Table. Issues of testing

1. Major documents of development and testing
Examples: Requirements specification, functional specification, technical specification, and code.

2. Development process and test process integration
Process model examples: Linear-sequential model (waterfall), iterative/incremental/evolutionary models, formal methods model, Extreme Programming (XP) model, Agile Software Development model, etc.
Integration model examples: Software Development Technologies U model, V model, etc.

3. Testing as a part of quality systems
Examples: International Organisation for Standards (ISO) quality management standards ISO 9000-9003, IEEE-standards, etc.

4. Testing and capability-maturity models
Examples: Software Engineering Institute (SEI) at Carnegie Mellon University, software process capability maturity model (CMM).
ISO Work Group 10 (Software Process Assessment), software process improvement and capability determination (SPICE).
Illinois Institute of Technology, testing maturity model (TMM).

5. Metrics and testing
Examples: Measuring quality, for example measuring of correctness, integrity, maintainability, usability, and testability. Software metrics in standards, for example Standard for Software Productivity Metrics, IEEE/ANSI (Standard 1045-1992), Standard for a Software Quality Metrics Methodology, IEEE/ANSI (Standard 1061-1992), and Standard Dictionary of Measures to Produce Reliable Software, IEEE/ANSI (Standard 982.1-1988). Complexity metrics, for example lines of code (LOC), function points, and McCabe's complexity metric.

6. Standardization and testing
Examples: Software development and testing standards.

7. Risk analysis and testing
Examples: Risk-based testing.

8. Timing and testing
Examples: Time schedule problems when standardization, development, and testing glide to parallel phases.

9. Testing as a part of software quality assurance
Examples: Testing is part of software quality assurance (SQA). SQA monitors both the software and the development processes. Software quality assurance is described in the Standard for Software Quality Assurance Plans, SQAP, IEEE/ANSI (Std 730-1989).

10. Configuration management and testing
Examples: Configuration management identifies the change and controls it. Testing software locates under the control of configuration management. Configuration management is described in standards Guide to Software Configuration Management (Reaff. 1993), and in Standard for Software Configuration Management Plans, 1990(Std 828-1990).

11. Software testing strategies
Examples: Software testing strategy consists for example of test planning, test case design, test execution, result data collection and evaluation.
Testing strategies: In white box testing tests are derived from internal design specification or code. In black box testing tests are derived from functional design specification or requirements.

12. Testing methods
Examples: White box testing contains, for example basis path testing, control structure testing, condition testing, branch testing, domain testing, data flow testing, and loop testing.
Black box testing contains, for example transaction flow modelling, finite state modelling, data flow modelling, timing modelling, equivalence partitioning, boundary value analysis, comparison testing, orthogonal array testing, error guessing, cause-effect graphing, and syntax testing.

13. Testing is verification plus validation
Examples: Verification ensures that we are building the product right. Checklists are the tools of the verification. Validation ensures that we are building the right product.
Verification and validation is part of software quality assurance (SQA).
Testing is divided, for example to unit testing, integration testing, usability testing, function testing, system testing, and acceptance testing.

14. Testing object-oriented analysis, object-oriented design, object-oriented programming
Examples: Object oriented methods, for example the Booch method, the Rumbaugh method, the Jacobson method, the Coad and Yourdon method, and the Wirfs-Brock method.
Unified modelling language (UML) combines the Booch method, the Rumbaugh method and the Jacobson method. Tests can be generated from the UML description.

15. Object-oriented testing
Examples: Testing of object-oriented analysis and object-oriented design brings new features compared to structured analysis and design. In black box testing test cases are derived from the object-behaviour model and from event flow diagram. White box testing methods can be applied to the operations defined for a class.

16. Testing with formal methods
Examples: Formal methods allow more complete and consistent specification, than conventional or object-oriented methods. Mathematical specification can be analyzed.

17. Cleanroom software engineering and testing
Examples: Cleanroom software engineering is a process with mathematical verification of correctness and statistical quality certification as a part of the testing.

18. Other software engineering practices and testing
Examples: Faults are minimized by efficient reuse of components. Environments include, for example OMG/CORBA, Microsoft COM, and Sun JavaBeans Components.

19. Software engineering and testing tools (automation)
Examples: Tools include, for example metrics and management tools, quality assurance tools, software configuration management tools, integration and testing tools, static analysis tools, dynamic analysis tools, and test management tools.

20. Testing tools (automation)
Examples: Categories of testing tools are, for example tools for reviews and inspections, tools for test planning, tools for test design and development, tools for test execution and evaluation, and tools for test support.

21. Organization for software testing
Examples: Developers and testers work as a team.
22. Testing systems
Examples: Development of testing systems, configuration and integration of the measuring equipment, simulation, and analysis.
23. Testing and maintenance
Examples: Faults that pass testing.

Table. Results of the specialist group (rating).

1. Testing automation and testing tools. (39)

- Description: This issue covers the methods and tools for automated software testing. The issue is widely discussed in the literature. Dustin, Rashka, and Paul discuss extensively the issue (Dustin, Rashka et al. 1999). Poston discusses (Poston 1996) how specification-based testing can be automated.
- Reasoning: The informants felt that this issue is important and there is potential. They said also that the automation makes it possible to adapt to the demand for the testing capacity. Testing automation and the testing tools are efficient in the repetitive situations (e.g., regression testing). Also programmers felt testing reasonable when they have automated tools to write unit level tests.
- Opposite opinions: Informants felt that testing automation and testing tools do not solve the problems of earlier phases. They also felt that there are not enough results.
- Conclusion: Informants feel that this issue is important because the cost savings are measurable and the mature applications exist, but automation and tools do not fit in every case.

2. Standardization (37)

- Description: This issue covers, for example applying standardization, standardization of the interfaces, the testing standards. Time schedule problems when standardization, development and testing glide to the parallel phases. The field of standardization is wide; Moore notes (Moore 1998) that in software engineering more than 300 standards are developed and maintained by more than 50 different organizations.
- Reasoning: Informants stated that standardization is the precondition of testing automation and standardization raises the quality of software.
- Opposite opinions: Informants felt that standardization is important work, but the investment is really big and the repayment period is long. Also programmers feel that obeying standards is boring work.
- Conclusion: Standardization is closely linked to the automation of testing.

3. Process development (30)

- Description: This issue covers, for example coupling of the development process to the testing process, artefacts and interaction between the processes, and measurements. Osterweil et al. discuss this issue (Osterweil, Clarke et al. 1996). Many models have been suggested. One of the models is the Software Development Technologies U model (Kit 1995).
- Reasoning: Informants said that daily testing of the results is important and the earlier the faults are found, the more money they save. Measurements they commented by saying that measurements are important, but we should measure the things not the people.
- Opposite opinions: Informants felt that this issue is already in a rather good condition and there is no sense in measuring the processes.
- Conclusion: The development process and the testing process should be coupled together, but it is unclear how this should be implemented.

4. Formal methods (28)

- Description: This issue covers, for example development and implementation of the less faults generating methods in software engineering (e.g., formal methods and the cleanroom in the software engineering or extensive reuse of the software components), and methods that minimize testing by producing higher quality code. The issue is discussed in articles concerning software quality. Voas discusses the issue (Voas 1999).
- Reasoning: Informants said that the formal methods are efficient in verification of the functional requirements especially in the critical systems. Formal methods combined with harmonization of all the processes are efficient.
- Opposite opinions: Informants felt that this issue is already in a rather good condition. Investing in the less faults generating methods does not bring big savings in testing. Formal methods were already tested, but anything remarkable was not reached.
- Conclusion: Formal methods are important, but their application area is limited because of the costs.

5. Testing methods and strategies (28)

- Description: This issue covers methods, techniques, and strategies that are used in testing. The issue is discussed widely (Beizer 1990).
- Reasoning: Informants said that testing methods and strategies are an important part of development and testing, there exists potential and methods are an interesting research area. This is important area because the amount of unnecessary testing is big.
- Opposite opinions: Informants felt that the savings caused by developing this issue are small and rather much research exists in this area.
- Conclusion: Methods and strategy development is needed.

6. Testing systems (27)

- Description: This issue covers, for example development of the testing systems, more extensive utilisation of testing equipment or replacement with the cheaper devices, configuration and integration of measuring equipment, simulation, and analysis. The problems of the testing laboratories are combined in this issue. Testing systems are expensive or testing can be done only with customer's equipment (e.g., paper machine or telecommunication network). This issue is taken from the interviews with the personnel of the testing laboratories.
- Reasoning: Informants said that the labour costs can be reduced by developing testing systems. Development of the testing systems is applied work of the industry.
- Opposite opinions: Development of the testing systems is daily work because of the changing technology. Rather much

research exists. More efficient is to develop light human methods.

- Conclusion: Development of the testing systems seems to be a serious problem for the testing laboratories, but it is not a big problem for all the industry.

7. Distinct issues (UML, TTCN, etc.) (26)

- Description: The distinct sectors of testing are combined in this issue, for example test generation from the UML description, the TTCN testing language, configuration management (etc. distinct sectors). Testing with the Unified Modelling Language (UML) is discussed (Dai, Grabowski et al. 2004). Testing and Test Control Notation (TTCN) language is discussed (Vassiliou-Gioles, Din et al. 2004).
- Reasoning: Informants estimated that UML connected with the TTCN language is potential in automated testing. Test generation from the UML description is important. Configuration management and traceability are essential especially with critical software. TTCN-3 and UML 2.0 are interesting.
- Opposite opinions: Informants estimated that UML connected with the TTCN language is applicable in limited testing environments. Software development is going to an unstructured direction.
- Conclusion: Test generation from the UML description and the TTCN language are new and growing issues.

8. Test results processing (24)

- Description: This issue combines the problems of the results processing. The test results processing system helps in avoiding unnecessary testing and in reusing existing results. This issue is taken from the interviews with the personnel of the serial production industry.
- Reasoning: Informants said that an efficient test result processing system is fundamental when using automated systems. This is an important area.
- Opposite opinions: Test results are processed and analyzed, but how the results are used. Rather much research exists. No big innovations exist in this area. This is not a problem.
- Conclusion: Informants seem to be for and against. Efficient test results processing systems are needed in the serial production industry.

9. Finding faults that pass testing (16)

- Description: This issue combines the questions of the maintenance phase, for example fault locating systems for the field service. Complete testing is impossible; the issue is discussed (Myers 1976).
- Reasoning: Some importance exists in selected areas.
- Opposite opinions: Tools for the maintenance phase should be included into the product. The benefit for finding the faults in the maintenance phase is small compared to the planning phase. No need exists for the maintenance phase systems.
- Conclusion: Feedback from the maintenance to the planning and the testing is important.