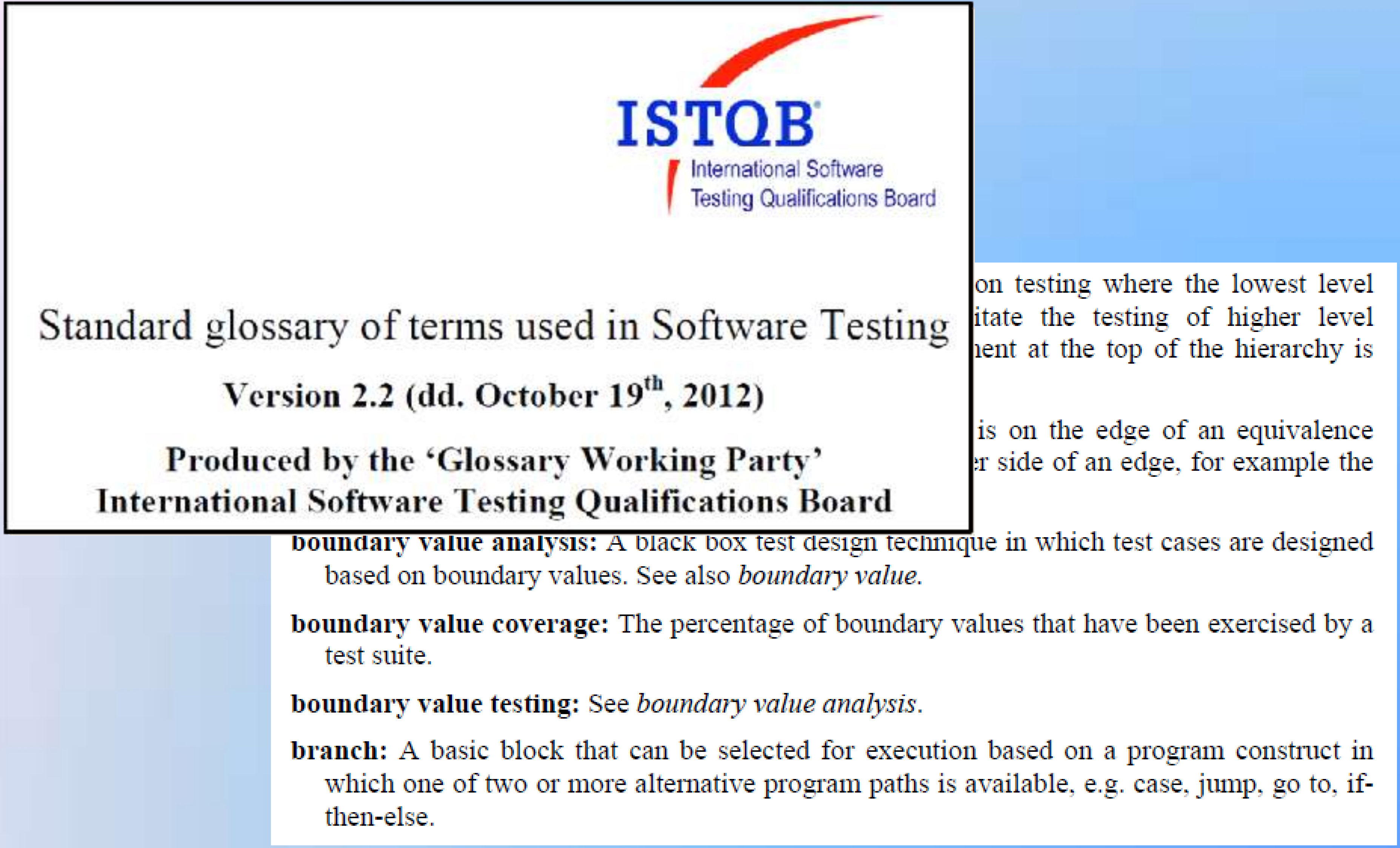
- Testing employing software tools which execute tests without manual
etc. testing.
- The use of software to control the execution of tests, the comparison
up of test preconditions, and other test control and test reporting fun

B (return to top of page)

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



Initial document - glossary



ISTQB

International Software Testing Qualifications Board

Standard glossary of terms used in Software Testing

Version 2.2 (dd. October 19th, 2012)

Produced by the 'Glossary Working Party'
International Software Testing Qualifications Board

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

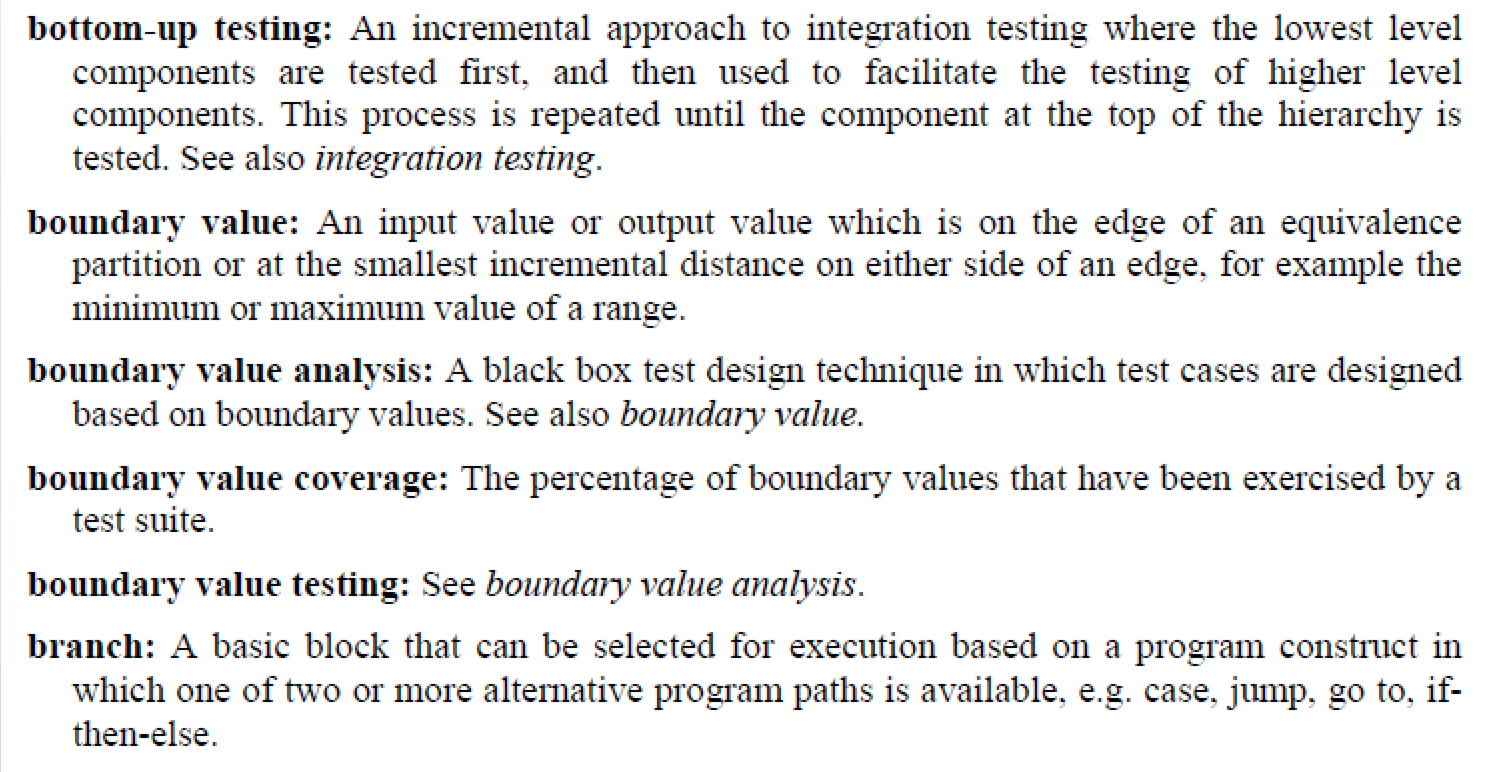
boundary value analysis: A black box test design technique in which test cases are designed based on boundary values. See also *boundary value*.

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

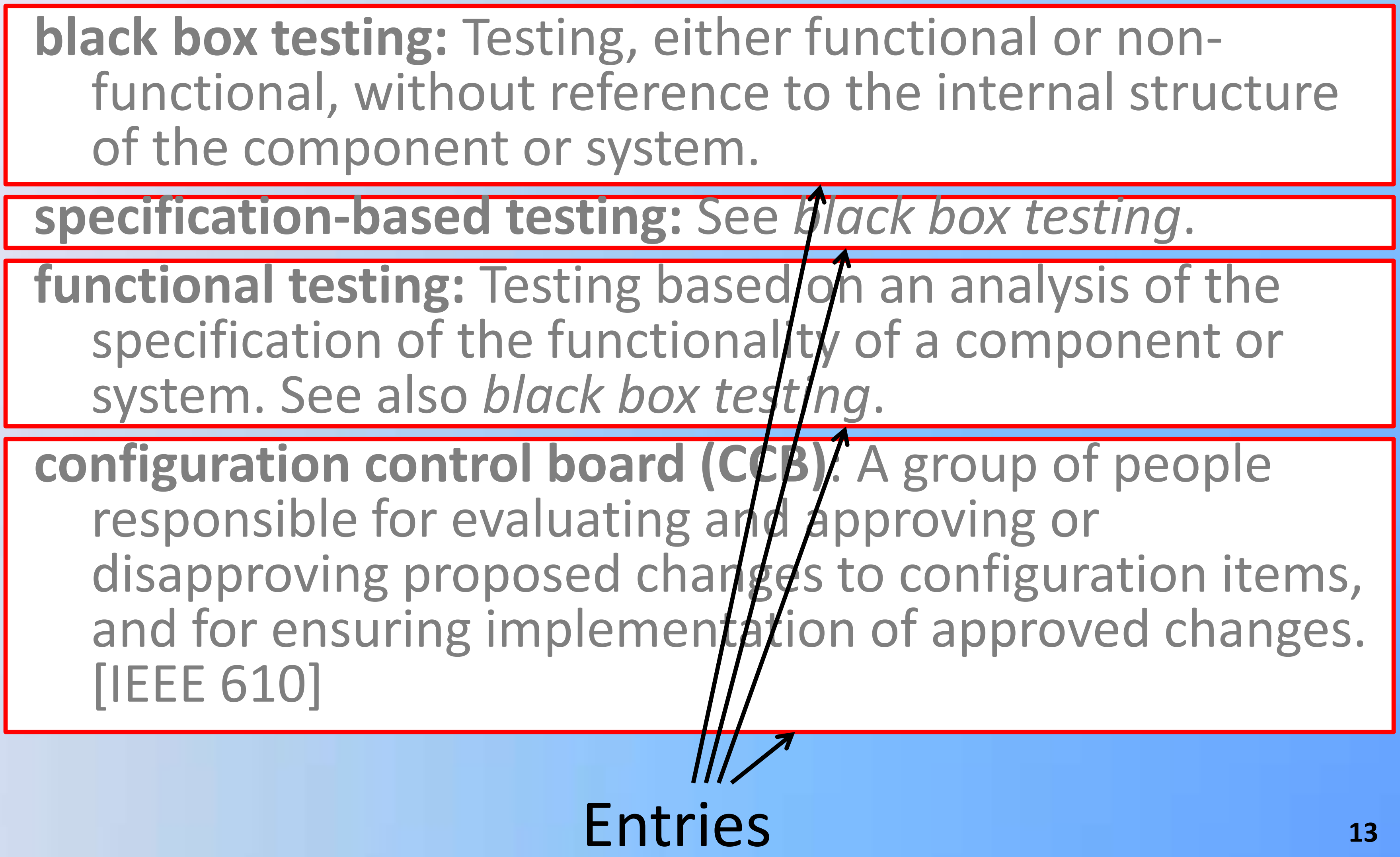
black box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.

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]

Term

Definition

Structure of the glossary

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

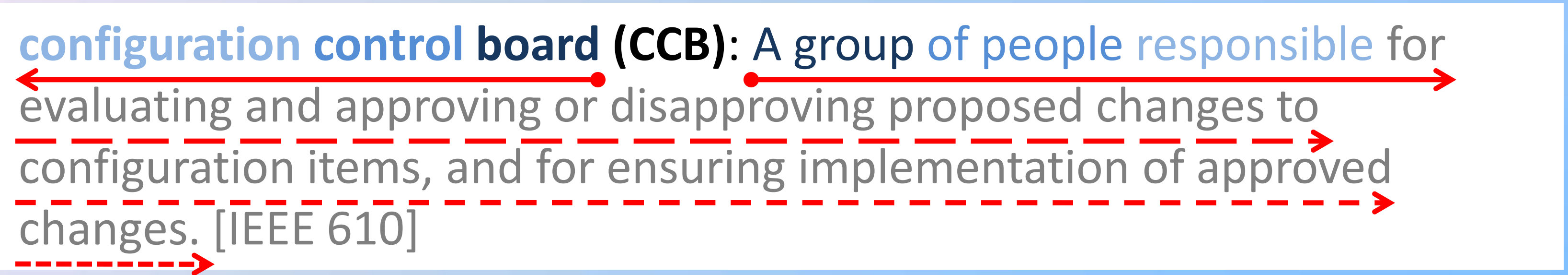
specification-based testing: See *black box testing*.

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]

Source Cross-reference Acronym Synonym

Finding of significant aspects (words)

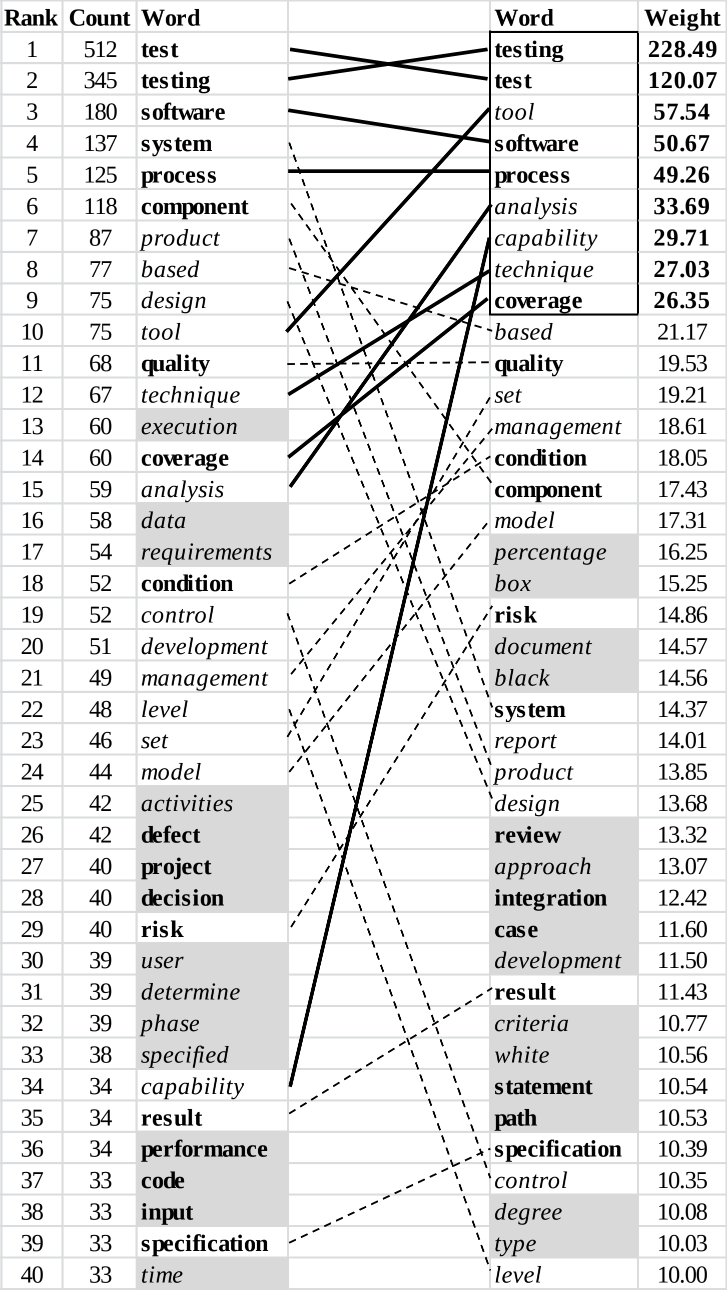


configuration control board (CCB): A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. [IEEE 610]

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.

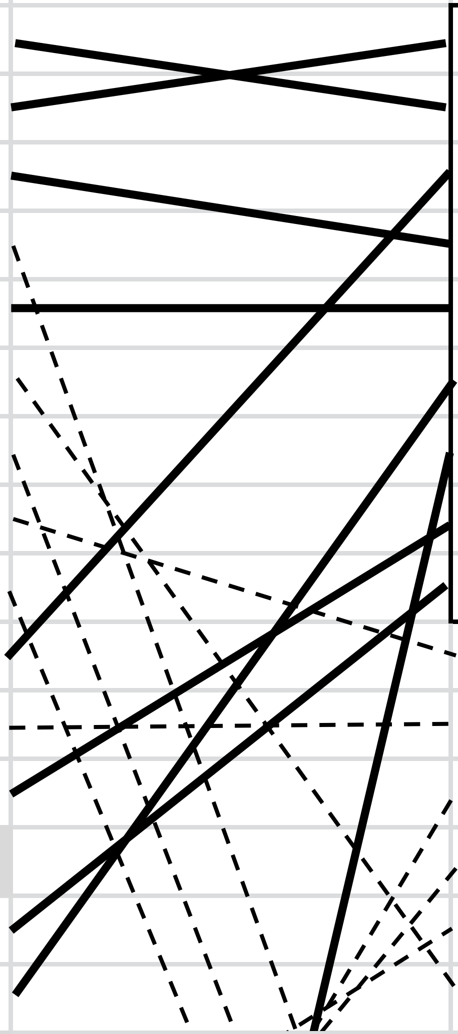
Word weighting process result (1/2)



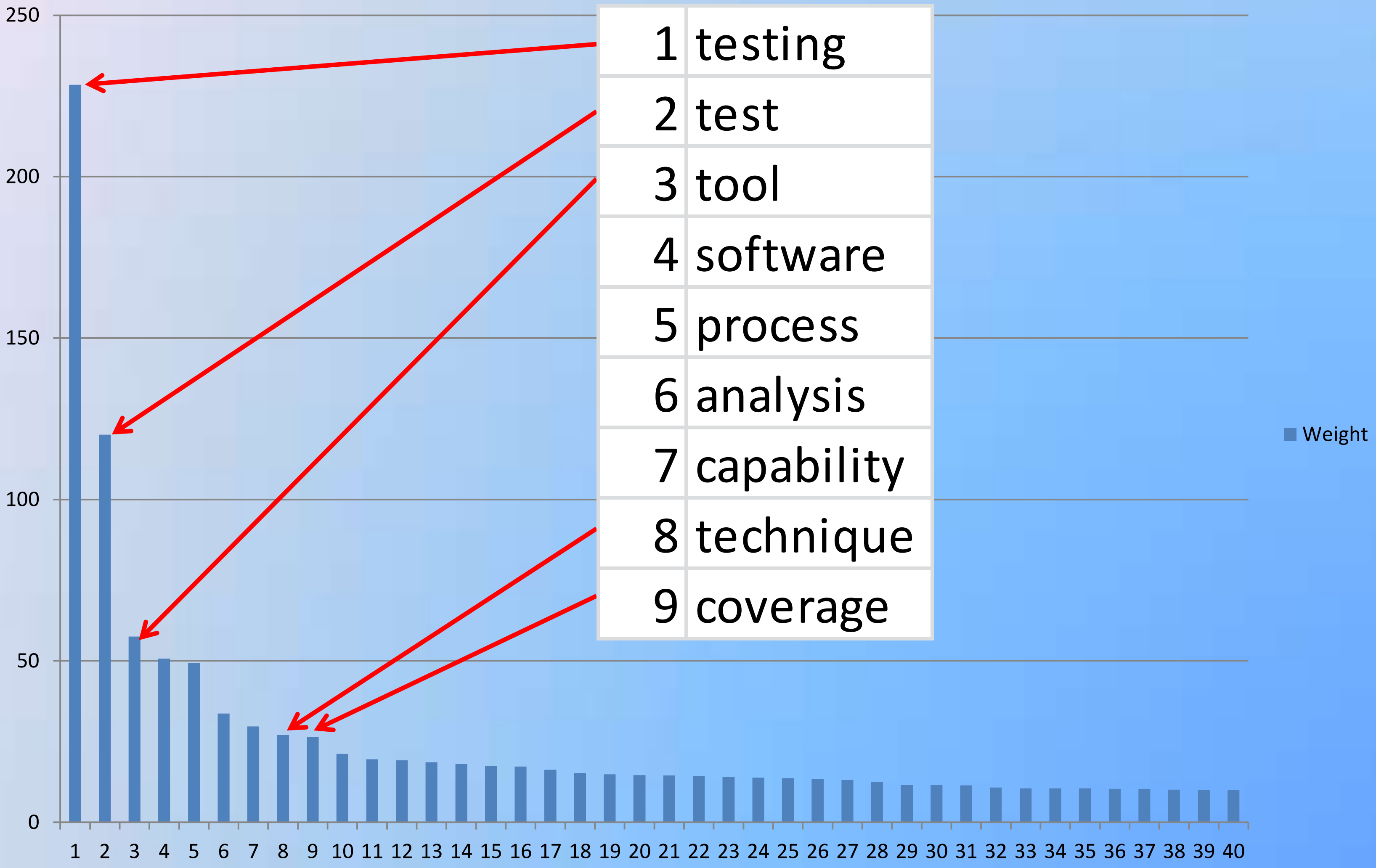
Rank	Count	Word		Word	Weight
1	512	**test**		**testing**	**228.49**
2	345	**testing**		**test**	**120.07**
3	180	**software**		*tool*	**57.54**
4	137	**system**		**software**	**50.67**
5	125	**process**		**process**	**49.26**
6	118	**component**		*analysis*	**33.69**
7	87	*product*		*capability*	**29.71**
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

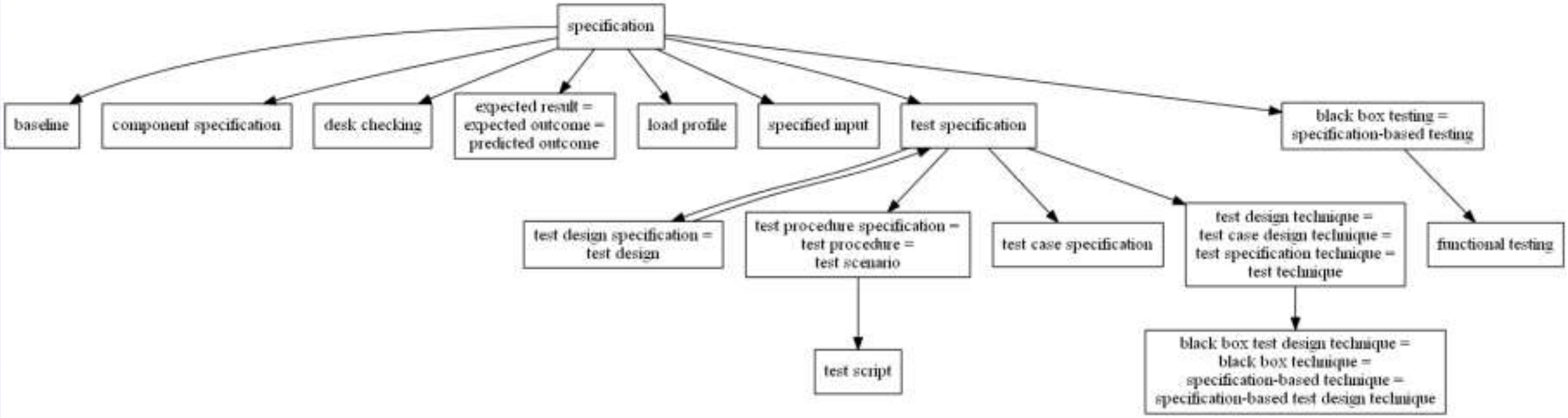


1	testing
2	test
3	tool
4	software
5	process
6	analysis
7	capability
8	technique
9	coverage

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)

Merging of aspect graphs


specification
baseline
component specification
desk checking
expected result = expected outcome = predicted outcome
load profile
specified input
test specification
black box testing = specification-based testing
test design specification = test design
test procedure specification = test procedure = test scenario
test case specification
test design technique = test case design technique = test specification technique = test technique
functional testing
test script
black box test design technique = black box technique = specification-based technique = specification-based test design technique

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

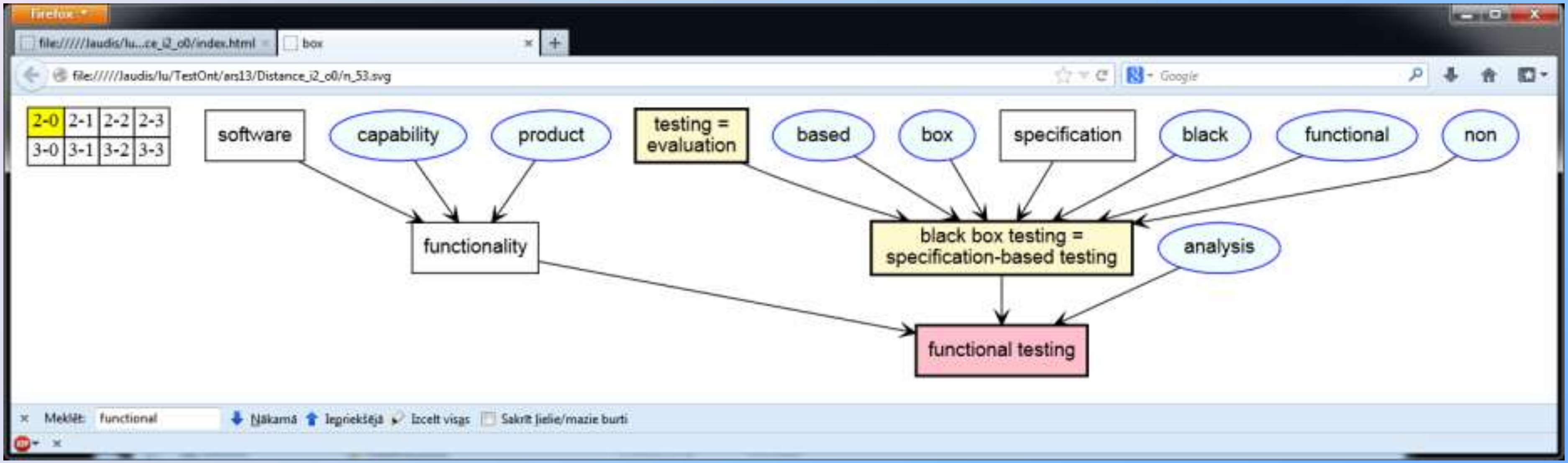
Total number of aspect graphs is **325** containing **608** unique term-nodes and **170** unique word-nodes.

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

Demonstration

- See the concept map at: http://science.df.lu.lv/tapost2013/
- Samples of demonstration screens follow

Demonstration

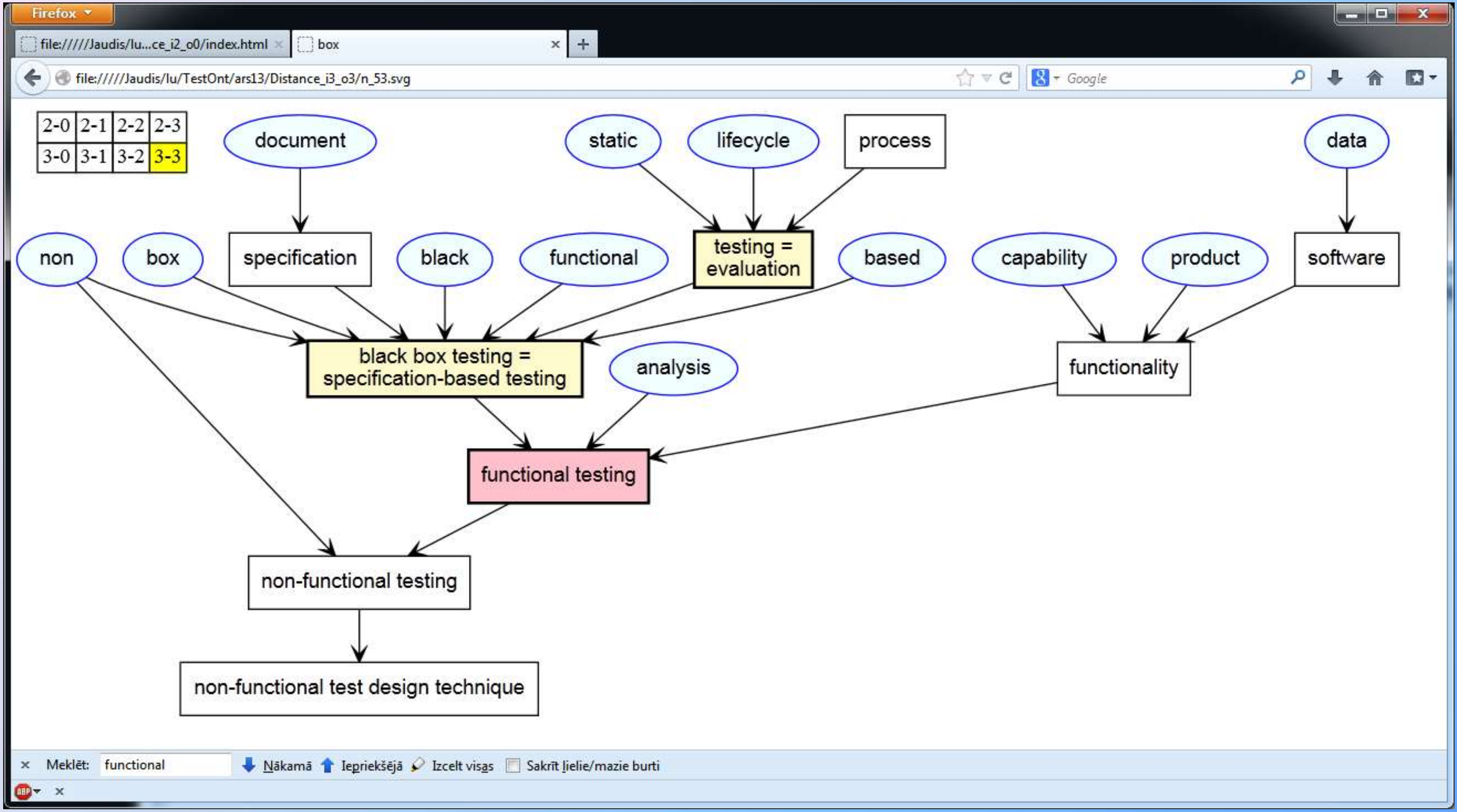


file://///laudis/lu/TestOnt/ars13/Distance_i2_o0/n_53.svg

2-0	2-1	2-2	2-3
3-0	3-1	3-2	3-3

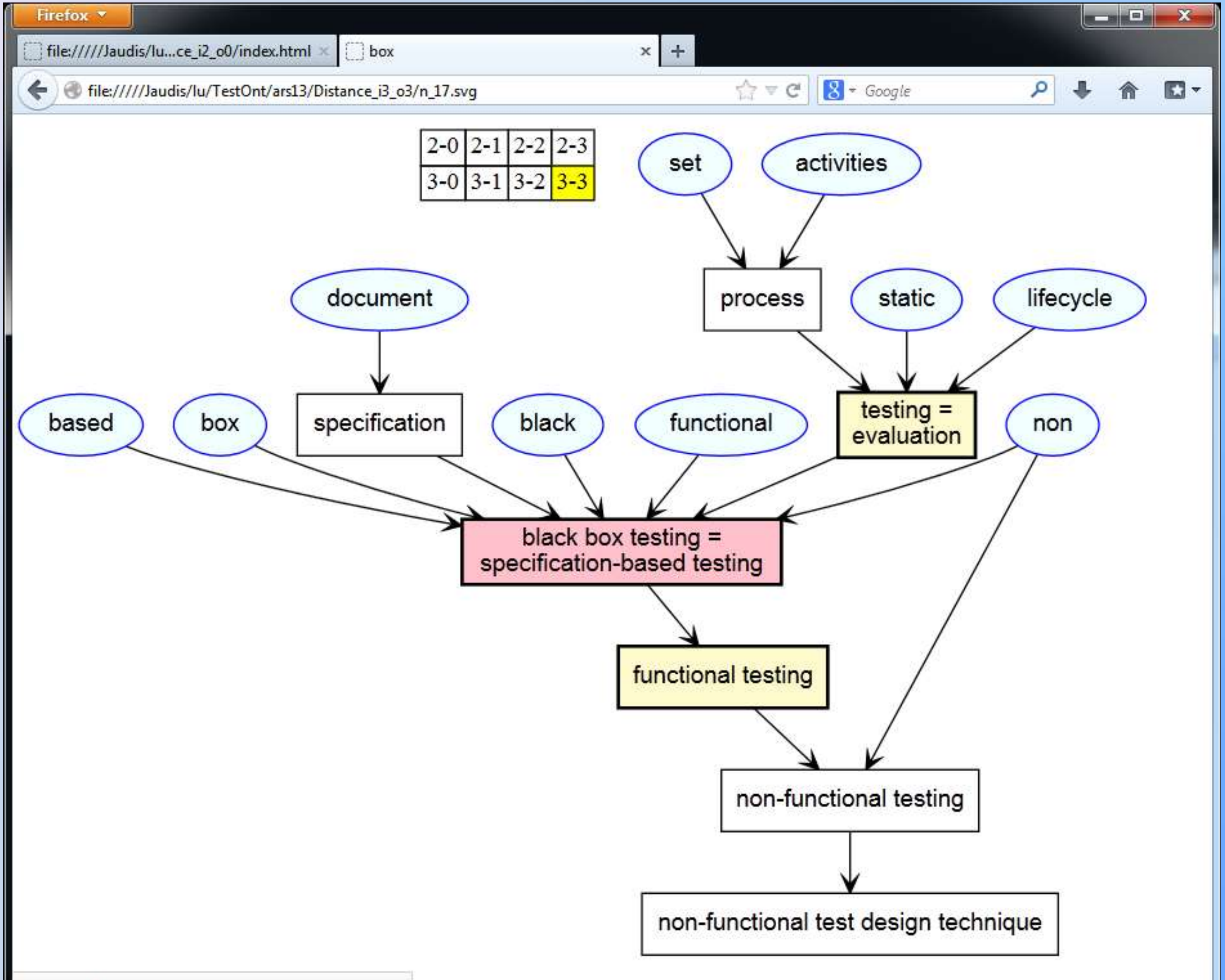
Meklēt: functional

Demonstration



2-0	2-1	2-2	2-3
3-0	3-1	3-2	3-3

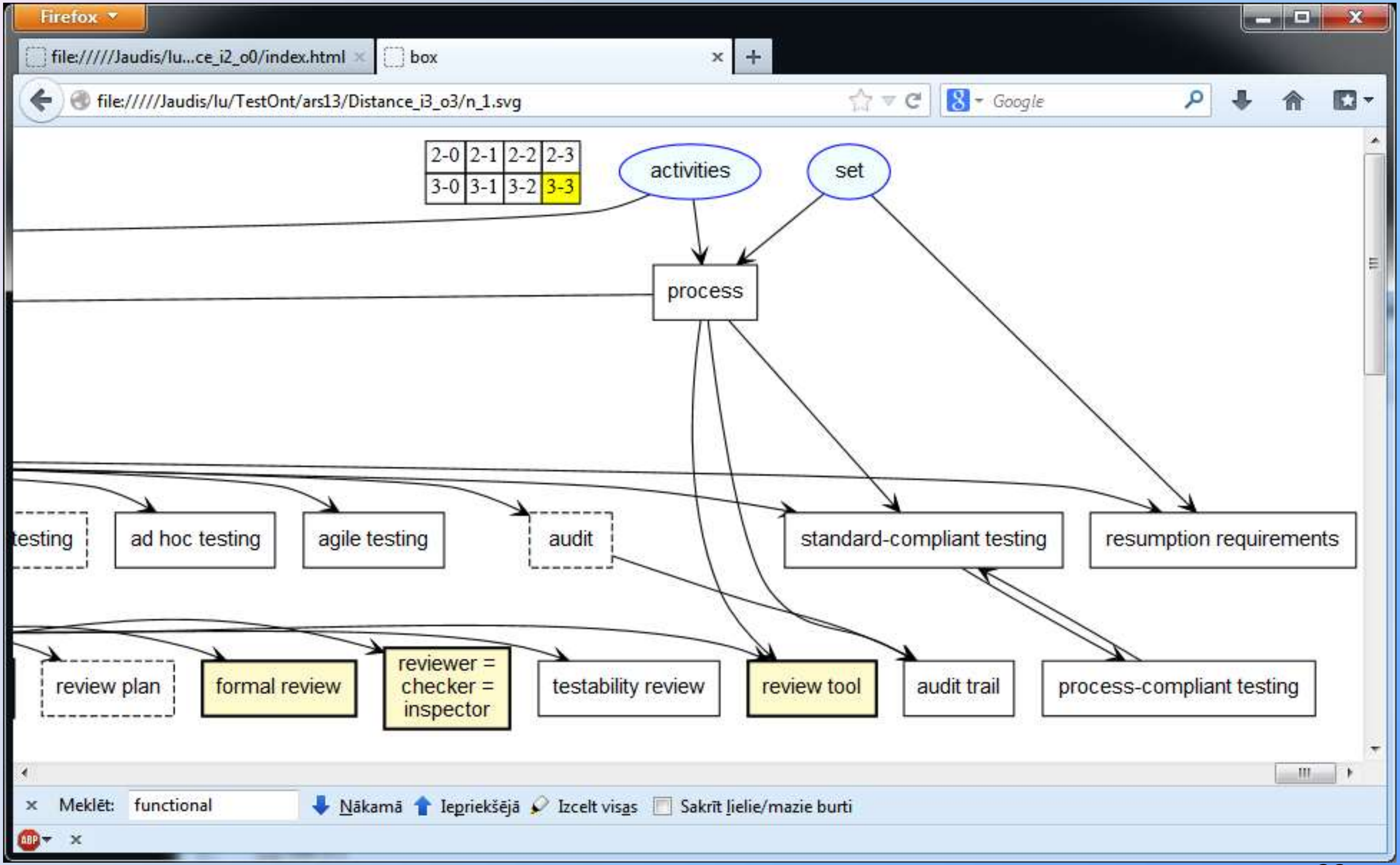
Demonstration



file://///Jaudis/lu/TestOnt/ars13/Distance_i3_o3/n_17.svg

2-0	2-1	2-2	2-3
3-0	3-1	3-2	3-3

Demonstration (small part of «*testing*»)

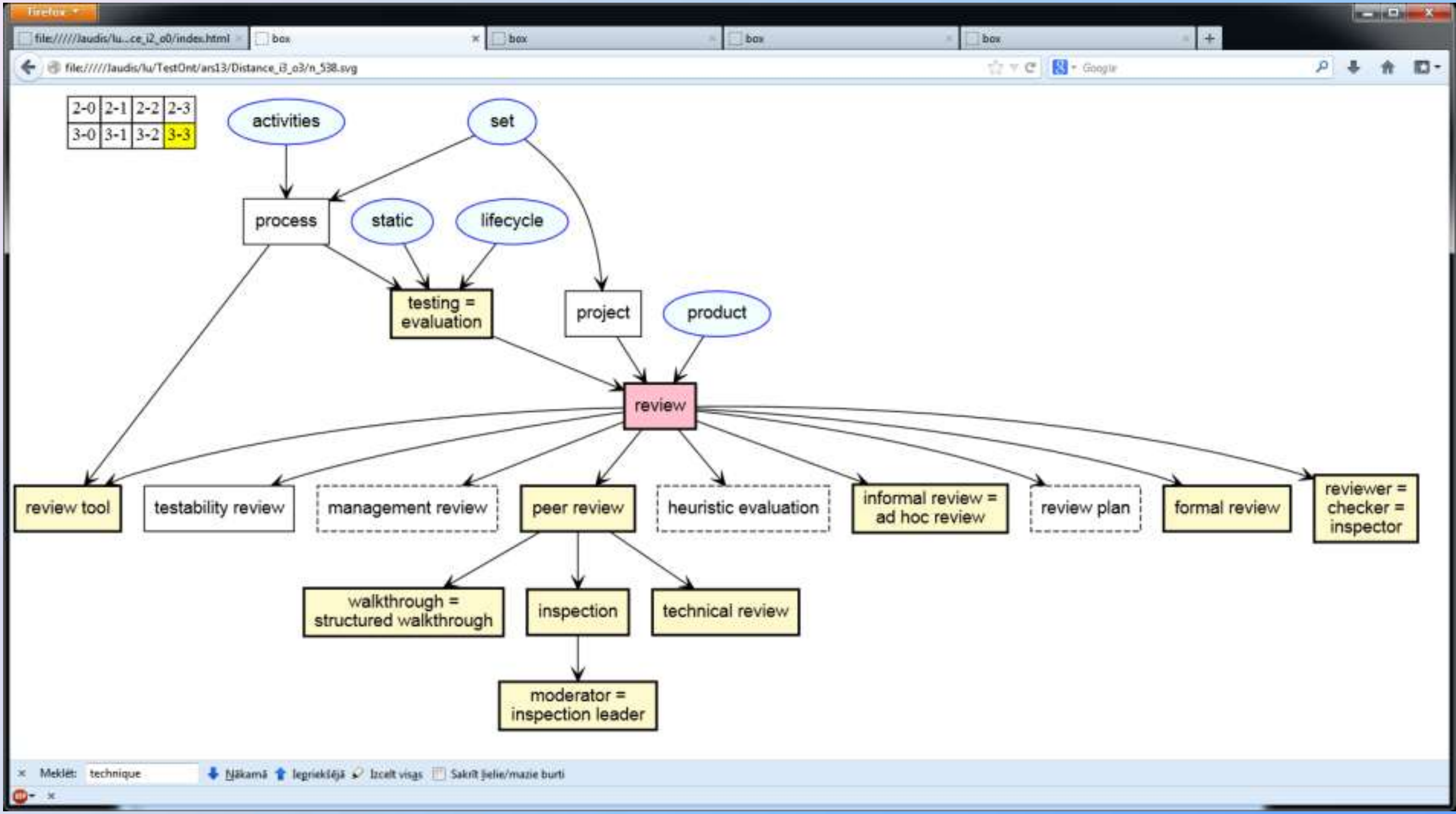


Firefox
file://///Jaudis/lu...ce_i2_o0/index.html
box
file://///Jaudis/lu/TestOnt/ars13/Distance_i3_o3/n_1.svg
Google
activities
set
process
testing
ad hoc testing
agile testing
audit
standard-compliant testing
resumption requirements
review plan
formal review
reviewer = checker = inspector
testability review
review tool
audit trail
process-compliant testing
Meklēt:
Nākamā
Iepriekšējā
Izcelt visas
Sakrīt lielie/mazie burti

2-0	2-1	2-2	2-3
3-0	3-1	3-2	3-3

Meklēt: functional

Demonstration



2-0	2-1	2-2	2-3
3-0	3-1	3-2	3-3

Conclusions

- Simple concept map for software testing domain is automatically generated from glossary
- It is only a prototype!
  - Use it on your own risk! ☺
- Thank to my colleagues (asoc. prof. Uldis Straujums and student Dainis Romans)
- We are improving our algorithms, new versions of concept map will be published
- Any criticism, proposal, suggestion is welcome

Thank you very much for your attention

Complementary material is available at http://science.df.lu.lv/tapost2013/