Teknillinen korkeakoulu Tietotekniikan osasto
Tietojenkäsittelyopin laboratorio A
Helsinki University of Technology Department of Computer Science and Engineering
Laboratory of Information Processing Science A
Espoo 2005 TKK-TKO-A43

NEEDS ASSESSMENT OF SOFTWARE SYSTEMS GRADUATES

Sami Surakka

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Computer Science and Engineering for public examination and debate in Auditorium T1 at the Helsinki University of Technology (Espoo, Finland) on 9th of December, 2005, at 12 noon.

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY

Helsinki University of Technology
Laboratory of Information Processing Science
P.O.Box 5400
FI – 02015 HUT
URL: http://www.cs.hut.fi/english.html

ISBN 951–22–7950–9 (printed)
ISBN 951–22–7951–7 (PDF)
ISSN 1239–6885 (printed)
URL: http://lib.tkk.fi/Diss/2005/isbn9512279517/

Otamedia Oy
Espoo 2005

HELSINKI UNIVERSITY OF TECHNOLOGY **http://www.hut.fi/**	**ABSTRACT PAGE**
Department/laboratory and Internet address Department of Computer Science and Engineering Laboratory of Information Processing Science http://www.cs.hut.fi	Publisher Helsinki University of Technology Department of Computer Science and Engineering Laboratory of Information Processing Science
Author: Sami Surakka	
Title: Needs assessment of Software Systems graduates	

Abstract

The research problem of the present thesis was: What technical skills do graduates from specialization in Software Systems need? Triangulation; that is, several research methods and data sources were used to solve this problem. The largest part of the thesis consisted of three questionnaires where Finnish software developers ($N = 11$), professors and lecturers ($N = 19$), and Master's students ($N = 24$) evaluated the importance of 42 subjects and skills such as discrete mathematics and object-oriented programming.

The second largest part of the thesis comprised two content analyses of job advertisements targeted at software developers. A trend analysis for the years 1990–2004 and a cross-sectional analysis of the year 2004 were conducted. In both analyses, the purpose was to find the most common technical skills sought in American job advertisements.

In addition, four smaller content analyses were conducted. Documents for these content analyses were the degree requirements of 31 top-level American research universities, and the internship reports, course catalog, and Master's theses of the Helsinki University of Technology. A concept analysis of the concept "software systems" was also carried out.

The main contributions of the present thesis are as follows:

- The thesis is so far the most versatile triangulation in the area in question. In particular, the content analysis of American degree requirements and the concept analysis of "software systems" were novel approaches.
- The thesis provided findings that the requirements for software developers have required greater versatility during the past 15 years. Todd, McKeen, and Gallupe reported similar change in 1995 for the 1970–1990 period. However, it was interesting to know if this trend had continued after 1990.
- According to the summarized results, the following technical subjects or skills were evaluated as being important: compilers, concurrent programming, data structures and algorithms, database management systems, distributed systems, object-oriented programming, operating systems, procedural programming, and software architectures. Most of these subjects or skills had already been reported as being important for software developers, for example, in the survey conducted by Lethbridge in 1998.
- The importance of physics and continuous mathematics was evaluated as being low. Previously, Lethbridge reported similar findings.
- In the job advertisement analyses of the present thesis, technical skills were analyzed in a more detailed manner than in the previous analyses on average. In particular, some results concerning distributed technology skills were new and more detailed than previously published.

Keywords

needs assessment, software developer, specialization in Software Systems, technical skills

Place	Year	Number of pages	Language of publication	Language of abstract
Espoo	2005	242	English	English

ISBN (printed)	ISSN and number or report code (printed)
951–22–7950–9	1239–6885, TKK-TKO-A43
ISBN (electronic)	**ISSN and number or report code (electronic)**
951–22–7951–7 (PDF)	—

URL (Internet address): http://lib.tkk.fi/Diss/2005/isbn9512279517/

Preface

The present research was conducted in the Laboratory of Information Processing Science at the Helsinki University of Technology where I have worked since 1994. However, my interest in the development of computer science education began in the late 1980s when I was still an undergraduate student. In 1989–1990, I was a board member of the student organization Data Guild where my position was freely translated Master of Studies ("opintomestari" in Finnish). I still remember the strong opinions I held then concerning the necessity for mathematics and physics. In a way, needs assessment is a scientific way of clarifying some issues that I instinctively knew already in 1990.

In 1999, the Laboratory of Information Processing Science was selected as a center of excellence in higher education in Finland for the 2001–2003 period. The Computer Science Education Research Group of the laboratory was launched in 2000, also including members outside the laboratory. Both these events were beneficial for the present thesis, on the one hand for the funding and on the other for motivation and collaboration.

I thank my supervisor Professor L. Malmi and instructor Professor Emeritus V. Meisalo for guidance, and the preliminary examiners Professor P. Schein and Associate Professor M. Ben-Ari for their valuable comments. In addition, I would like to thank Doctors S. Törmä and E. Nuutila for participating in the planning of the questionnaire targeted at software developers, Associate Professor T. C. Lethbridge for submitting the data from his survey, Lecturer I. Mellin for commenting on the statistical testing, and Valtasana Ltd. and Ruth Vilmi Online Education Ltd. for the revision of the English language.

Espoo, November 2005

Sami Surakka

Contents

Abstract 3

Preface 5

Contents 7

Part I: Overview 11

1 Introduction 11
- 1.1 Background 11
- 1.2 Research problems and objectives 13
- 1.3 Scope of the thesis 15
- 1.4 Contributions 16
- 1.5 Style and structure of the thesis 17

2 Literature review 19
- 2.1 Context in educational sciences 19
- 2.2 Computer science education as a research area 23
- 2.3 Some approaches to organize curricula 25
- 2.4 Papers in major computer science education publications 27
- 2.5 Needs assessments in the field of information technology 29
- 2.6 Concept analysis of “software systems” and content analysis of degree requirements 43
- 2.7 Normative studies 45
- 2.8 Cognitive skills 47
- 2.9 Necessary skills in the future 49

3 Research methods 51
- 3.1 Needs assessment 51
- 3.2 Triangulation 52
- 3.3 Concept analysis 54
- 3.4 Delphi method 55
- 3.5 Content analysis 56
- 3.6 Survey 57
- 3.7 Trend analysis 57
- 3.8 Single cross-sectional study 58
- 3.9 Case study 58
- 3.10 Statistical analysis 59
- 3.11 Evaluating validity and reliability 60

Part II: Specialization in Software Systems defined by courses 62

4 Concept analysis of "software systems" 63
4.1 Research method 63
4.2 Results 65
4.3 Evaluation 69

5 Content analysis of degree requirements 71
5.1 Research method 71
5.2 Results 73
5.3 Evaluation 76

6 Triangulation: Concept analysis of "software systems" versus content analysis of degree requirements 78

Part III: Questionnaires 80

7 Delphi study targeted at software developers: Technical skills 81
7.1 Research method 81
7.2 Results 84
7.3 Evaluation 89

8 Delphi study targeted at professors and lecturers 90
8.1 Research method 90
8.2 Results 90
8.3 Evaluation 97

9 Survey targeted at Master's students 99
9.1 Research method 99
9.2 Results 100
9.3 Evaluation 106

10 Delphi study targeted at software developers: Cognitive skills 109
10.1 Research method 109
10.2 Results 111
10.3 Evaluation 116

11 Triangulation of questionnaires 118
11.1 Results of three questionnaires 118
11.2 Professors and lecturers' explanations for differences 121
11.3 Other explanations for differences 122
11.4 Correlation of results 122
11.5 Main findings of Part III 123

Part IV: Job advertisement analyses 124

12 Trend analysis of job advertisements 125
12.1 Research method 125
12.2 Results 128
12.3 Evaluation 130

13 Cross-sectional content analysis of job advertisements 132
13.1 Research method 132
13.2 Results 133
13.3 Evaluation 140

14 Triangulation of job advertisement analyses 144
14.1 Number of required skills 144
14.2 Proportions of various skills 144
14.3 Random changes 146

Part V: Viewpoint of basic studies 148

15 Description of case example 149
15.1 Old degree structure 149
15.2 Generality of degree program 151

16 Content analysis of Master's theses 153
16.1 Research method 153
16.2 Results 153
16.3 Evaluation 154

17 Content analysis of internship reports 156
17.1 Research method 156
17.2 Results 157
17.3 Evaluation 157

18 Content analysis of course prerequisites 158
18.1 Research method 158
18.2 Results 159
18.3 Evaluation 160

19 Triangulation for basic studies 161

Part VI: Putting it all together 162

20 Summative triangulation 162

21 Discussion 167
21.1 Comparison with previous research 167
21.2 Evaluation of the thesis 188
21.3 Conclusions 190
21.4 Recommendations 192
21.5 Professional or academic emphasis in the curriculum? 198
21.6 Admission procedures 199
21.7 Future research 200

Part VII: Case of Helsinki University of Technology 205

22 Related work 205

23 Description of case example 206
23.1 Scope by structure of Bachelor's degree 206
23.2 Scope by structure of new Master's degree 209
23.3 Generality of specialization in Software Systems 210

24 Comparison 212
24.1 Offered specializations 212
24.2 Required courses of specialization in Software Systems 214

25 Recommendations 216
25.1 New modules 216
25.2 Requirements of A2 and A3 modules 217

Part VIII: Summary of the thesis 221

References 226

Appendices 234
Appendix A: Author's publications related to the present thesis 234
Appendix B: Software development strategies 236
Appendix C: Selected institutions and degree programs 237
Appendix D: Planning of Question 15 238
Appendix E: Conversion of Lethbridge's items 240

Part I: Overview

This part is an overview of the thesis. The introduction of the thesis, related literature, and the research methods used are presented here.

1 Introduction

Curriculum design is a complicated issue including many different points of view that often contradict each other. Designing a computer science (CS) curriculum is even more difficult because of the rapid evolution of the whole field. Therefore, joint international efforts such as Computing Curricula 2001 (Engel & Roberts, 2001) have been carried out to build general frameworks for designing and comparing different curricula. Computing Curricula 2001 is useful for designing introductory and intermediate studies because it provides the recommendation of a common core (p. 17), sample approaches for the intermediate level (pp. 36–39), and three different sample curricula (pp. 45–53).

However, Computing Curricula 2001 is less useful for designing specializations because it is limited to undergraduate programs (p. 1) whereas specializations are more typical in graduate programs. Specializations were covered in the report only on a general level but no recommendations for specific specializations were provided. They wrote about the subareas of computer science as follows (p. 52): "However, the number of electives should be large enough to provide depth in at least one subarea of computer science. We propose a minimum of three advanced electives, …" One can assume that the word "subarea" refers to approximately the same concept as the word "specialization."

In the present thesis, the curriculum design problem was approached by performing a needs assessment. Needs assessment as a research method is presented later in Section 3.1. Section 2.1 explains why needs assessment was selected from the various evaluation methods available.

1.1 Background

The present thesis originated from the need to better understand what topics and skills should be included in the Master's level education of specialization in Software Systems at the Helsinki University of Technology (later often referred to as "the institution"). The Laboratory of Information Processing Science is responsible for this specialization. The same laboratory is also responsible for the basic-level education in programming that was selected as a center of excellence in higher education in Finland for the periods 2001–2003 and 2004–2006 (Moitus, 2000; Parpala & Seppälä, 2003).

In the laboratory, there has been long-term development in and self-evaluation of the basic programming courses as part of the application process for a center of excellence in higher education. However, there has not been a similar development effort for more advanced studies (study years 3–5). One purpose of the present thesis was to direct more effort at the development of the advanced studies.

At the beginning of the thesis project, the present research was planned as a case study. However, during the thesis work this plan was gradually dropped for various reasons. For example, American job advertisements were used instead of Finnish job advertisements in order to get bigger samples and results that would be interesting for a wider readership. At the end of the project, the proportion of the case-specific part was not sufficiently large to allow the present thesis to be classified as a case study.

1.1.1 Professionally or academically oriented curricula?

An interesting question is whether university education in computer science should emphasize scientific considerations or the needs of industry. Hirmanpour, Hilburn, and Kornecki (1995, p. 126) wrote about the design of computer science curricula:

> A principal issue in the design of computer science curricula is the designer's view of computer science as a discipline [6]. Is it a science or engineering discipline? The design of most curricula is based on some combined view of computer science as having both science and engineering components. Curriculum 1991 argues that every computing program should emphasize three education paradigms: theory, abstraction and design [1]. In [2] curriculum development is presented in terms of a combination of choices from three different continuum: breadth versus depth coverage, passive versus active learning styles and "practice" versus "theory" abstraction levels. There is typically a balance between these three, where the emphasis depends upon the curriculum goals and objectives. For example, a curriculum intended to prepare students for graduate school would emphasize breadth of coverage, passive learning and "theory", while a professionally oriented curriculum would concentrate on depth of coverage in certain key areas, active learning styles and "practice".

The present thesis can be described as professionally oriented; that is, needs of the industry, the requirements of entry-level positions, and coverage in the particular area known as Software Systems are emphasized. This choice was not deliberately made at the beginning of the thesis project but was rather an indirect consequence of the characteristics of the case example and the

properties of needs assessment as a method. However, out of the nine data sources of the present thesis, the following three have a more academic emphasis: the degree requirements of research universities (Section 5), professors and lecturers (Sections 4 and 8), and the course catalog of the institution (Section 18). All the data sources of the present thesis are presented together later, in Figure 2 (Section 3.2).

This discussion is related to the concepts of discipline-based programs, domain-based programs, and decontextualized curricula, which are considered later in Section 2.3.

1.1.2 Soft skills versus technical skills

An on-going discussion has been whether or not soft skills are more important than technical skills for success in an information technology career. Soft skills refer to, for example, communications and project management skills whereas technical skills refer to, for example, operating systems and programming languages. According to Litecky, Arnett, and Prabhakar (2004, p. 69), research findings on this question have been contradictory. They suggested as an explanation a two-stage model of recruiting where technical skills were first used for filtering the candidates. Then, soft skills and other external factors were used for the final choice. Young and Lee (1997, p. 5) asked recruiters to rank the criteria used for evaluating information systems (IS) graduates. The order was as follows: 1. internship, 2. technical skills, 3. communication skills, 4. grade point average, and 5.–7. various soft skills. They asked what criteria recruiters used for selecting institutions for recruiting internships but they did not ask which characteristics of students were important to get an internship.

The present thesis was targeted mainly at technical skills because it was assumed that (a) technical skills were essential to get an internship and the first entry-level position, (b) on average, soft skills will become gradually more important if a person advances in his or her career and moves to management or other senior-level positions, and the fact that (c) the requirements of specialization in Software Systems at the institution were technical when the present thesis project was started.

1.2 Research problems and objectives

The present thesis is a descriptive body of research; that is, it did not have any hypothesis. The main research problem of the present thesis was the following:

> What technical skills do graduates from specialization in Software Systems need in their work after graduation?

As mentioned previously, technical skills refer to, for example, operating systems and programming languages. As will be shown later in Section 2.5, there are several previous research projects in which approximately the same problem has been investigated. However, the present research is so far the most versatile triangulation in the area in question. Triangulation means that several research methods and data sources were used to solve this problem. Triangulation as a method will be explained more in Section 3.2.

In addition to the main problem, the thesis had the following subproblems:

1. What does the concept "software systems" mean?
2. Has the number of required technical skills increased during the past 15 years in job advertisements targeted at software developers? Todd, McKeen, and Gallupe (1995) reported that the number of technical phrases in job advertisements for programmer positions increased from the mean of 2.2 in 1970 to 4.2 in 1990. Has this increase continued after the year 1990?
3. In particular, how has the number of required distributed technology skills increased? World Wide Web technology was released in 1993. After this, the number of web sites has increased rapidly. As a consequence, skills related to distributed systems should now be required more often than they were ten years ago.
4. What are the differences, if any, between the required skills of programmers, software engineers, and software developers?
5. What are the differences between entry-level and senior-level software developer positions?
6. How well do entry-level job requirements for software developers correspond with the requirements of a typical undergraduate program in computer science? This subproblem can also be classified as a planning problem, rather than as a research problem.
7. How are the needs found as the answer to the main problem different from the planned degree requirements of the institution in the academic year 2005–2006? This subproblem can also be classified as a planning problem, rather than as a research problem.

When estimated using the number of pages, over 90% of the thesis is targeted at solving the main problem and the subproblems 1–6 (Sections 2–21). The case-specific subproblem 7 is considered only in Sections 23–25.

The main objective was to solve the main research question using needs assessment and triangulation. The research methods used are presented later in Section 3. Section 3.2 explains why triangulation was selected. The additional objective was to make recommendations related to the case-specific problem.

1.3 Scope of the thesis

In this section, the scope of the present thesis is explained using the age of graduates, job titles, and typical course names. Age is relevant because some skills might be useful in the long run but less useful in an entry-level position. Job titles are relevant because the purpose is to investigate necessary skills in certain tasks. Course names are used to explain the scope because a typical reader of the present thesis might be a professor or lecturer who works at a CS department. In addition, an explanation is given as to why American data sources were used.

1.3.1 Scope by age of graduates

Next, the scope of the thesis is explained by using the age of graduates. Typical and simplified activities from university studies to pension according to the Finnish education system are presented in Table 1. From these activities, the present thesis is targeted at entry-level positions after graduation from a university. In particular, the present thesis is not targeted at internship positions or positions that are more typical of the 31- to 65-year age group.

Table 1. Typical and simplified activities from university studies to pension.

Age in years	Activity
19–25	University studies (Master's degree) and internships
26–30	Entry-level position
31–65	Mid-level, senior, or manager position
66–	Pension

1.3.2 Scope by job titles

From various information technology (IT) positions, such as those of consultants, database administrators, project managers, and systems administrators, the present thesis was targeted at software developer positions. The term "software developers" was used to denote programmers and software engineers as well. In particular, this limitation was used during the job advertisements analyses. Other possible job titles would be, for example, software specialist, software architect, and project manager, but these were not used because they were considered as being not suitable for recent graduates.

According to Gallivan, Truex, and Kvasny (2004, p. 74), Programmer/Analyst and Software Engineer were the most common IT job titles in 2001 (proportions 24% and 17%, respectively). In particular, an important motivator for the scope of the present thesis was an assumption that software developer positions are important for education because they are probably even more common as entry-level positions. That is, it was assumed

that graduates do not typically start their careers, for example, as project managers or consultants.

1.3.3 Scope by course names

Next, the scope of the present thesis is characterized by listing course names. The list is not proposed as being complete but these courses are used as examples of typical courses:

Capstone Project
Compilers
Computer Networks
Concurrent Programming
Databases
Design and Analysis of Algorithms
Introduction to Software Engineering
Operating Systems

1.3.4 Use of American data sources

In the present thesis, American data sources were used in Sections 5, 12, and 13 for practical reasons and in order to get results that would be probably interesting for a wider readership. The practical reasons were as follows:

- Finnish data were not used for these content analyses because it was assumed that a large number of job advertisements and universities might be useful in order to get large enough samples. In other words, Finland is too small a country.
- The selection was limited to English-speaking countries in order for the author of the present thesis to be able to understand degree requirements and job advertisements. From different English-speaking countries, the USA was selected because it has the greatest population. It was assumed that the number of job advertisements and universities would be greater as a consequence of the greater population.

1.4 Contributions

Next, the most important contributions of the present thesis are presented:

1. The present thesis provided findings that the requirements for software developers increased and have required greater versatility during the past 15 years. This general trend was reported apparently for the first time in 1995 for the 1970–1990 period (Todd et al., 1995). However, it was interesting to know if this trend had continued after 1990.

2. The present thesis provided supporting findings that continuous mathematics and physics are not important for software developers. Previously, Lethbridge (2000) reported similar results. These supporting results were useful because Lethbridge's methodology was criticized (Kitchenham & Pfleeger, 2002, p. 17). The necessity for these subjects was an important question because the proportion of continuous mathematics and physics is large in computer science education on average.
3. In the job advertisement analyses of the present thesis, technical skills were analyzed in a more detailed manner than in the previous analyses on average. In particular, some results concerning distributed technology skills were new and more detailed than previously published.
4. In the questionnaires of the present thesis, different programming paradigms were analyzed in a more detailed or different manner than previously. Based on the results, it was possible to conclude the order of importance of these paradigms.

The following were contributions from the viewpoint of research methods:

5. The thesis is so far the most versatile triangulation in the area in question. In particular, the content analysis of American degree requirements and the concept analysis of "software systems" were novel parts.
6. Previously, statistical tests were often used in surveys but rarely in job advertisement analyses. This was interesting because job advertisement analysis was the most common research type in this area. In the present thesis, statistical tests were also used to analyze the results of job advertisement analyses.

The author's publications related to the present thesis are listed in Appendix A.

1.5 Style and structure of the thesis

The APA Publication Manual (American Psychological Association, 2001) was used for style. The manual was chosen because it is used by the Computer Science Education journal. For vocabulary and grammar, American English was used because the APA Publication Manual is American.

The names of courses and specializations are written in initial capital letters, for example, Operating Systems and the names of subjects in lowercase letters, for example, operating systems. However, the first word of the name of a subject is written in capital letters in certain situations such as in tables: for example, Operating systems. The names of the areas of education are written in lowercase letters, for example, area of software systems or field of software engineering. The names of degree programs are written in initial capitals if they

refer to some particular program and in lowercase letters if they refer to programs in general.

Previous literature and the research methods used are presented in Part I. The specialization in Software Systems is defined or characterized by courses in Part II. In Part III, the results of three questionnaires and their triangulation are presented. Content analyses of job advertisements are presented in Part IV. Part V presents results that are more related to basic studies (years 1–2). All results are summarized, the present thesis is discussed, and recommendations are presented in Part VI. Case-specific results and recommendations are presented in Part VII. Finally, the summary of the thesis is presented in Part VIII.

The work relating to Parts II–V is deliberately presented in Section 2, rather than distributed through Parts II–V and, also deliberately, most section evaluations in Parts II–V (e.g., Section 4.3) are quite brief and cover only validity and reliability. The results are compared to previous findings; possible differences are considered later in the general discussion (Sections 21.1 and 21.2). This structure is used in order to enable easy movement from one section to another in Parts II–V. This principle is repeated at the beginning of Section 4, as this is the first section where the principle is applied. The principle is not repeated in other sections.

2 Literature review

Previous literature is reviewed in this section, which is divided into eight subsections. The purpose of the first subsection is to place the present thesis in context from the viewpoint of educational sciences. Second, computer science education is presented as a research area. Third, papers in major CS education publications are presented. Previous research is presented according to the research method or type in Sections 2.5–2.7. Section 2.8 presents previous literature on the cognitive skills of software developers. Finally, needs assessments where future skills were evaluated are presented.

Typically in each subsection, first, some relevant concepts or research results are presented and then, an explanation is given as to how these issues relate to the present thesis.

2.1 Context in educational sciences

The context of the present thesis from the viewpoint of the educational sciences is explained or characterized in alternative ways in this subsection. These topics are mainly related to curriculum evaluation, design, and research.

2.1.1 Subordinate concepts of "evaluation"

In the present subsection, the thesis is characterized using three subordinate concepts of the concept "evaluation." According to the ERIC Thesaurus (Educational Resources Information Center, n.d.), needs assessment is classified as a subordinate concept of evaluation. In the ERIC Thesaurus (ibid.), other subordinate concepts for evaluation are, for example, program evaluation and curriculum evaluation. The relationships of these concepts are presented in Figure 1.

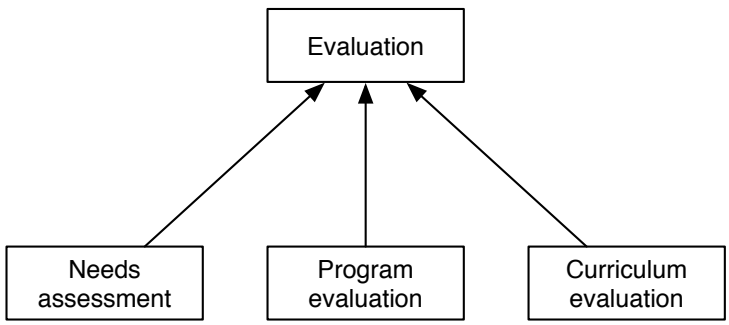

Figure 1. Relationships between some concepts.

According to the British Education Thesaurus (Marder, 1991, pp. 53, 80, 148, & 176), the definitions of these concepts are as follows:

> Evaluation
> Appraising or judging persons, organisations or things in relation to stated objectives, standard or criteria
>
> Needs assessment
> Identifying needs and deciding on priorities among them
>
> Programme evaluation
> Judging the feasibility, efficacy, value, etc. of a programme in relation to stated objectives, standards or criteria
>
> Curriculum evaluation
> Determining the efficacy, value, etc. of a specific curriculum in terms of the validity of objectives, relevance, and sequence of content and achievement of specified goals

In the present thesis, the author decided to use needs assessment as the main approach instead of other evaluation approaches for the following reasons:

- Needs assessment was very suitable for solving the main problem of the present thesis. As was mentioned previously, the main problem was: "What technical skills do graduates from specialization in Software Systems need in their work after graduation?"
- The specialization in Software Systems can be considered as being more industry oriented than some other common specializations in CS programs such as Artificial Intelligence. As a consequence, it was a possible and straightforward choice to gather data concerning industry needs, for example, using job advertisements.
- Only one specialization of the degree program was selected as the target of the evaluation, not the whole program. Therefore, program evaluation was not a suitable approach.
- The case-specific Part VII of the present thesis can be classified as curriculum evaluation when the results of the needs assessment will be used to evaluate the specialization in Software Systems at the institution. However, curriculum evaluation is not the main approach of the present thesis because the case-specific part is a moderate portion of the entire thesis and not among its main contributions.

2.1.2 Research paradigms in education

In this subsection, the present thesis is considered from the viewpoint of research paradigms in education. Husén (1994, p. 5051) wrote:

> The twentieth century has seen the conflict between two main paradigms employed in researching educational problems. The one is modeled on the natural sciences with the emphasis on empirical quantifiable observations which lend themselves to analyses by means of mathematical tools. The task of research is to establish causal relationships, to explain (*Erklären*). The other paradigm is derived from the humanities with an emphasis on holistic and qualitative information and interpretive approaches (Verstehen).

The present thesis is mainly modeled according to the first paradigm because empirical quantifiable observations and statistical test were used. However, the purpose was not to explain causal relationships but merely describe the current situation and trends during the past 15 years.

2.1.3 Approaches of curriculum research

In the present subsection, the present thesis is considered from the viewpoint of curriculum research, known also as curriculum inquiry. Jenkins wrote (1991, p. 46):

> Simplistically, curriculum research is an umbrella term for the application of research techniques to problems of understanding posed by curriculum proposals, activities, or consequences. …. curriculum research is a practical rather than a theoretic art, typically concerned with defensible judgments rather than warrantable conclusions.

Clandinin and Connelly (1994, p. 1316) wrote "Consequently there is no agreed-upon structure of inquiry in curriculum studies." and presented their own set of six perspectives on schooling, which are the bases for six forms of curriculum inquiry. They wrote (p. 1317):

> The forms are: analytic, which seeks to analyze schooling in its component parts; portrait, which characterizes schooling as a working whole; intentional, which characterizes schooling in terms of its purposes and outcomes; structure–function, which characterizes schooling in terms of its structure and functions; societal, which characterizes schooling in microcosm or society; and narrative, which

> characterizes schooling in terms of the personal and social history of schooling and its participants.

From these six forms, the present thesis belongs under the heading "intentional." Clandinin and Connelly (1994, p. 1317) wrote about this form as follows (which fits the present thesis well):

> This reductive form of curriculum inquiry conceives of schooling in terms of student, teacher, and social accomplishments. Schooling is a process of specifying intentions and working out methods and structures for them to be realized. Researchers ask questions about people's wants in the form of needs assessments and opinion polls. Other questions are asked about the gap between the intentions and outcomes,…

2.1.4 Measurement, assessment, or evaluation?

The concepts "measurement" and "assessment" are close to the concept "evaluation." Next, the differences between these three concepts are clarified because these concepts are sometimes confused. The following text presents some quotations from Keeves (1994, pp. 362–364):

> Three concepts are widely used in this field, namely, "measurement," "assessment," and "evaluation." They have, however, different meanings, and it is necessary to draw as clear a distinction as possible between these here different concepts. The simple dictionary definition of measurement, that is, "assigning a numerical quantity to" is appropriate in most measurements in education and indicates the essential nature of the measurement process.
>
> As far as possible in field of education the term "assessment" is reserved for application to people. Moreover, with few exceptions, individual students are tested in the operation of assessment. Nevertheless, it is not uncommon in assessment for individual students to be administered tests, but for little or no importance to be attached to their individual results, and the individual data are aggregated to a group level, prior to analysis, interpretation, and reporting.
>
> In general, it would seem desirable to reserve the term "evaluation" in education to operations associated with nonperson entities, such as curricula, programs, interventions, methods of teaching, and organizational factors.

In the present thesis, needs assessment is an appropriate term because the assessment is targeted at the needs of Software Systems graduates; that is, it is applied to people. The term "evaluation" can be used in the case-specific

Part VII because it is targeted at the degree requirements that can be classified as a whole as a nonperson entity.

2.1.5 Levels of education

From different educational levels starting at preschool education, the present thesis is limited to higher education. According to the ERIC Thesaurus (Educational Resources Information Center, n.d.), the definition of higher education is: "All education beyond the secondary level leading to a formal degree." DeZure (2003, pp. 510–511) wrote about various trends in higher education: "What has changed are the goals of learning—from emphasis on knowledge of disciplinary facts and concepts (what students know) to broadly defined competencies (what students are able to do with what they know) …" From this view, the present thesis could be classified as old-fashioned because it concentrated more on contents than competencies. In particular, the purpose of the case-specific Part VII was to suggest changes to the degree requirements.

2.2 Computer science education as a research area

A description of computer science education as a research area was presented in the book Computer Science Education Research (Fincher & Petre, 2004, pp. 1–8). Fincher and Petre wrote (p. 8): "CS education is new. It co-exists in places with other sorts of publication (like SIGCSE), and where it starts and stops, where the edges of the endeavor are, is not yet entirely clear." They divided publications into four categories and assumed that practice-based "experience" papers were probably the most common type of paper today. These practice-based papers have evidence but are not strong on argumentation. It was their opinion that most CS education research papers should contain both evidence and argument when argument meant rationale, argumentation, or theory. (ibid., p. 2) They listed ten areas of CS education research (p. 3):

- student understanding
- animation/visualization/simulation systems
- teaching methods
- assessment
- educational technology
- the transfer of professional practice into a classroom
- the incorporation of new development and new technologies into the classroom
- transferring to remote learning ("e-learning")
- recruitment and retention of students
- the construction of the discipline itself.

From these ten areas, the present thesis belongs to two areas: assessment and the construction of the discipline itself. They wrote about the last area (p. 5):

> The final category is of a different kind, concerning questions about the construction of the discipline. In some other domains, for example mathematics, there is *didactics*, a sense of what it is we're supposed to teach, an acknowledgement of what we should cover as fundamental principles, and an associated understanding of which curricular areas are advanced and which are optional.

Apparently, the concept "didactics" has not been used commonly in English-speaking countries because it was not presented, for example, in the British Education Thesaurus (Marder, 1991) and was covered not at all or only briefly in the encyclopedias (e.g., Lewy, 1991) used during the literature search for the present thesis. According to Wulf (1991), the concept has been used in the German-speaking countries. Wulf wrote (p. 231):

> In the German-speaking countries from the seventeenth to the middle of the twentieth century, questions concerning aims, content, methods, and materials for education in general, and the classroom in particular, were mostly subsumed under the concept of didactics. This concept is derived from the Greek word *didaskein* and means the theory of teaching, instruction, or more specifically, education and content. …. In contrast to the concept "curriculum," "didactics" is mainly concerned with problems of curriculum theory, and does not include the aspects of thorough planning and evaluation through the use of systematically developed learning aids.

Didactics is relevant to the present thesis because it emphasizes contents. The concept can be further divided into general didactics and subject didactics, for example, the didactics of mathematics. The concept of subject didactics is relevant to the present thesis that is limited only to one field of science; that is, computer science.

However, one can also consider that needs assessment is one type of evaluation and thus a part of the area "assessment" presented by Fincher and Petre (2004). According to their description (ibid., p. 5), this area was mainly related to the assessment of students, for example, to automatic grading. However, they wrote (p. 5): "There are also studies that consider assessment in broader context, examining assessment from a curricular or cross-institutional perspective." This broader context is relevant to the present thesis. As explained in Section 2.1, the term "evaluation" is more suitable than the term "assessment" when the object of evaluation is a nonperson entity. Thus, a more

suitable name for this area would be "assessment and evaluation" instead of "assessment."

During the literature search for the present thesis, the tables of contents for the last 5–10 years of the following publications were browsed: Computer Science Education journals, the proceedings of the ITiCSE conference, the proceedings of the SIGCSE Symposium, and the SIGCSE Bulletin. From the SIGCSE Bulletin, only June and December issues were included because March and September issues were approximately the same as the proceedings of the SIGCSE Symposium and the ITiCSE conference. Based on this browsing, needs assessment was seen to be a rare or virtually non-existent area within CS education research. Only 32 (2%) of 1,583 papers were at least a little relevant to the present thesis. No needs assessments were found. The closest relevant papers were typically about program evaluation (e.g., Sanders & McCartney, 2003), program accreditation (e.g., Zweben, Reichgelt, & Yaverbaum, 2005), Computing Curricula 2001 (e.g., Roberts, Cover, Davies, Schneider, & Sloan, 2002), or some specialization (e.g., Fekete & Kummerfeld, 2002).

2.3 Some approaches to organize curricula

In the present section, some approaches to organizing an undergraduate program are considered. Some of these approaches are based on a particular theory of learning, such as constructivism, whereas others are more like models for organizing subject matter without foundation in any specific theory. The present thesis is not based on any particular theory of learning.

Some approaches are designed with a single course in mind, while others are designed for a whole degree program; however, most can be applied to both. A number of approaches are listed in order to show that there are several alternatives: decontextualized education (e.g., Ben-Ari, 2004, p. 87), discipline-based programs (e.g., Engel & Roberts, 2001, p. 49), domain-based curricula (e.g., Hirmanpour et al., 1995), integrated curricula (e.g., Glatthorn & Foshay, 1991), the module approach (e.g., Postlethwait, 1991), situated learning (e.g., Ben-Ari, 2004), and spiral curricula (e.g., Foshay, 1991). From these various approaches, the author selected the following three because they are the most relevant to the present thesis: decontextualized education, discipline-based programs, and domain-based curricula. These are presented and briefly discussed at the end of the present section.

In addition, some references are selected so that they can be related to the question of whether university education in computer science should emphasize scientific considerations or the needs of industry (Section 1.1.1). Some of these references consider university education in general, not computer

science education in particular. This question also concerns disciplines other than computer science. Mitter (1990, p. 408) wrote:

> Higher education in United States—and, to give another example, in Japan—usually starts with a broad general base comprising arts, sciences, and social sciences. It is gradually completed by courses qualifying the students for specialized academic careers. Traditional European universities, on the other hand, offer specialized studies from the beginning.

Gade (1991, p. 1087) wrote about the change in the USA:

> Not only has there been concern about how well teaching and learning is being done, but also about the content of what is taught and learned: the curriculum. In 1971 bachelor's degrees awarded were divided almost equally between arts and sciences and job-related subjects (business, education, etc.) By 1983 the arts and sciences share had dropped to about 36 percent of degrees awarded while job-related degrees constituted 64 percent of the total.

Enrollments have increased dramatically in many computer science programs during the past 20 years, including the case example. Mitter (1990, p. 409) continued about the consequences of mass education:

> The effect has not only been quantitative but also qualitative because the expansion has predominantly strengthened the profession-oriented task of higher education and thus weakened the concept of "purposeless" studies which had formerly determined the curricula in the arts faculties.

One related question for "purposeless" studies is whether the curriculum is decontextualized. One possible consequence is that, in a decontextualized curriculum, students might not specialize at all, or specialization might be postponed for as long as possible. Ben-Ari (2004, p. 87) wrote:

> In contrast, the assumption behind conventional schools is that the knowledge that is learned will prove applicable in future, even though the knowledge is not presented in specific context nor with the intention of training the students for a specific occupation. In elementary and high schools, teaching of subjects is not contextual, in the sense that all students learn the same mathematics, regardless of whether they intend to become research mathematics, stock brokers or supermarket cashiers.

> Furthermore, apprenticeship requires that the future occupation of a student be determined at a very early age, whereas decontextualized education enables this decision to be deferred until after high-school or even after college.

Discipline-based and domain-based programs in many ways stand opposed to decontextualized education. Engel and Roberts (2001, p. 49) wrote about discipline-based computer science programs as follows:

> In the United States and Canada, students at a university generally take a large fraction of their course work outside their area of specialization. In other countries, this generalist approach to university education is rare. Instead, students are expected to concentrate on a single field of study, possible augmented by a few courses in closely related disciplines. We refer to such curricula as discipline based. The discipline-based approach is typical for computer science curricula in England, for example, where such programs have a three-year duration.

According to Engel and Roberts (2001, p. 50), the number of computer science courses is 21 in a model for a discipline-based program and 15 in a model program at a research university in the USA. That is, the number of computer science courses in a discipline-based model is considerably greater even though it lasts one year less than the model for a research university in the USA. If these models truly represent reality, one possible consequence might be that specializations are more common in discipline-based programs. The case example is discipline-based and offers specializations.

In a domain-based computer science curriculum, an application domain such as finance or telecommunications is selected and the curriculum designed accordingly. Apparently such programs are rare, but at least one exists: Hirmanpour et al. (1995) presented the computer science program in the aeronautical domain. In the case example, the whole degree program is not domain-based, but four of its specializations are domain-based (telecommunications).

2.4 Papers in major computer science education publications

The most relevant papers published in major computer science education publications in the last 5–10 years are presented here. The author of the present thesis selected the following publications as being major publications in the area of CS education: Computer Science Education journal, the proceedings of the ITiCSE conference, the proceedings of the SIGCSE Symposium, and the

SIGCSE Bulletin. As mentioned in the previous subsection, 32 at least slightly relevant papers were found. From these 32 papers, only a few are presented here because most of the papers were of little relevance.

Most papers about specializations were targeted at specializations other than Software Systems but Fekete and Kummerfeld's (2002) paper about majoring in Software Development was very relevant to the present thesis. Their paper described a proposal apparently for the academic year 2002–2003. According to the description targeted at potential students (p. 73), "Typical job titles for graduates would be Programmer, Software Engineer, Software Developer, or Software Architect." These job titles are almost the same as those used to define the scope of the present thesis (Section 1.3.2) where only the title Software Architect was not used in the present thesis. The proposed courses of the major were three freshman programming courses emphasizing object-oriented programming and group work, lower division courses Analysis & Design, Code Construction & Testing, Concurrent Programming, and upper division courses Advanced Analysis & Design, Advanced Code Construction, and Testing. In addition, a student would be required to take a capstone project and choose one course from the following: User Interfaces, Database Applications, or Distributed Object Systems.

Sanders and McCartney (2003) conducted a survey of accredited CS programs in the USA and asked what methods were used for program evaluation. The proportions were as follows ($N = 47$): senior exit survey 89%, external advisory panel 68%, alumni survey 83%, employer survey 43%, written exams (external) 15%, written exams (internal) 9%, portfolios (department maintained) 6%, portfolios (student maintained) 2%, and oral exams 0%. In the present thesis, a senior exit survey was used.

The final report of Computing Curricula 2001 (Engel & Roberts, 2001) is detailed enough to be somewhat relevant to the present thesis. However, the papers in Computing Curricula 2001 (e.g., Roberts et al., 2002) were too general to be relevant. Similarly, the papers about accreditation were too general to be relevant because the papers were, for example, about the structure of new accreditation criteria (Zweben et al., 2005). Also the ABET/CAC accreditation criteria (Accreditation Board for Engineering and Technology, 2004) *per se* was not very relevant because only the minimum requirement for the extent of advanced studies was presented (ibid., p. 3) but no requirements were presented for the content of advanced studies.

Roberts' (2000) paper was interesting from the viewpoint of the recruiting process of top-level software developers. He wrote how productivity differences between ordinary and the best software developers affect recruiting (p. 85):

> Thus, companies whose business depends on software production will try to hire applicants from pools in which the likelihood of finding the

most talented individuals is high, such as graduates from top computer science departments, successful participants in collegiate programming contests sponsored by the ACM and similar organizations, or entrepreneurs who have developed successful freeware and shareware systems on their own. Competition to attract employees from these populations is intense.

2.5 Needs assessments in the field of information technology

Most previous needs assessments in the field of information technology have been carried out by professors and lecturers who work for information systems (IS) or information technology (IT) degree programs, not for computer science (CS) or software engineering (SE) programs. The results have been typically published in publications such as the MIS Quarterly, Journal of Computer Information Systems, and the proceedings of ACM's Special Interest Group for Computer Personnel Research (SIGCPR, currently merged with SIGMIS).

Nakayama and Sutcliffe's (2000; 2001) papers are good starting points for any reader wishing to get an overview. The first paper is an introduction to research on IT skill issues. They mentioned four major research areas: (a) classification/categorization of skills, (b) career orientation/path, (c) portfolio of skills required and/or desired, and (d) skill acquisition and transfer. Using these areas, the present thesis belongs to the area classification/categorization of skills.

Nakayama and Sutcliffe's (2001) second paper concentrated more on the area "portfolio of skills required and/or desired." Here, portfolio means a viewpoint of a company; for example, what skills a company should have in its IT skills portfolio. They reviewed 102 IT skills portfolio related papers that appeared in 1985–2000 and listed some basic information about these papers. From various categories, the papers listed in the categories "Skills Portfolio & Requirements" (pp. 107–108) and "Education" (p. 109) were the most relevant to the present thesis. They classified the papers as empirical, conceptual, or research-in-process. The empirical papers were the most relevant to the present thesis.

2.5.1 Classification of previous publications

Forty-five publications that were related to the present research were found. These publications were classified according to several criteria. First, the overall structure contained in the Computing Curricula 2001 report (Engel & Roberts, 2001, p. 2) was used: 91% of the publications come from the field of information systems (IS) or information technology (IT), and 2% from the field

of computer science or software engineering. No publication came from the field of computer engineering. Classification did not succeed in 7% of the publications.

Second, the publications were classified according to the research methods: content analyses 47%, surveys 16%, multi-method research other than triangulations 16%, triangulations 13%, interviews 2%, and 7% had no research method. All content analyses were job advertisement analyses and the papers without research methods were literature reviews or conceptual studies.

Third, the publications were classified according to target job positions. For example, the target job positions of Lethbridge's (2000, p. 45) survey were software developer and software manager because he asked "How useful have the details of this specific material been to you in your career as a software developer or software manager?" In most cases, it was not possible to classify a single job position but the research was targeted at all kinds of IS positions. The proportions were as follows (n = 35): all kinds of IS positions 69%, software developer 20%, systems analyst 14%, manager 9%, and end-user support specialist 3%. The sum of proportions was greater than 100% because some research had more than one target job position.

Fourth, the publications were classified according to the respondents. This classification was relevant only to surveys and interviews. In particular, surveys of IT professionals often had respondents of a different kind and only pooled results were presented. In these cases, the background information of the respondents was used for classification. For example, Lethbridge apparently did not ask job titles but his (1999, p. 75) respondents used 40% of their time in programming and, therefore, his survey was classified as a survey of software developers. Each group was classified as separate if a survey had several different respondent groups and their results were separated. The proportions of respondent groups were as follows (n = 19): managers and directors 37%, professors and lecturers 21%, software developers 21%, systems analysts 16%, students 11%, consultants 5%, recruiters 5%, and unspecified IT professionals 26%. The sum of the proportions was greater than 100% because some research had more than one respondent group.

Fifth, it was classified if statistical tests were used to analyze the results. This proportion was counted from any empirical research where statistical tests could be reasonable used. For example, Nakyama and Sutcliffe's (2000) paper was a literature review and use of statistical tests was not suitable. From 39 bodies of research, 31% reported the use of statistical tests. There was an interesting difference between the job advertisement analyses, and the interviews and surveys. Sixty-three percent of the interviews and surveys used statistical tests but in the job advertisement analyses the proportion was only 5%.

Sixth, the publications were classified according to the publication years: 1986–90 9%, 1991–95 33%, 1996–2000 40%, and 2000–2005 18%.

Next, previous publications will be presented divided into subsections according to the research methods used. The order of the subsections is such that the most relevant publications are presented first. First, the triangulations are presented. Second, the surveys targeted at software developers are presented because Lethbridge's (2000) survey was very relevant. After this, the order of subsections is no longer based on relevance but is according to the structure of the present thesis. Inside each subsection, the most relevant publications are presented first when possible. The publications are ordered according to the author names if the publications are equally relevant. At the end of Section 2.5, a summary is presented because the number of subsections is so large.

2.5.2 Triangulations

Typical triangulations have been surveys where the same questionnaire items were used for different respondent groups and the results of these groups were compared (Green, 1989; Kim, Shim, & Yoon, 1999; Knapp, 1993; Lee, Trauth, & Farwell, 1995; Mawhinney, Morrel, & Morris, 1994; Mawhinney, Morrell, Morris, & Helms, 1995; Mawhinney, Morrell, Morris, & Monroe, 1999; Nelson, 1991).

Mawhinney et al. (1994; 1995; 1999) conducted the most relevant previous triangulation where they used the same questionnaire items for three different groups that were similar to the present thesis. The biggest difference was that they used employers (apparently managers and directors) as respondents instead of software developers. Their 1994 paper presented the results of the employers' needs. In their 1995 paper students' opinions and the employers' needs were compared. The publication in 1999 was a research-in-progress paper without results when the topic was the comparison of the employers' needs versus the opinions of professors and lecturers. They (1995) found that the responses to more than 75% of the total of 162 questionnaire items were significantly different between the students and the employers. In all cases where the difference was significant, the mean of the students' answers was greater than the mean of the employers. That is, the students evaluated those items as being more important than the employers did. Across all 162 items, agreement between the students and the employers was high because the rank-ordered means for these two groups correlated strongly.

Kim et al. (1999) conducted a survey regarding the perceived importance of 30 IS issues. Their two respondent groups were IS professionals, and professors and lecturers. Based on their results, the two groups agreed on the relative importance of 18 items and perceived the importance of 12 items differently. Six out of 30 items were technical. Out of these six items, "Internet and electronic commerce" was the only item where there was a statistically significant difference between the answers of the two groups. The professors and lecturers evaluated the item as being more important than the IS

professionals did. In particular, their paper was relevant to the present thesis because it also included two items about distributed technology skills. Based on the results, both groups evaluated the items "Developing and maintaining distributed systems" and "Client/server computing" as being quite important. The differences between the means of the two groups were statistically not significant.

Lee et al. (1995, p. 313) wrote: "The lower-level IS jobs are rapidly disappearing, and the requirements for IS professionals are becoming more demanding in multiple dimensions, particularly in areas of business functional knowledge and interpersonal/management skills …. We argue further that the concept of a generic curriculum to meet the educational needs of all future IS professionals is obsolete, and different IS curricula must be tailored to meet the needs of different IS careers." This argument is interesting in relation to using specializations. One could interpret their argument to imply also that the need for using specializations was increasing as well. In addition, they wrote (p. 332): "The three stakeholder groups in this study, IS managers, IS consultants, and end-user managers, showed remarkable consistency in their vision of the skills and knowledge required by the successful IS professional of the future." It is interesting and relevant to the present thesis that the answers of different groups were consistent. However, the details of their research were only moderately relevant because the respondent groups were so different to those in the present thesis.

Knapp (1993) conducted two surveys in the Chicago metropolitan area. One survey was targeted at industry and the other at educational institutions. She asked what was taught in the institutions and what the need was for various technologies such as programming languages in industry. She found that the correspondence between the answers seemed to be good. This is an interesting and relevant finding on a general level but the details of her research were not relevant to the present thesis because it was targeted mainly at mainframe skills.

Two previous triangulations were not relevant to the present thesis because the respondent groups used were so different to those in the present thesis. Green's (1989) respondents were systems analysts and users. Nelson's (1991) respondents were IS personnel and end-users.

In addition, in some research more than one research method was used but the author of the present thesis did not classify these as triangulations because other research methods were used as a supporting phase before a survey (Bailey & Stefanik, 2001) or the results obtained by different methods were just presented but not compared (Sawyer, Eschenfelder, Diekema, & McClure, 1998; Young & Lee, 1997).

2.5.3 Surveys

Previous surveys are presented in the following order according to the respondents: software developers, other IT professionals such as managers working in industry, professors and lecturers, and finally students.

Software developers as respondents

From previous surveys to software developers, Lethbridge's (2000) survey was the most relevant. Lethbridge reported his research in more detail in his 1999 report (Lethbridge, 1999). He (ibid., p. 1) asked respondents about 75 educational topics: How much they had learned about the topic in their formal education, how much they knew about it at the time of answering, and how important the topic has been for their career? According to his results, the five most important topics for their career were data structures, specific programming languages, software design and patterns, requirement gathering and analysis, and software architecture (ibid., p. 32). Topics that were taught relatively more than their importance might warrant were physics, chemistry, and different areas of mathematics (ibid., p. 62). Some of Lethbridge's results are presented later in Section 21.1.3 when his results are compared with the results of the present thesis.

However, the methodology employed in Lethbridge's survey has been criticized. Kitchenham and Pfleeger (2002, p. 17) wrote "Thus, Lethbridge's target population was vague and his sampling-method non-existent. So although he described the demographic properties of his respondents (age, the highest education qualification, nationality etc.), no generalization of his results is possible." The author of the present thesis agrees with this criticism but it is his opinion that Lethbridge's research was not as poor as has been previously suggested. One excellent aspect of the research was that the respondents had on average 12.4 years of software development experience and spent most of their time on programming tasks (Lethbridge, 1999, pp. 75–76). One could interpret the demographics to mean that most of the respondents were at least intermediate experts, and might even be leading experts in software development. Thus, Lethbridge's research should not be classified as a survey but as a focus group study of experienced software developers. Regardless, one purpose of the present thesis was to confirm, complement, or disprove some of Lethbridge's results.

Bailey and Stefanik (2001) studied the knowledge, skills, and abilities considered necessary by different groups of IT personnel in the USA. In their paper, they concentrated on the findings that related to programmers. The focus groups identified 85 areas of knowledge, skills, and abilities that were important for programmers. From these 85 skills, 53 were classified as technical. According to the results of the survey ($N = 227$), the most important

technical skills were "Ability to read, understand and modify programs written by others" and "Ability to code programs." The least important technical skills were "Knowledge of RPG" and "Knowledge of Novell NetWare." However, many of their questionnaire items for technical skills were so different to those in the present thesis that the comparison of the results was difficult.

Beise, Padget, and Canoe (1991) organized a survey where 22 IS undergraduate programs around the USA took part. The survey asked, for example, about specific courses and their relevance to respondents' current work. As a consequence of several participating programs, the number of graduates who answered was large (N = 924). The most common respondent group was systems analysts (16%) but the author of the present thesis classified their research as a survey to software developers because the combined proportion of three software developer groups was greater than the proportion of systems analysts (Programming–3GL 13%, Programming–Systems 10%, and Programming–4GL 5%). They found that programming positions were common for recent graduates: approximately 68% of the respondents who graduated 1–3 years before answering the questionnaire worked as programmers when the proportion was approximately 30% for those who graduated at least seven years before the answering. They wrote (p. 20): "Respondents indicated that programming skills were important as preparation for all types of IS jobs by providing a mean response of 3.76 out of 5 to a question rating the importance of programming skills." Contrary to what Beise et al. expected, this was the opinion of the older respondents as well.

Haywood and Madden (2000) conducted a small-scale survey (N = 22) of IT professionals who graduated from the same institution 1–4 years before answering the questionnaire. Seventy-three percent of the respondents were "involved in software development" but job titles were not reported. Based on the results, programming skills were important because 14 respondents evaluated them as essential. Obviously, this was an expected result when 16 respondents were involved in software development. The questionnaire also contained items such as "database technology," "artificial intelligence," and "mathematics" that were interesting to the present thesis. Unfortunately, these results were not presented in the paper.

Other IT professionals as respondents

Thirteen surveys were found where respondents were managers, recruiters, systems analysts, or unspecified IT professionals. The most relevant was Mawhinney, Morrel, and Morris' (1994) survey of persons responsible for hiring personnel in information systems/data processing (N = 192). According to their results, the most common entry-level positions were in the area of Application Programming, job-related experience was highly valued, and certifications were rated at a low level of importance. Out of 31 items presented

in Table 1 of their paper, the author of the present thesis classified 13 items as being technical skills. From these 13 items, the items "Database" and "Computer Architecture/Hardware" were evaluated as being the most important whereas the items "Expert Systems/AI" and "Assembly Language" were evaluated as being the least important. In addition, they asked the importance of various general knowledge areas for a new employee. The most important item was "Ability to learn" and the least important "History, Art, Music." The importance of mathematics and natural sciences was queried as well, which is rare in surveys in the field of IS. The means were approximately as follows: Mathematics—Algebra 3.1, Mathematics—Calculus 2.2, and Natural Sciences 1.8 when the scale was 0 = Low, … , 5 = High.

The following two papers were somewhat relevant to the present thesis because at least one part of the paper was targeted at entry-level positions. These papers are listed according to the author names because their relevances were approximately equal:

- Watson, Young, Miranda, Robichaux, and Seerley (1990) conducted a survey where respondents were mainly directors and managers. The respondents ($N = 20$) ranked 20 skills that new management information systems (MIS) graduates should have if they seek an entry-level position as a programmer, systems analyst, or end-user support professional. For programmers, the five skills evaluated as being the most important were Application Programming Languages (COBOL, Pascal), Systems Analysis and Design (Life Cycle, Prototyping), Problem Solving, Data Base Concepts/Data Structures, and Operating Systems/JCL. Two skills evaluated as being least important were Expert Systems/Artificial Intelligence and Legal Aspects of Computing. Although their paper was published 15 years ago, it was somewhat relevant to the present thesis because one part of their survey was targeted at entry-level software developer positions.
- In Young and Lee's (1997) survey, 57% of the respondents were recruiters and 43% managers. According to the results (p. 49), the most common job titles filled by incoming IS employees were new application programmer and consultant. In addition, they asked the respondents ($N = 40$) to evaluate how necessary 30 skills were for new IS graduates. Some interpersonal skills such as Verbal Skills were evaluated as being the most important but several technical skills were evaluated as being important as well, for example, High-level Languages, Object-Oriented Languages, and Client/Server Tools. Apple/Mac Operating System and Low-Level Languages were evaluated as being not important.

The following papers were only moderately relevant to or useful for the present thesis. These papers are listed alphabetically by author:

- Kim et al. (1999) conducted a survey of unspecified IS professionals ($N = 140$) and reported results on 30 items that were partly technical, for

example, client/server computing but mainly soft skills such as project management. According to the answers, the most important technical skill was telecommunications and networking. However, their research was only moderately relevant to the present thesis because apparently it was not targeted at entry-level or software developer positions but the respondents had to evaluate the importance of given issues "in the field over the next three years" (p. 514). Probably the respondents interpreted that "the field" meant all kinds of IS positions at all experience levels.

- Knapp (1993) asked companies ($N = 45$) located in or near Chicago what programming languages, hardware, and application software packages they used. The most common programming languages and their proportions were Cobol 84%, Assembler 31%, C 20%, Fortran 13%, RPG II/III 7%, and Pascal 2%. In hardware, IBM was the most common in all other categories. The results concerning applications software packages such as word processors are not presented here because they were not relevant to the present thesis. The industry hiring practices were queried as well. She wrote (p. 24): "Their entry-level positions are typical. New employees start as Junior Programmers, Programmers, and Operations Personnel." Apparently, she did not ask the necessary skills for a certain position or positions but she asked what programming languages, hardware, and application software packages companies were using.
- Lee et al. (1995) used focus group meetings and a survey. The respondents of the survey were IS managers, user managers, and IS consultants. They were asked to evaluate the importance of various skills in 1994 and what they thought they would be in 1997; that is, three years in the future. The most important technical skill was "COBOL, or other third generation language" and the least important "Expert systems/AI" (p. 339). However, the respondents were asked to evaluate the importance of skills "… in supporting the computing needs of your company" (p. 338); that is, for the whole company, not for a particular position or experience level.
- Monin and Dewe (1994) surveyed IS professionals in New Zealand. Forty-nine percent of the respondents ($N = 443$) were managers or directors, and 27% consultants. The respondents rated the importance of various skills (a) at the time they first entered a position with their current job designation and (b) at present in their job. However, their research was moderately relevant to the present thesis because only approximately 15% of the respondents worked as software developers (p. 212).
- Nelson (1991) conducted a survey of IS professionals ($N = 150$). The questionnaire had 30 items on the usefulness of various topics. Seven items were technical. Out of these seven items, the item "Data access" was evaluated as being the most useful (p. 523). However, apparently he asked how useful the skills were for the respondent's job at the time of answering (p. 522), not how useful they would be for an entry-level employee or were

for the respondent's first position. The mean age of respondents was 36 years (p. 510). Thus, it was difficult to interpret how relevant his results were to entry-level positions.

- Sawyer et al. (1998) conducted a case study where several research methods were used. The respondents worked for the same company. In the survey, the respondents ($N = 140$) were asked to evaluate their current level of knowledge regarding the skill, current need for this skill, and their perception of their need for this skill three years in the future. Here, some results of questions about current needs are presented. The results of single items were not reported because the questionnaire included almost 600 items. According to the aggregated findings, the respondents evaluated business functional, technology management, and interpersonal management skills as being more necessary than information technology skills. The category "Information technology" included eleven subcategories such as "Distributed technology," "Operating systems," and "Database." From these eleven subcategories, "Infrastructure" was evaluated as being the most necessary. However, it was not possible to evaluate how relevant the findings were for entry-level software developer positions because the demographics of the respondents were not reported.

The following three papers were not relevant to or useful for the present thesis. Green's (1989) paper was not relevant because it concentrated mainly on soft skills: the only technical item was "programming." Orr and Hellens' (2000) publication was a research-in-progress paper without results. Winer's (1989) paper was not relevant because it concentrated on RPG and mainframes.

Professors and lecturers as respondents

Kim et al. (1999) conducted a survey of IS educators ($N = 51$) in the USA and reported results on 30 items that were partly technical such as client/server computing but mainly soft skills such as project management. According to the answers, the most important technical skill was telecommunications and networking. However, their research was only moderately relevant to the present thesis because apparently it was not targeted at entry-level or software developer positions but the respondents had to evaluate the importance of given issues "in the field over the next three years" (p. 514). Probably the respondents interpreted that "the field" meant all kinds of IS positions at all experience levels.

Mawhinney, Morrell, Morris, and Monroe (1999) conducted a survey of IS department chairs and area coordinators in the USA. However, their publication was a research-in-progress paper where no results of this survey were presented.

The following two papers were not relevant to the present thesis. Winer's (1989) research was not relevant because it concentrated on RPG and mainframes. Knap (1993) asked what topics were taught but apparently she did not ask the professors and lecturers what skills were necessary after graduation. Thus, this part of her survey was not actually a needs assessment but a survey of degree requirements.

Students as respondents

Only two previous papers were found where students were respondents. Mawhinney, Morrell, Morris, and Helms (1995, p. 233) asked students to answer "as they would expect the individuals in the industry to respond." In other words, the students were to guess at the industry responses. However, their paper was an extended abstract where no results from individual skills were reported.

Hingorani and Sankar (1995) asked students (N = 46) if they would prefer a position as systems analyst, programmer, end-user support person, or general manager after graduation. Thus, they asked about preferences but not about needs when needs are defined as in the present thesis (see later Section 3.1). Therefore, their survey was not a needs assessment.

2.5.4 Longitudinal research

From the 17 job advertisement analyses mentioned in Section 2.5, seven were longitudinal research (Athey & Plotnicki, 1998; Gallivan et al., 2004; Litecky, Prabhakar, & Arnett, 1996; Maier, Clark, & Remington, 1998; Prabhakar, Litecky, & Arnett, 1995; Todd et al., 1995; Trower, 1995). However, Litecky et al. and Prabhakar et al. contained results only from two or three subsequent years from the period 1993–1995. In the other longitudinal research the periods were 6–21 years. The periods used by Litecky et al. and Prabhakar et al. were so short that the author of the present thesis did not classify their research as longitudinal. The five other longitudinal researches were the most relevant to the present thesis. All these five researches were targeted at the IS/IT field. Next, these researches are presented.

The research of Todd et al. (1995) was the most relevant to the trend analysis of the present thesis because it had detailed results about software developers. They analyzed technical, business, and systems skills for programmers, systems analysts, and managers. The samples were collected from the years 1970, 1975, 1980, 1985, and 1990. They found that the requirements of programmers changed and have required greater versatility from 1970 to 1990. Technical skills were commonly required both in 1970 and in 1990 (92% and 96% of the advertisements, respectively) but in 1990 system and business skills were required considerably more often than in 1970. The

proportions of systems skills were 54% in 1970 and 68% in 1990. In 1970, the proportion of business skills was 28% and in 1990 60%. In addition, they found that the frequency of technical phrases per advertisement doubled from the mean 2.2 in 1970 to 4.3 in 1970. They wrote (p. 6): "In other words, while a 1970 job ad for a programmer indicated the need to know one operating system plus one programming language (typically COBOL), an ad in 1990 indicated that a programmer was expected to have skills in multiple operating systems and programming languages."

The following three bodies of researches were equally relevant to the present thesis. They were targeted at all kinds of IS positions.

- Athey and Plotnicki (1998) analyzed individual technical skills such as Cobol, Oracle, and Windows, and categories such as mainframes and minicomputers. They reported results from ten different cities as well. The samples were from the years 1989, 1992, 1993, and 1996. They found, for example, that demand for C/C++ and Visual Basic increased whereas the demand for Cobol and RPG decreased. Demand for all database skills used increased while Oracle was the most common skill in 1996. They wrote (p. 76): "Over 70% of all job opportunities require some knowledge of relational database technology" that apparently meant the situation in 1996. From platform skills, they reported (p. 76) that the proportion of mainframe and minicomputer skills decreased and microcomputer skills increased. The results of distributed technology skills were only from the years 1993 and 1996. The proportions of "Client Server" were 6.9% and 7.4%, respectively.
- The samples of Gallivan et al. (2004) were from the years 1988, 1995, 2001, 2002, and 2003. However, the results of 2002 and 2003 are not considered here because less comparable results were reported from these years. For example, the means of required skills were reported for the years 1988, 1995, and 2001 but not for the years 2002 and 2003. They analyzed more general issues such as changes in job titles as well but only technical skills are considered here. They found, for example, that (a) the mean of required technical skills per advertisement increased from 3.0 in 1988 to 4.2 in 2001 (p. 75), (b) the proportion of the category Operating Systems decreased from 26% in 1998 to 14% in 2001 (p. 76), (c) the proportion of the category Programming languages decreased from 43% in 1988 to 34% in 2001 (p. 76), and (d) the proportion of the category Networks/Communications increased from 20% in 1988 to 34% in 2001 (p. 76). In addition, they reported that the proportion of Compiler decreased from 22% in 1988 to only 1% in 2001 (p. 77). They did not explain what the item Compiler meant but anyhow, this is an interesting result because a course on compilers is commonly offered in CS programs. They did not report results on distributed technology skills.
- Maier et al. (1998) analyzed mostly individual technical skills such as C, Cobol, and Unix. The samples were from the years 1978/79, 1983/84,

1988/89, and 1993/1994. They too found an increasing trend in the number of required skills. They wrote (p. 38): "The typical advertisement mentioned an average of 2.6 skills in the late 1970s and 3.5 skills per advertisement in the mid-1990s." For example, they reported the following increased proportions in 1978/79 and 1993/94 (p. 41): C 0.2% and 29.3%, Oracle 0% and 11.9%, Unix 0.2% and 25.5%, and PowerBuilder 0% and 4.3%. In particular, the increase in PowerBuilder is interesting because it is a distributed technology skill. The proportion of Cobol decreased from 50.2% in 1978/79 to 23.3% in 1993/94 (p. 41).

- Trower (1995) analyzed individual technical skills and combined them as categories such as "3GL" that included Cobol, Fortran, or Pascal (p. 597). The samples come from the period 1990–1995 at one-year intervals. He (p. 599) reported that the number of advertisements in the Client/Server, Network, Object-Oriented, and Relational DB categories increased between the years 1990 and 1995. It was not possible to count the proportions because the sizes of the subsamples were not reported. However, he reported an index of all job advertisements as well and the numbers of these categories increased faster than the index did. Similarly, he reported that the following categories decreased relative to the index: CASE, Mainframe, 4GL, and 3GL.

Besides the previous scientific publications, there is one non-scientific report series that is worth mentioning. A British company Salary Services Ltd. has conducted job advertisement analyses for the past 15 years. The data were collected from several British newspapers and web recruiting services. In particular, this report series was interesting because it was problematic to get past data from web recruiting services (see later Section 12.1.1). A recent report (Salary Services, 2004a) was a combined longitudinal analysis and a detailed cross-sectional analysis of a single year as Sections 12 and 13 of the present thesis. The data were processed automatically, and the sample size was much greater than in the previously mentioned scientific longitudinal research. From the viewpoint of trend analysis, the interesting part was the ranking list of 150 individual technical skills from the years 1999–2003 with a one-year interval (pp. 224–229). Although this report was not a scientific publication, it is the author's opinion that it was convincing. As an example, in Table 2 are presented the ranks of the five most common programming languages in 2003. Note that the ranks presented in the table are not continuous because the results include skills other than programming languages as well. For example, in 2003 the two most common skills were SQL and Unix.

Table 2. Ranks of five most common programming languages in 2003. Source: Salary Services (2004a).

Year	C++	Java	C	Visual Basic	C#
1999	1	8	7	5	—
2000	1	4	13	5	150
2001	1	6	7	5	128
2002	2	8	5	7	67
2003	3	5	6	9	27

Note. Dash (—) indicates that the rank was not reported.

Cheney, Hale, and Kasper's (1990) research was a longitudinal study but they used interviews, not the content analysis of job advertisements as a research method. They interviewed senior IS managers in 1978, 1987, and 1988 using the same 20 questionnaire items. During the 1988 interviews, the respondents evaluated the importance of these items in the future (in 1995). The respondents evaluated the importance of these 20 items for three employee groups: project managers, systems analysts/designers, and programmers. Seven out of 20 items were technical. Here, only some findings concerning programmers and technical skills are presented. They did not report the aggregated importance of items by using means, for example, but only if the differences between the different years were statistically significant. Based on the results, the importance of the items File Design, Application Programming Languages, and Operating Systems was decreasing and the importance of the items DBMS and Telecommunications Concepts was increasing (p. 242).

2.5.5 Job advertisement analyses

Here, only the job advertisement analyses that have been published after the year 1999 are presented. Only the more recent publications are presented in this subsection because older results were relevant to the trend analysis but less relevant to the cross-sectional analysis of the present thesis. Two non-scientific reports and the slides of one conference presentation are presented first because they are more relevant than the scientific and professional publications:

- The quarterly reports from a British research company Salary Services Ltd. (2004a) have already been mentioned. The report of the last quarter of 2003 contains more than 250 pages. Here, only some results are presented as examples. The ten most common software skills in the last quarter of 2003 were SQL, Unix, C++, Oracle, Java, C, Office, Windows NT, Visual Basic, and SQL Server (p. 224). The most common skills for individual job positions were reported as well. For example, the five most common skills for software engineer in the last quarter of 2003 were C, C++, embedded, Java, and Unix (p. 253).

- The slides of Prabhakar, Arnett, and Litecky's conference presentation contained the results of individual technical skills such as Unix (C. Litecky, personal communication, December 8, 2004). The data source was apparently the web recruiting service Monster.com, $N = 4{,}070$, and the sampling period was September 2004. These results were relevant to the present thesis because they were recent and included some distributed technology skills as well. The ten most common IT skills and their proportions were (Slide 11): Web Programming 25.3%, Unix 16.6%, C++ and C# 15.7%, Java not "Script" 15.1%, Oracle Database 14.9%, SQL Programming 14.5%, .Net Development 12.4%, "Windows NT, 2002, XP, 2003" 12.2%, SAP/ERP 9.5%, and SQL Server 9.3%. The proportion of the item Client/Server or "C/S" was 4.4%. Apparently their coding scheme or way of reporting was a combination of individual skills and categories where several individual skills were combined. This was not reported on the slides but one could assume, for example, that the item Web Programming included more than one individual skill such as J2EE and JSP. However, it probably did not include .Net because it was presented as the separate item .Net Development. Nevertheless, the high proportions of the items Web Programming and .Net Development indicated that the need for distributed technology skills was relatively high in 2004.
- The content analysis of job advertisements in the ITAA report (Information Technology Association of America, 2002, pp. 45–53) was based on Dice's data, and thus, the data source was the same as in the cross-sectional analysis of the present thesis. Apparently, the sample included IT positions of all kinds, not just software developer positions. The sample size was approximately 30,000 (p. 45). The time of data collection was not reported but it appeared to be the first half of 2002. According to the results, the ten most common IT skills were C++, Oracle, SQL, Windows NT, Java, Windows 2000, Access, Routers, SAP, and XML (p. 45). Interestingly, C was not mentioned among the 20 most common skills (p. 45). This was probably a mistake because assembler's position was 20 and it is hard to believe that assembler would be more common than C in 2002.

The following papers are published in scientific or professional publications. Litecky and Arnett's (2001) paper contained results from both newspaper advertisements and from a web recruiting service. Data were collected during April 1999. The web service used was Monster.com when Dice was used in the cross-sectional analysis of the present thesis. Their paper included results about individual technical skills such as Cobol, Unix, and Oracle. Apparently, the sample included IT positions of all kinds ($N = 7{,}492$), not just software developer positions. According to their results, the five most common technical skills were Network—Win NT, Unix, Other relational, C++, and Windows '95 (p. 1923).

The samples of Gallivan et al. (2004) were from the years 1988, 1995, 2001, 2002, and 2003. Here, only some results of the years 2001–2003 are presented because they are more recent. The sample of 2003 was collected from the web recruiting service Monster.com whereas the samples of 2001 and 2002 were from Computerworld and the newspaper Atlanta Journal Constitution. For example, they reported that the proportions of the following programming languages or categories were in 2001: C 33%, OOP 26%, 4GL 22%, and Cobol 6% (p. 77). OOP meant object-oriented languages, the meaning of 4GL was not explained but it included SQL. In 2002, the proportions of Cobol and C were 12% and 15%, respectively (p. 79). In 2003, the proportions of Cobol and C were 2% and 6%, respectively (p. 79).

Adelman's (2000) paper was less relevant to the present thesis because its main topic was certification. He analyzed skills such as C++ as well but these results were not reported in the paper.

2.5.6 Summary

Here, only some more general findings and conclusions are summarized. For example, findings about individual skills such as C++ are not presented. The most relevant findings of the previous needs assessments in the field of information technology were:

- The mean of the number of required technical skills in job advertisements increased in 1970–2001. This was reported by three longitudinal analyses (Gallivan et al., 2004; Maier et al., 1998; Todd et al., 1995). Gallivan et al. wrote (p. 64): "… employers are seeking an ever-increasing number and variety of skill sets from the new hires."
- In some longitudinal analyses it was reported that need for mainframe skills decreased and need for database skills increased during the past 10–20 years (e.g., Athey & Plotnicki, 1998).
- In the most relevant triangulation, agreement between the students and the employers was high because the rank-ordered means for these two groups correlated strongly (Mawhinney et al., 1995).

2.6 Concept analysis of "software systems" and content analysis of degree requirements

During the literature search of the present thesis, no previous concept analyses of the concept "software systems" were found. Therefore, concept analyses of nearby concepts were searched as well. Only one relevant paper was found: McGuffee (2000) analyzed the concept "computer science." He did not give his own definition but mainly listed problems in previous definitions.

Similar content analyses of degree requirements were sought in the field of computer science and nearby fields. The results of this literature search were modest. Only one somewhat relevant publication was found (Reichgelt & Jovanovic, 2003). Their research was aimed at software management and they inspected the information technology programs of 22 institutions. Thus, the methodology and data sources were similar to the present thesis. However, their research was limited to the topic of a single course—or they made only suggestions about a single course (pp. 34–35)—whereas in the present thesis, the target area included several courses.

According to the draft of Computing Curricula 2005 (Shackelford et al., 2005, pp. 49–50), the Computing Ontology Project "is currently developing a framework for modeling all computing subject matter across the computing disciplines …. The Computing Ontology Project is well underway, and you can expect to hear of its progress via publications and conference presentations over at least next two years."

Next, the results of the literature search on content analysis of degree requirements are presented. As a research method, content analysis is near to or related to subject matter analysis. Textbook analysis is one type of subject matter analysis. For example, Tucker, Keleman, and Bruce (2001, p. 244) reviewed the common books of a data structures course in order to determine the extent to which mathematical topics were integrated into the CS curriculum. This analysis was limited to a single course as well.

During the literature search of similar content analyses, nearby fields were searched only slightly. One publication in the field of mathematics was found (Schmidt, Houang, & Cogan, 2002). They analyzed the requirements of mathematics in comprehensive schools (grades 1–8). Thus, their research was similar to that of the present thesis in the respect that the target area was wider than a single course. However, the scale in their research was much larger than in the present thesis. Their research was (p. 2) "…the most extensive and far-reaching cross-national comparative study ever attempted." For example, 42 countries participated in at least one part of the research.

McCauley and Manaris (2002) conducted a survey of ABET/CAC accredited undergraduate programs in CS and reported, for example, how often various upper-level courses were required. The survey did not concentrate on specializations in Software Systems but even so, their research was partly relevant to the present thesis. They (p. 2) reported that the three most common programming languages taught first in the academic year 2001–2002 and their proportions were Java 49%, C++ 40%, and C 11%. Five most commonly required upper-level courses in the academic year 2001–2002 and their proportions were Operating Systems 96%, Programming Languages 87%, "Ethical, Social Issues" 76%, Software Engineering 76%, and Architecture 69% (p. 4).

2.7 Normative studies

Computing Curricula 2001 (Engel & Roberts, 2001) is an impressive example of a normative approach to curriculum design. However, this is a recommendation of suitable contents rather than an educational research report. No definition for the concepts "normative research" or "normative study" was found during the literature search for the present thesis. According to Merriam-Webster's Collegiate Dictionary (Encyclopædia Britannica, n.d.), the word "normative" is an adjective and means (the first meaning) "of, relating to, or determining norms or standards." The following texts are quotations from Litecky et al. (2004, pp. 69–70):

> Herein, normative studies are characteristically aimed at producing authoritative pronouncements on IS career development and are generally issued under the aegis of academic and professional societies.
>
> Despite the strengths of the normative approach encompassed in (2, 1, 21), there seems to be a lack of focus on the depth and specificity of skills that are demanded in the market, and a weakness in terms of recognition of the importance of these skills in IS recruiting. This seems to arise because those efforts are primarily directed to long-range career development and not to the immediate characteristics that lead to the recruiting of IS personnel. An academic view of the required job skills is dominant in the normative reports; and on-job-training, non-college, and other sources of job skills are not a part of the models. In contrast, market oriented studies find and identify specific IT job skills, many of which are filled by employees who are outside the collegiate purview. The authors' observation of the normative approach contrasting relationship to IS recruiting, is an important motivator for examining the paradox in IS job skills research.

For the present thesis, the most interesting parts of Computing Curricula 2001 were core topics of the computer science body of knowledge (p. 17) and the three-year sample program for a discipline-based degree program outside the USA (pp. 48–50). The areas and core hours of the body of knowledge are presented in Table 3. The table is reordered first according to the core hours and then according to the area names.

Table 3. Areas and core hours of computer science body of knowledge. Source: Engel and Roberts (2001, p. 17).

Area	**Core hours**	**Area**	**Core hours**
Discrete Structures	43	Social and Professional Issues	16
Programming Fundamentals	38	Net-Centric Computing	15
Architecture and Organization	36	Information Management	10
Algorithms and Complexity	31	Intelligent Systems	10
Software Engineering	31	Human-Computer Interaction	8
Programming Languages	21	Graphics and Visual Computing	3
Operating Systems	18	Computational Science	0

The report presents the following three sample curricula (p. 46):

1. A research-oriented university in the USA
2. A university in which undergraduate education is focused on a single discipline, as is typically the case in countries outside North America
3. An institution, such as a liberal-arts college in the United States, with a small computer science department.

From these three sample curricula, the second option is the most relevant to the present thesis because the degree program of the institution is discipline based and does not require a large fraction of course work to be taken outside computer science. The discipline-based sample curriculum has a three-year duration whereas the model for a research university in the USA has a four-year duration. Two interesting differences between these two models are: (a) that the discipline-based model "does not include a specific course in science but instead assumes that this material can be integrated into the elective structure" (p. 49) and (b) the intermediate courses of the discipline-based model are based on the systems-based approach (p. 38) whereas the research university model is based on the traditional topic-based approach (pp. 36–37).

As part of the Computing Curricula 2001 project, an overview report should be published later. The draft of the overview report was published in April 2005 (Shackelford et al., 2005). On pages 35–36 of the draft are presented the common requirements for computing degrees. These requirements are relevant to the case-specific part of the present thesis because the area of the degree program of the institution is so broad that it could be classified rather as a Computing program than a Computer Science program. Actually, the English translation of the program name is not Computer Science but Computer Science and Engineering. The details of these requirements are not presented here because the report is a draft.

In addition, the following three curricula reports can be classified as the predecessors of Computing Curricula 2001: Curriculum '68, Curriculum '78, and Computing Curricula 1991 (Engel & Roberts, 2001, p. 6). These previous reports are only listed but not commented on because they are less relevant for

the present thesis. As part of the Computing Curricula 2001 project, four separate reports have been or will be published for the degree programs Computer Engineering, Computer Science, Information Systems, and Software Engineering. The report targeted at Computer Science programs is (Engel & Roberts, 2001). The other three reports are only listed in this subsection because they are less relevant to the present thesis.

2.8 Cognitive skills

The present subsection is related only or mainly to Section 10 where results on the cognitive skills of software developers are presented. What are cognitive skills? According to the ERIC Thesaurus (Educational Resources Information Center, n.d.), the term thinking skills should be used for the term cognitive skills. The description for the term thinking skills is the following:

> Interrelated, generally "higher-order" cognitive skills that enable human beings to comprehend experiences and information, apply knowledge, express complex concepts, make decisions, criticize and revise unsuitable constructs, and solve problems—used frequently for a cognitive approach to learning that views explicit "thinking skills" at the teachable level.

During the literature search, no research papers were found where the Delphi method was used in the field of psychology of programming. This is understandable because it is not common to use even questionnaires as a research instrument in this field. During the literature search, only seven papers were found where a questionnaire was used (e.g., Capretz, 2003). However, none of these papers is really related to the present research apart from the use of questionnaires. Because of the lack of similar research, some more general references are presented next. At the end of this subsection it is explained how these issues relate to the present thesis. Greeno and Simon (1988) wrote "Computer programming may be characterized 'as a whole' as a design task." Brooks (1983) wrote about design task domains:

> …, two fundamental activities in design task domains are composition and comprehension. Composition is the development of a design and comprehension results in an understanding of a design. The essence of the composition task in programming is to map a description of what the program is to accomplish, in the language of real-world problem domains, into a detailed list of instructions to the computer designating exactly how to accomplish those goals in the programming language

> domain. Comprehension of a program may be viewed as the reverse series of transformations from how to what.

Stanislaw, Hesketh, Kanavaros, Hesketh, and Robinson (1994) divided expertise in computer programming into two components: time-based expertise and multiskilling expertise. They wrote (p. 351): "Time-based expertise corresponds to the conventional notion of expertise, and is a function solely of the time spent on programming. Multiskilling expertise, by contrast, accrues through exposure to a variety of programming languages and tasks, and is related to the cognitive development of higher-level programming schemata." Détienne (2002, p. 35) wrote that one of the characteristics that distinguish "super experts" or "exceptional designers" from other experts is: "a broader rather than longer experience: the number of projects in which they have been involved, the number and variety of the programming languages they know."

In addition, Détienne (2002, p. 35) wrote that experts carry out some aspects of programming tasks completely automatically. She referred to Wiedenbeck (1985, p. 383) who found that experts were faster and made fewer mistakes than novices when both groups had to do a series of timed true/false decisions about short, textbook-type program segments. Perhaps, for example, the following skills are automated gradually when the programming experience increases: (a) using the basic commands of an editor such as Emacs and the programming system frequently used, and (b) knowing the details of the syntax and the code conventions of a certain programming language such as C.

The previous issues relate to the present research as follows: (a) Two activities, composition and comprehension, were used to interpret and divide the results of the present research. (b) The division time-based expertise vs. multiskilling expertise was used so that it was required that at least half of the respondents should be characterized as multiskilled experts. (c) The concept of skill automation was used with the questions about cognitive skills: the first question concerned higher-level skills and the second question concerned skills that might be partially or totally automated.

One aspect of cognitive skills is different design strategies. Détienne (2002, p. 26) wrote that experts have a broader range of more versatile strategies than novices. In addition, she wrote (p. 26): "Design strategies can be classified along several axes: top-down vs bottom-up, forward vs backward development, breadth-first vs depth-first, procedural vs declarative." See Appendix B for the explanation of these strategies. This division of strategies was used during the second questionnaire round in the Delphi study targeted at software developers. Originally, Visser and Hoc (1990, pp. 241–244) presented these design strategies and used somewhat different names. However, in the present research the names presented by Détienne (2002) were used.

2.9 Necessary skills in the future

This literature review about forecasting the necessary skills in the future was limited to needs assessments where the future needs were evaluated as well. These publications were found when needs assessments were sought (presented previously in Section 2.5). In other words, no separate literature search that would concentrate on the future needs was performed because forecasting the future was only a small part of the present thesis.

Sawyer et al. (1998) conducted a survey where the respondents evaluated their current need for this skill and their perception of their need for this skill in 2001; that is, three years in the future. According to the aggregated findings, the respondents evaluated business functional, technology management, and interpersonal management skills as being more necessary than information technology skills. The category "Information technology" included eleven subcategories such as Distributed technology, Operating systems, and Database. From these eleven subcategories, Desktop Support was evaluated as being the most necessary. Sawyer et al. did not use statistical tests to analyze the differences but generally, the differences between the current and the future levels were very small. The overall mean of all technical skills was the same (2.3) for the current and the future level. However, it was not possible to evaluate how relevant the findings were for entry-level software developer positions because the demographics of the respondents were not reported.

Cheney et al. (1990) interviewed senior IS managers in 1978, 1987, and 1988 using the same 20 questionnaire items. During the 1988 interviews, the respondents evaluated the importance of these items in 1995, which was seven years in the future. The respondents evaluated the importance of 20 items for three employee groups: project managers, systems analysts/designers, and programmers. Seven out of 20 questionnaire items were technical. Here, only some statistically significant ($p < .01$) findings concerning programmers and technical skills are presented. They did not report the aggregated importance of items by using means, for example, but only if the differences between the different years were statistically significant. Based on the results, the item Operating Systems was evaluated as being less important in the future and the items DBMS and Telecommunications Concepts were evaluated as being more important in the future (p. 242).

Thus, Cheney et al. found statistically significant differences between the current needs and forecasted needs in technical skills when apparently Sawyer et al. did not. A possible explanation is that Cheney et al. asked their respondents to forecast seven years ahead and specified the target positions whereas Sawyer et al. asked only about three years ahead and asked about the need for skills in the respondents' own work. It is not surprising that the Sawyer et al. respondents evaluated their needs in the category Business Functional Skills to increase in the future but their needs in the category Information

Technology Skills to stay at the same level if the respondents were mostly managers and directors.

3 Research methods

The research methods used are presented in this section. The order of presentation mostly follows the structure of the present thesis. For brevity, most descriptions are brief because the number of methods used is so large. The descriptions of needs assessment and triangulation are presented first and are more comprehensive because they are the main research methods or superordinate concepts for the methods used in the present thesis.

Details on applying these methods in the present thesis are presented later in sections entitled "Research method" in Parts II–V (e.g., Section 4.1).

3.1 Needs assessment

Needs assessment was selected as the main research approach for the present thesis because it was suitable for solving the following main research problem: "What technical skills do graduates from specialization in Software Systems need in their work after graduation?" Next, needs assessment is explained as a research approach. The following text comprises quotations from Suarez (1994, pp. 4056–4057):

> Needs assessment is an information-gathering and analysis process which results in the identification of the needs of individuals, groups, institutions, communities, or societies. In education, the process of needs assessment has been used, for example, to identify the needs of students for instruction in a given subject area; to determine weaknesses in students' overall academic achievement; to determine the needs of teachers for educational training; and to determine the future needs of local, regional, and national educational systems. It is the intent of needs assessments to identify areas in which deficits exist, desired performance has not been attained, or problems may be expected in the future. The results of needs assessments are then used for further action such as planning or remediation to improve the situation.
>
> Educational needs have been assessed and analyzed for centuries. However, formalized assessments of educational needs were not conducted on a widespread basis until the middle of the twentieth century (Suarez 1981).
>
> The determination of needs is such a broad concept, applicable in so many situations, that a common conceptual model and set of needs assessment procedures have not emerged. Major variations in needs assessments appear in: (a) the definition used for the term "need," (b) the purposes for which needs assessments are conducted, (c) the

> standards by which needs are identified, and (d) the strategies and procedures used in the process.
>
> The majority of needs assessment studies, however, has been based on a variation of one of three definitions of the term "need." The most widely used definition of "need" for need assessments is that of a discrepancy. This definition, introduced by Kaufman (1972), suggest that needs are areas in which actual status is less than targeted status. Another commonly used definition of "need" is that of a want or preference. A more stringent and less used concept of need for needs assessment studies is that of deficit. A need is said to exist if the absence or a deficiency in the area of interest is harmful.

In the present thesis, the purpose of the needs assessment was to provide information for the planning of the degree requirements and Kaufman's definition of need was used. Such assessment of, for example, a course, a specialization, or a degree program could be carried out using at least three different approaches:

- It could be assessed what knowledge or skills were needed before a student started to study the topic. The results obtained could be called prerequisites or entrance requirements.
- A student's needs could be assessed when he or she started to study a topic. The motivation and general expectations for studying the topic could be considered. However, this could be difficult. Often a student could define his or her needs only on a very general level.
- It could be assessed what knowledge or skills were needed later, for example, after graduation. In the present thesis, this approach was used.

3.2 Triangulation

The following text comprises quotations from Denzin (1994, pp. 6461–6463):

> Triangulation is the application and combination of several research methodologies in the study of the same phenomenon. The diverse methods and measures that are combined should relate in some specified way to the theoretical constructs under examination. Triangulation can be employed in both quantitative and qualitative studies. Traditionally, in quantitative studies triangulation has been used as a method of validation. That is, the researcher uses multiple methods to validate observations.
>
> While it is commonly assumed that triangulation is the use of multiple methods in the same study of the same phenomenon, this is only one form of the strategy. There are four basic types of

triangulation: (a) data triangulation, involving time, space, and persons; (b) investigator triangulation, which consists of the use of multiple, rather than single observers; (c) theory triangulation, which consists of using more than one theoretical scheme in the interpretation of the phenomenon; (d) methodological triangulation, which involves using more than one method and may consist of within-method or between-method strategies. There is also multiple triangulation, when the researcher combines in one investigation multiple observers, theoretical perspectives, sources of data, and methodologies (Denzin 1989).

The present thesis is a multiple triangulation when data triangulation and methodological triangulation were used. Triangulation was selected as the research method for the present thesis because on the one hand it was assumed that different data sources and research methods would complement each other and on the other hand it was used as a method of validation.

The different data sources and research methods of the present thesis are presented in Figure 2. Nine different data sources and three different research methods were used. In the figure, the data sources are placed inside the boxes. The names of the research methods are placed above or left of the dashed lines. The data sources of the case study are located on the right side of the figure. It can be noticed from the figure that content analysis was the most common research method. The concept analysis is not presented in the figure because it can be classified as being a part of the Delphi study targeted at the professors and lecturers.

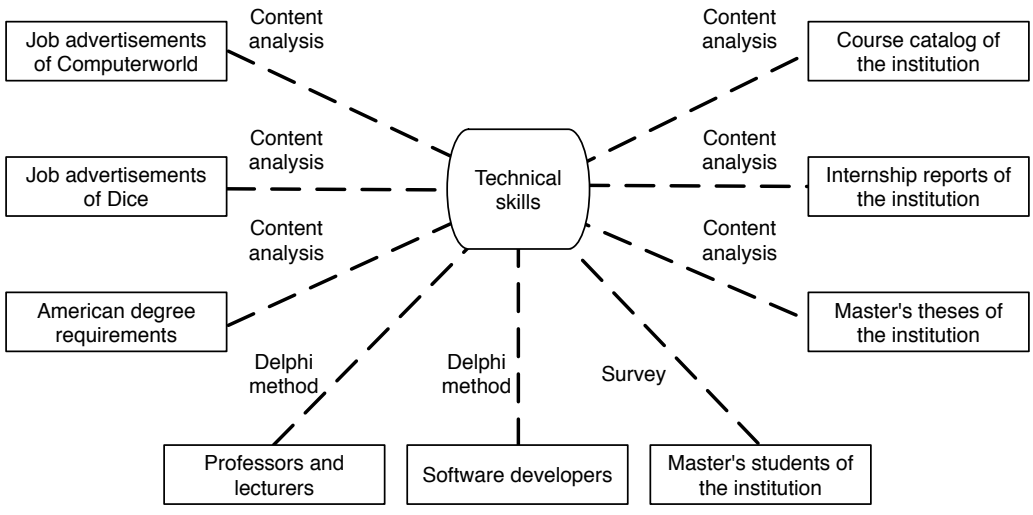

Figure 2. Data sources and research methods of present thesis.

Possible respondent groups would be at least the following groups:

- professors and lecturers working at CS departments

- managers and directors of software developers
- experienced software developers
- alumni; that is, older graduates of the institution
- current senior students or recent graduates of the institution
- recruiters of employers, in particular, the recruiters of large companies.

From these groups, professors and lecturers, experienced software developers, and current senior students or recent graduates of the institution were selected as the respondent groups of the present thesis. The experienced software developers were an obvious choice for research of this kind. The professors and lecturers were selected because it was assumed that they have well-founded opinions on the importance of the topics that they teach and on nearby topics. One could consider that the students were not a suitable respondent group because they did not have enough working experience. However, the respondents of the present thesis were Master's students who could be classified as Bachelors having 1–3 years of full-time working experience on average (Sections 9.2.1 and 9.3). Thus, it was assumed that these students would have enough relevant working experience in entry-level positions, in particular.

Recruiters, managers, and directors were not used but instead job advertisements were analyzed. The alumni of the institution were not used because it was assumed that the group would be similar as experienced software developers.

The selection of data sources for the job advertisement analyses is explained in Sections 1.3.4, 12.1.1 and 13.1.

The data sources of the case study presented in Part V; that is, the course catalog, internship reports, and Master's theses of the institution, were selected from the results obtained earlier as part of the Licentiate's thesis project of the author (see Appendix A: Surakka, 2001). From this material, only the most suitable parts were selected for this Doctoral thesis. For example, the results of Finnish newspaper advertisements were excluded because it was considered that the results of American job advertisements were sufficient (Part IV).

3.3 Concept analysis

Concept analysis was selected as a method of the present thesis because it was suitable for solving the following subproblem: What does the concept "software systems" mean? Tennyson (1994, p. 1022) wrote about concept analysis:

> There are two basic types of analysis: (a) a content (task) analysis that focuses on defining the critical characteristics of the concepts and the

relationships of those characteristics according to superordinate and subordinate organizations; and (b) a contextual analysis that focuses on the memory or knowledge base organization of the concepts. The first analytic method identifies the external structure of the concepts (either a taxonomy or hierarchy) but does so independently of how it might actually be stored in human memory.

In the present thesis, an analysis of the first type was conducted. Perhaps the best known method of concept analysis is the Wilson method (e.g., Avant, 2000). The method contains the following eleven steps (ibid., p. 55). However, in the present thesis, only four of these eleven steps were applied (1, 3, 4, and 6).

1. Isolating questions of concept
2. Finding right answers
3. Model cases
4. Contrary cases
5. Related cases
6. Borderline cases
7. Invented cases
8. Social context
9. Underlying anxiety
10. Practical results
11. Results in language

3.4 Delphi method

An overview of the Delphi method can be found, for example, from Wilhelm (2001). The method was originally used to forecast the future; the name originates from "the oracles of Delphi" where Delphi refers to an ancient Greek island. However, in the present thesis forecasting the future was only a small part and the method was selected for other reasons. These reasons are explained later at the end of this subsection.

Some basic properties of the method are the following. First, there are several questionnaire rounds. Second, the results from the previous round are used as material for the next round. Thus the respondents may change or tune their previous answers. The method allows group communication without gathering all respondents in the same place at the same time, which in this case would have been difficult to achieve. Moreover, in this way the respondents had more time to consider their answers and make their views more explicit.

Delphi is a qualitative research method, where the quality rather than the number of respondents is the more important factor. The statistical reliability of the results is therefore not the general goal, and thus the number of respondents need not be very large.

The main reasons for using the Delphi method were: (a) the method allowed group communication without gathering all respondents in the same place at the same time, and (b) a second questionnaire round was assumed to be

beneficial because some topics were rather vague. When the Delphi method was used, it was possible to ask refined questions on these topics during the second questionnaire round. In particular, this was the case for the concept analysis (Section 4) and the cognitive skills of software developers (Section 10).

3.5 Content analysis

Anderson (1994, p. 1074) wrote: "Content analysis, sometimes referred as document analysis, includes the methods and techniques researchers use to examine, analyze and make inferences about human communications. Typically, communications consist of printed or written text but may also comprise photographs, cartoons, illustrations, broadcasts, and verbal interactions." The basic goal of content analysis "is to take a verbal, non-quantitative document and transform it into quantitative data (Bailey, 1978)" (Cohen et al., 2000, p. 164). Some good properties of content analysis are: (a) it is a non-disturbing method because data occur regardless of whether the research is carried out or not, and (b) it is often possible to get a representative sample.

In the present thesis, content analysis was used because it is a self-evident choice for analyzing documents such as job advertisements. Other methods, if there are any, were not even considered as an alternative. As Anderson wrote, content analysis is "sometimes referred as document analysis." Content analysis is related to the main problem and to several subproblems of the present thesis because the method was used to analyze different documents for different purposes.

According to Anderson (1994, p. 1076), typical procedures for most content analyses are the following: 1. identify the corpus or universe to be studied, 2. define the categories into which the universe is to be partitioned, and 3. determine the units for analysis.

In the present thesis, several corpora, categories, and units for analysis were used because content analysis was used to analyze different documents for different purposes. In most content analyses of the present thesis, the categories were decided before the coding, which is apparently a typical order. However, the order was different in the job advertisement analyses where all data on technical skills were first coded without the use of categories and the broader skill categories were determined later.

3.6 Survey

The following text comprises quotations from Rosier (1994, p. 5854):

> Survey research methods in education describe procedures for the collection of information associated with education. Surveys are conducted to accomplish two main purposes. First, descriptive or enumerative surveys are used to obtain descriptive information. Second, analytic, exploratory or comparative surveys are designed to examine relationships between measures and variables in the survey.
>
> There are, in general, two types of surveys. First, there is the complete census of all members of the target population. Second, there is a sample survey in which a known proportion of the target population is under survey. The collection of data from a complete population is generally both costly and a burden on the people involved.
>
> Analytic surveys typically collect information on a range of predictor or explanatory variables that are hypothesized to influence the criterion variables of interest.

In the present thesis, the Master's students were surveyed. The survey was used instead of, for example, interviews in order to be able to compare the results on the importance of various subjects and skills with the results of the Delphi studies targeted at software developers and professors and lecturers. The Delphi method was not used for the students for practical reasons that will be explained later in Section 9.1.

The goal was to collect data from all members of the target population. Sampling was not used because the target population was so small. The research was a descriptive survey.

3.7 Trend analysis

Trend analysis is one type of longitudinal research. Cohen et al. (2000, p. 174) wrote: "A clear distinction is drawn between longitudinal and cross-sectional studies. The longitudinal study gathers data over an extended period of time; a short-term investigation may take several weeks or months; a long-term study can extend over many years." They continued (p. 174): "Where a few selected factors are studied continuously over time, the term 'trend study' is employed." and (p. 175) "Essentially, the trend study examines recorded data to establish patterns of change that have already occurred in order to predict what will be likely to occur in the future."

In the present thesis, trend analysis was used when job advertisements from the year 1990 to 2004 were analyzed for content (Section 12). Trend

analysis was an evident choice and no other methods were considered. The purpose was to solve the following two subproblems of the thesis:

- Has the number of required technical skills increased during the past 15 years in job advertisements targeted at software developers? Todd et al. (1995) reported that the number of technical phrases in job advertisements for programmer positions increased from the mean of 2.2 in 1970 to 4.2 in 1990. Has this increase continued after the year 1990?
- In particular, how has the number of required distributed technology skills increased? World Wide Web technology was released in 1993. After this, the number of web sites has increased rapidly. As a consequence, skills related to distributed systems should now be required more often than ten years ago.

3.8 Single cross-sectional study

Lietz and Keeves (1994, p. 1216) wrote about single cross-sectional studies as follows:

> These research studies consider one target population at one point of time only. The limitation to one target population and one time point is primarily determined by considerations of personnel and cost, which require that such studies can be completed relative quickly at modest cost.

In the present thesis, the concept of a "single cross-sectional study" was used for naming purposes; that is, to differentiate the job advertisement analysis where only the job advertisements of the year 2004 were analyzed (Section 13) from the trend analysis of job advertisements (Section 12). The purpose of the single cross-sectional analysis was to solve the main problem and the following two subproblems of the present thesis:

- What are differences, if any, between the required skills of programmers, software engineers, and software developers?
- What are the differences between entry-level and senior-level software developer positions?

3.9 Case study

Sturman (1994, p. 640) wrote:

> "Case study" is a generic term for the investigation of an individual, group, or phenomenon. While the techniques used in the investigation may be varied, and may include both qualitative and quantitative

> approaches, the distinguishing feature of case study is the belief that human systems develop a characteristic wholeness or integrity and are not simply a loose collection of traits. As a consequence of this belief, case study researches hold that to understand a case, to explain why things happen as they do, and to generalize or predict from a single example requires an in-depth investigation of the interdependencies of parts and of patterns that emerge.

Parts V and VII of the present thesis can be classified as case studies. In Part V, the case example is the Degree Program of Computer Science and Engineering at the Helsinki University of Technology. Part V is only moderately related to the main problem and subproblems of the present thesis, which is explained more at the beginning of Part V.

In Part VII, the case example is the specialization in Software Systems of the same institution. The purpose of Part VII was to solve the following subproblem of the present thesis: "How are the needs found as the answer to the main problem different from the planned degree requirements of the institution in the academic year 2005–2006?" As was mentioned previously, this subproblem can be classified as a planning problem as well, rather than as a research problem.

3.10 Statistical analysis

Different statistical tests were used in the different parts of the present thesis because the sample sizes varied, for example.

It is assumed that a typical reader of the present thesis is not familiar with nonparametric statistical tests but knows the Student's *t* test (e.g., Milton & Arnold, 2003, pp. 347–349) that is often used to compare means. In the questionnaires of the present thesis (Sections 7–0), the Student's *t* test was not suitable because the scale was ordinal and the sample sizes were so small. There were no statistical tests available for comparing means in the situation of this kind and the Mann-Whitney test (Conover, 1999, pp. 271–275) was the most suitable option. Note that the Mann-Whitney test compares the ranks; that is, the order of items as a whole, not the means.

Nonparametric statistics were used in Sections 11 and 21.1.3 as well where the Spearman rank correlation coefficient r_s (ibid., p. 314) was calculated in order to compare the results of the questionnaires. In addition, it was calculated if the correlation was statistically significant. The procedure was explained in Conover (ibid., pp. 316–318).

In the job advertisement analyses (Sections 12 and 13), the use of nonparametric statistics was not necessary. The Student's *t* test and the Smith-Satterthwaite procedure (e.g., Milton & Arnold, 2003, pp. 347–349) were used

to test if the difference between two means was statistically significant. The *z*-test for proportions (e.g., ibid., p. 324) was used to test if the difference between two proportions was statistically significant. The two-sided confidence intervals for proportions were calculated using an equation from Milton and Arnold (ibid., p. 315). Apparently, this equation did not have a name.

For significance, the following limits were used: not significant $p \geq .05$, almost significant $p < .05$, significant $p < .01$, and very significant $p < .001$. Typically, the confidence level $1 - \alpha = .99$ was used to avoid Type I errors because the number of questionnaire items, for example, was so large.

The statistical tests were calculated with Microsoft Excel or a statistical analysis program NCSS.

3.11 Evaluating validity and reliability

In Parts II–V, validity and reliability of the results are discussed at the end of each section where a new research area is presented (e.g., Section 4.3). Typically, these sections are titled "Evaluation." According to Cohen et al. (2000), there are several types of validity and reliability. For example, the following validity types were listed (ibid., p. 105): content validity, criterion-related validity, construct validity, internal validity, and external validity. Only content validity and external validity are discussed, because the author considered them as the most relevant to the present thesis. They wrote about these validity types as follows (ibid., p. 109):

- "To demonstrate this form of validity [content validity] the instrument must show that it fairly and comprehensive covers the domain or items that it purports to cover."
- "External validity refers to the degree to which the results can be generalized to the wider population, cases or situation."

They continued (ibid., p. 117):

> Reliability is essentially a synonym for consistency and replicability over time, over instruments and over groups of respondents. It is concerned with precision and accuracy; some feature, e.g. height, can be measured precisely, whilst others, e.g. musical ability, cannot. For research to be reliable it must demonstrate that if it were to be carried out on a similar group of respondents in a similar context (however defined), then similar results would be found. There are three principal types of reliability: stability, equivalence and internal consistency.

Only stability over time is considered, because the author considered it most relevant to most of the research work involved in the present thesis. However,

in the trend analyses of job advertisements presented in Section 12 and of the Master's theses presented in Section 16.2, stability over time is not even expected.

Part II: Specialization in Software Systems defined by courses

In Part II, the purpose is to characterize the area of specialization in Software Systems using typical course names. Two different methods were used: the concept analysis of the concept "software systems" and the content analysis of degree requirements. In addition, the results obtained by these two methods were compared. This part is not a needs assessment and thus, it is different to the other parts of the thesis.

The purpose of the concept analysis is to solve the following subproblem of the present thesis: "What does the concept 'software systems' mean?" The content analysis tries to answer the main problem "What technical skills do graduates from specialization in Software Systems need in their work after graduation?" because the degree requirements should be or might be such that they cover the most common requirements in the job market.

The scope of this part of the thesis is presented in Figure 3 using the data sources and research methods. The figure is the same as Figure 2 presented previously in Section 3.2 but the boxes related to this part are filled in gray.

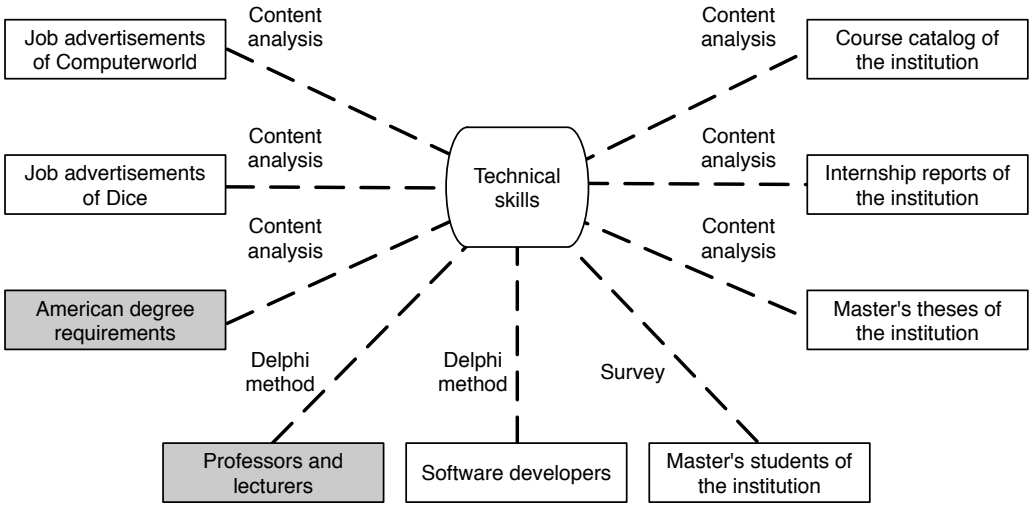

Figure 3. Scope of Part II of thesis.

4 Concept analysis of “software systems”

The purpose of the present research was to solve the following subproblem of the present thesis: What does the concept “software systems” mean? In addition, the Finnish concepts “ohjelmistotekniikka,” “ohjelmistotuotanto,” and “ohjelmistojärjestelmät” were analyzed. The word “ohjelmistotekniikka” has no good English translation. It can be translated as software techniques, software technology, or software engineering. The word “ohjelmistotuotanto” is typically translated into English as software engineering and the word “ohjelmistojärjestelmät” as software systems.

These four concepts were selected because (a) the concepts “software systems” and “ohjelmistojärjestelmät” were central to the thesis and (b) it was assumed that the two other concepts were nearby or superordinate concepts for these two central concepts.

As was explained in Section 1.5, the work related to the present research and to the other research of Parts II–V is presented in Section 2, not distributed through Parts II–V and most section evaluations in Parts II–V (e.g., Section 4.3) are quite brief and cover only validity and reliability. The results are compared with previous findings and possible differences are considered later in the general discussion (Sections 21.1 and 21.2). This structure is used in order to move easily from one section to another in Parts II–V.

4.1 Research method

In the present research, the Delphi method was applied. Two questionnaire rounds were performed between January and April 2005. The Delphi method was chosen for the present research because the analysis could be more detailed on the second questionnaire round. That is, during the first round the topic was the broader concept “ohjelmistotekniikka” and the other concepts were asked on the second round.

In addition, the WWW pages, the study guides, or the course catalogs of 14 Finnish universities were analyzed before the Delphi study was started. The purpose of this analysis was to determine how common the Finnish concepts “ohjelmistotekniikka,” “ohjelmistotuotanto,” and “ohjelmistojärjestelmät” were and how these concepts were defined. This analysis can be classified as being part of the planning of the first questionnaire as well.

4.1.1 Finding respondents

Nineteen respondents were selected when the goal was to find from 10 to 20 Finnish university professors and lecturers from the area of software systems and software engineering. In Delphi studies, recommendations are often used to select the respondents. However, recommendations were not used in the present research because it was assumed that any university professor or lecturer of certain teaching areas would be suitable enough.

Finland has 16 universities that offer a computer science or similar degree. From these 16 universities, the Helsinki University of Technology, the Tampere University of Technology, and the University of Helsinki were chosen because the intakes of new computer science students were the biggest in these three universities. The intakes were 135, 170, and 267 students in 2004, respectively.

From these three universities, the respondents were selected mainly according to the courses they taught. The courses used were: Analysis and Design of Algorithms, Compilers, Data Structures and Algorithms, Databases, Distributed Systems, Embedded Systems, Operating Systems, Programming Languages, Software Architectures, Software Engineering, and Software Project. Position, working experience, and publications of the candidates were not used as criteria. The only minimum criterion was a Master's degree. Twenty-five professors and lecturers were asked and 19 of them approved to take part.

4.1.2 Questionnaires

Two questionnaire rounds were conducted. The questionnaires are available on the web page of the institution (Surakka, 2005b). The topics of the first questionnaire were (a) forecasting the future in the area of software engineering and software systems, (b) the Finnish concept "ohjelmistotekniikka," (c) evaluating the respondent's own skills and knowledge on various skills and subjects, and (d) the importance of various subjects and skills for graduates. The questions were presented in Finnish and the respondents answered in Finnish.

During the second round, the respondents were divided into two groups that answered different questionnaires. Eleven (58%) respondents were classified as professors and lecturers in the area of software systems and 8 (42%) respondents as professors and lecturers in the area of software engineering. The software systems respondents were asked about the concept "software systems" and the software engineering respondents about the concept "software engineering." The questions were presented in English and the respondents had to answer in English. Ten questions were asked about the concept "software system."

4.2 Results

First, the results of the analysis of the WWW pages, the study guides, and the course catalogs of Finnish institutions are presented. Second, the background information of the respondents is presented. Third, the results of the concept "software systems" are presented.

4.2.1 Analysis of WWW pages, study guides, and course catalogs of Finnish institutions

Finland has 20 institutions that have university status. Sixteen of them offer computing programs. Two of these 16 institutions were not selected for the present analysis because the purpose was to analyze Finnish concepts whereas these two institutions used Swedish as the main language. Thus, the WWW pages, the study guides, or the course catalogs of the computing degree programs of 14 Finnish institutions were analyzed. This analysis was conducted in the spring of 2005 and the requirements were from the academic year 2004–2005.

The word "ohjelmistotekniikka" was used at nine (64%) and "ohjelmistotuotanto" at eleven (79%) institutions as a name of a course, specialization, option, track, or degree program. The Helsinki University of Technology was the only one (7%) that used the word "ohjelmistojärjestelmät." Typically "ohjelmistotekniikka" was used as a name of a broader entity such as specialization and "ohjelmistotuotanto" was used as a course name. However, at one (7%) institution "ohjelmistotekniikka" was used as a course name and at three (21%) institutions "ohjelmistotuotanto" was used as a specialization name. "Ohjelmistojärjestelmät" was used as a specialization name.

In addition, the English WWW pages, the study guides, or the course catalogs of three institutions were analyzed in more detail. The purpose was to find out how the concept "ohjelmistotekniikka" was defined and how it was translated into English. Based on the descriptions of the Helsinki University of Technology, the Tampere University of Technology and the University of Helsinki, the concept "ohjelmistotekniikka" was defined as a superordinate concept that covered several topics. The following topics were mentioned at least in one description or were required: artificial intelligence, compilers, computer graphics, databases, design and analysis of algorithms, distributed systems, embedded systems, operating systems, programming languages, software engineering, and usability. These topics cover or intersect with nine of the 14 areas of Computing Curricula 2001 (Table 3 in Section 2.7). The areas not covered were: Discrete Structures, Architecture and Organization, Net-Centric Computing, Social and Professional Issues, and Computational Science. Thus, the concept "ohjelmistotekniikka" could be interpreted as being almost as broad as the concept "computer science."

Interestingly, all three universities used different English translations for "ohjelmistotekniikka." The Tampere University of Technology used the translation "software systems," the Helsinki University of Technology "software techniques," and the University of Helsinki "software engineering."

Apparently, the Helsinki University of Technology was the only Finnish university-level institution that offered specialization in Software Systems. The description or the introductory text of the specialization was as follows (Helsinki University of Technology, 2004):

> The main goal of the major of software systems is to train professionals for designing and implementing reliable and efficient software. The central areas of teaching are data structures and algorithms and the following demanding programming technologies: compiler techniques, embedded systems, operating systems and database systems. The design methods of efficient programs and their analytical and experimental performance evaluation are the central contents of the major.

4.2.2 Background information of respondents

In this subsection, background information is presented only from those respondents who answered the second questionnaire targeted at the software systems professors and lecturers ($n = 10$). A more detailed description of the whole respondent group ($N = 19$) is presented later in Section 8.2.1.

The following background information was not typically asked from the respondents but found at their personal WWW pages or in various registers of the institution. Ninety percent of the respondents were male. On April 1, 2005, their ages varied from 36 to 63 years and the mean was 47.4 years. Forty percent of them were professors and the others were lecturers. Ninety percent had a Doctoral and 10% Master's degree. Sixty percent worked at the Helsinki University of Technology, 30% at the University of Helsinki, and 10% at the Tampere University of Technology.

4.2.3 Concept "software systems"

In the present subsection, only the results of Question 8 of the first questionnaire and Questions 8–18 of the second questionnaire that concern the concept "software systems" are presented. The questionnaires are available on the web page of the institution (Surakka, 2005b). During the first questionnaire round the respondents were not asked about "software systems" but they were asked about the Finnish concept "ohjelmistotekniikka." Most respondents understood "ohjelmistotekniikka" as a superordinate concept that included both software systems and software engineering topics.

The results of Questions 9 and 10 of the second questionnaire are presented in Table 4 and Table 5, respectively. Only the courses that were mentioned by at least three respondents are presented. In Table 4, the course Programming means various programming courses that were mentioned, for example "Basic Course in Programming" and "Programming in C."

Table 4. Courses that were most often mentioned in response to the question "Mention courses that belong in the center of the area of software systems" (n = 10).

Course	Proportion (%)
Operating Systems	80
Compilers	50
Distributed Systems	50
Programming	50
Data Structures and Algorithms	40
Concurrent Programming	30
Databases	30
Embedded Systems	30
Software Architecture	30

Table 5. Courses that were most often mentioned in response to the question "Mention courses that are borderline-cases for the area of software systems" (n = 10).

Course	Proportion (%)
Artificial Intelligence	70
Databases	40
Algorithm Analysis	30
Computer Graphics	30
Embedded Systems	30
Programming Languages	30

Counter-examples of the Finnish concept "ohjelmistotekniikka" were asked in Question 8 of the first questionnaire. This was a broader concept than "software systems" but close enough that these answers were relevant to the present thesis. The results are presented in Table 6. Only the courses that were mentioned by at least three respondents are presented.

Table 6. Courses that were mentioned most often in response to the question "Mention courses that are counter-examples for 'ohjelmistotekniikka' but still computer science and engineering." (N = 18)

Course	Proportion (%)
Computer Architecture	39
Usability	39
Signal Processing	28
Contents Production	22
Numerical Analysis	17
Theoretical Computer Science	17

Note. "Ohjelmistotekniikka" could be translated as software techniques, software technology, or software engineering.

The rest of the questions were used in the second questionnaire. Question 8 asked if the respondent knew of any definition for the concept "software systems" presented in the literature. All respondents selected the option "No." Question 11 was "Do you think the concepts 'software engineering' and 'software systems' are different?" All respondents selected the option "Yes." Question 12 was "How are these two concepts different?" Based on the answers, the respondents had a uniform opinion on differences. Typical comments were (Question 13):

- "Software engineering is the process to create software systems."
- "Software systems is a narrower concept. Software engineering includes software systems. Software business, team management, and development processes do not belong to software systems."
- "Software engineering concentrates more on the process and tools."

In Question 14, the respondents were asked to give a general grade for the following definition of a software system (Hordeski, 1978): "The entire set of programs and software development aids used in a microcomputer system." The scale was 1 = Poor, … , 4 = Excellent. The mean of the answers was 1.3 when all respondents gave the grade "Poor" or "Satisfactory." In Question 15, the respondents commented on the definition. Some of the comments were:

- "The word 'microcomputer' is strange, why not just 'computer?'"
- "Software development tools are not necessarily a part of a software system."
- "I consider 'software systems' as a field of education, not as a set of programs in a certain computer/computers."

In Question 16, the respondents were asked to give a general grade for the following definition that the author of the present thesis wrote using definitions for the terms "software" and "system" from the Institute of Electrical and Electronics Engineers (1990):

> A collection of computer programs, procedures, and possibly associated documentation and data organized to accomplish a specific function or a set of functions.

The mean of answers was 2.4 when the scale was 1 = Poor, … , 4 = Excellent. The distribution of the answers was: Poor 10%, Satisfactory 50%, Good 30%, and Excellent 10%. According to the Mann-Whitney test (Conover, 1999, pp. 271–275), the difference between these answers and the answers on Hordeski's (1978) definition was statistically significant ($p < .01$). Thus, this definition was evaluated as being better than Hordeski's definition. However, this definition had some problems as well. In Question 17, the respondents commented on the definition. Some of the comments were:

- "What are 'procedures' in this definition (if not parts of a program)?"
- "Do we want to have a definition for 'software systems' as a discipline or a 'software system' as an object?"
- "The 'system aspect' is not clear in the definition."
- "This does not correspond to a subject in a university."

In Question 18, the respondents were asked if they had any other comments concerning the concept "software systems." Two respondents asked if the concept was redundant and thought that it was enough to have just the concept "software." One respondent was amazed that the concept "software systems" was not defined. One respondent wrote: "There is a difference between 'software system' (technical term) and 'software systems' (educational field)."

4.3 Evaluation

First, the content validity is considered. The content validity of the results presented in relation to English course names in Section 4.2.3 is probably satisfactory because the respondents were asked to use typical English course names. This was a possible source of misunderstandings, because the native language of the respondents was Finnish or Swedish. However, this problem was not serious because it was likely that all respondents used English frequently in their work and were familiar with the English terms.

Second, the external validity of the results presented in Section 4.2.1 is considered. Obviously, for the Finnish concepts "ohjelmistotekniikka," "ohjelmistotuotanto," and "ohjelmistojärjestelmät," one should only consider whether or not the results can be generalized inside Finland. These results can probably be generalized to Finnish colleges as well, because Finnish colleges might use specialization and course names used in the Finnish universities, but not other way around.

Third, the external validity of the results presented in Section 4.2.3 is considered. The sample of respondents was intentionally biased because it was selected. However, getting a random sample from an average population is not even a typical goal in Delphi studies. All respondents were Finnish. This could have been the cause of the items related to telecommunications being evaluated as more central because telecommunication companies are important employers in Finland. However, based on the results, the respondents did not answer this way because the course Computer Networks is not among the central courses presented in Table 4, nor even among the borderline courses presented in Table 5. Therefore, it is reasonable to assume that the external validity of these results is good or, at least, satisfactory.

The stability of the results over time is probably quite good, or at least satisfactory, because the concepts in question have been used and the most commonly mentioned courses have been typically offered for several years. Generally, the meaning of new concepts might be more often contradictory or likely to be interpreted in different ways than the meaning of older concepts.

The results of the present research are not compared with previous publications because no previous concept analyses of the concept “software systems” were found (see Section 2.6).

5 Content analysis of degree requirements

In this section, the purpose was to solve how commonly specialization in Software Systems was offered in American computer science programs and which courses were the most commonly required in those specializations. In addition, the prerequisite relationships of these most common courses were explored. The main findings of the present research will be published later in the proceedings of Koli Calling conference (Surakka, in press-a).

First, the details of the research method are presented. Second, the results of the present research are presented and analyzed. Third, the research is briefly discussed.

5.1 Research method

In the present research, a quantitative content analysis of degree requirements was carried out; that is, the frequencies of different phrases such as "Software Systems" and "Artificial Intelligence" were simply counted when it was analyzed which were the most common specialization names, for example.

5.1.1 Sampling

The institutions were selected using the ranking lists by Geist, Chetuparambil, Hedetniemi, and Turner (1996, p. 98, Table 5) and U.S. News (2004, p. 72) for computer science Doctoral programs. The twenty best institutions of both ranking lists were selected. Actually, from U.S. News 24 institutions were selected because five institutions were tied in place 20. Altogether the number of selected institutions was 31 because some institutions were mentioned in both lists.

Another sampling strategy would be to use the list of accredited computer science programs. This sampling strategy was tested first when 20 accredited programs were selected randomly. However, only a few of these programs had specializations. This was a serious problem for the further analyses and the use of the accredited programs was rejected as a sampling strategy. For the purpose of the present research, the ranking lists worked much better because 20 (65%) of the selected 31 institutions had specializations for undergraduate programs. This is understandable because the number of advanced courses has to be quite large before specializations are beneficial or needed.

For the sample of undergraduate programs, Bachelor's in Science (BSc) in Computer Science program was selected if an institution offered several

undergraduate programs in computing. The closest alternative was selected if no such program was offered. For the sample of graduate programs, Master's in Science (MSc) in Computer Science was selected if an institution offered several graduate programs in computing. The closest alternative was selected if no computer science program was offered. The Doctoral program was selected if no Master's program was offered or the Master's program did not have specializations but the Doctoral program had. The all-around option such as "General Computer Science" was selected if an institution offered several MSc programs in computer science. A Master's program was selected regardless of whether it was terminal or not; that is, whether it was possible to continue in a Doctoral program. The selected institutions and degree programs are listed in Appendix C.

5.1.2 Coding

From each institution, the degree requirements of computing programs were sought from the web pages of the institution. The data were gathered in the years 2003–2005. Most of the data were gathered in the autumn of 2003 and therefore were typically from the academic year 2003–2004. Next, some details of coding of the specialization names, course names, and course prerequisites are explained.

Specializations or equivalent classifications were coded. An equivalent classification was, for example, areas of elective courses or areas used in course codes. Typically only the specialization names were used for coding and the course requirements were not read. However, the specialization name "Systems" was an exception to this rule because the course requirements were read as well. A specialization name "Systems" was converted as "Computer Systems" if it emphasized hardware topics and as "Software Systems" if it emphasized software topics. Similar specialization names were changed to typical names when possible. For example, the specialization name "Intelligent Systems" was changed to "Artificial Intelligence." The proportion was counted for each specialization when the purpose was to find the most common specializations. For example, 18 undergraduate programs offered specializations and 13 of them offered a specialization in Computer Systems. Thus, the proportion of Computer Systems was 72% for the undergraduate programs.

Next, it is explained how the proportions of the courses of the specializations in Software Systems were counted. The purpose was to determine which courses were required or offered as elective most often. It was typical that a specialization included from three to six required or elective courses. Each course was counted once regardless of whether it was required or elective. Course names were converted as typical course names if possible. These typical course names were selected so that the chosen name was

normally the most common among similar course names. Most of these conversions were simple. For example, the course name "Introduction to Compilers" was changed to "Compilers." Some course names were not typical and it was hard to classify the course based on the course name alone. In these cases, the course description was used for the name change if it was available. Some untypical course names were not converted because the course description was not available or the author was not able to classify the course. The proportion was counted for each course. For example, the undergraduate programs offered ten specializations in Software Systems and seven of these required the course Computer Networks or offered it as an elective. Thus, the proportion of Computer Networks was 70% for the undergraduate programs.

The proportions were counted for the course prerequisites as well. The purpose was to solve if there are strong relationships between some topics. The greater proportion indicates that a course was required more often as a prerequisite. The data were gathered from all 31 institutions of the sample, not just from those institutions that offered specializations in Software Systems. The proportions were not counted for all computer science courses but only for those courses that were the most common in the specializations in Software Systems. The analysis was not separated according to whether the course was offered in an undergraduate or a graduate program. The required and co-required courses were coded but the courses that were only recommended as prerequisites were not coded. The course names were converted as typical course names in the same manner as explained previously for the analysis of the courses. For example, 27 Compilers courses were found and 10 of them required Programming Languages as a prerequisite. Thus, the proportion of this relationship was 37%.

5.2 Results

First, information on the selected institutions and the degree programs is presented. Second, the most common specializations in the degree programs are presented. Third, the most common courses of specializations in Software Systems are presented. Finally, the prerequisites of these courses are presented.

5.2.1 Selected institutions and degree programs

According to the year 2000 edition of the Carnegie Classification of Institutions of Higher Education (Carnegie Foundation, 2005), all 31 selected institutions belonged to the category Doctoral/Research Universities—Extensive. This means that during the period the Carnegie Foundation studied the institutions, the institution awarded 50 or more Doctoral degrees per year across at least 15 disciplines. Thus, the sample was not representative of all institutions that

offered computing programs. In addition, it was counted from the information presented on the web pages of the Carnegie Foundation (2005) that 42% of the selected institutions were privately funded.

Eighty-four percent of the selected undergraduate programs were BSc programs and 16% were other programs. Seventy-seven percent of the selected graduate programs were MSc programs and 23% were PhD programs. In addition, it was counted from the information presented on the web pages of Accreditation Board for Engineering and Technology (2005a) that 39% of the selected undergraduate programs were accredited.

More details on the selected institutions and programs are presented in Appendix C.

5.2.2 Specializations

The most common specializations in the selected degree programs are presented in Table 7. The rows are ordered first according to the proportions of the column Undergraduate and then according to the column Graduate. As expected, the graduate programs offered specializations more often than the undergraduate programs. Out of 31 institutions, 29 (94%) graduate programs and 18 (58%) undergraduate programs offered specializations. The following four specializations are most common for both undergraduate and graduate programs: Computer Systems, Theoretical Computer Science, Software Systems, and Artificial Intelligence.

Table 7. Proportions (%) of offered specializations in selected degree programs.

Specialization	**Undergraduate** (n = 18)	**Graduate** (n = 29)
Computer Systems	72	55
Theoretical Computer Science	67	72
Software Systems	56	62
Artificial Intelligence	50	69
Scientific Computing	44	28
Programming Languages	39	28
Computer Graphics	28	35
Computer Networks	22	24
Algorithms	22	21
Databases	11	35
Software Engineering	11	14
Usability	11	14

5.2.3 Courses

The most common courses of the specializations in Software Systems are presented in Table 8. The courses are ordered first according to the column Undergraduate and then according to the column Graduate. As in Table 7, the

column Undergraduate refers to the undergraduate programs and Graduate to the graduate programs of the present research.

Table 8. Proportions (%) of most common courses of specializations in Software Systems.

Course	**Undergraduate** (n = 10)	**Graduate** (n = 18)
Computer Networks	70	44
Compilers	60	61
Databases	60	61
Operating Systems	60	44
Programming Languages	40	61
Software Engineering	40	22
Computer Architecture	40	17
Computer Graphics	40	17
Distributed Systems	30	50
Advanced Operating Systems	20	50

5.2.4 Course prerequisites

The prerequisites of the courses that were presented previously in Table 8 were analyzed. The results are presented in Figure 4. For example, the arrow between the courses Compilers and Programming Languages indicates that Programming Languages was the prerequisite of Compilers. The number inside the box indicates how many courses that had prerequisites presented on the web pages were found in the sample. The number above or right of the arrow is the proportion of how often the course was required as a prerequisite. For example, 27 Compilers courses were found and 10 (37%) of them required Programming Languages as a prerequisite. Only those prerequisite courses are shown that were required at least five times. At the lowest part of the figure are presented some typical freshman or sophomore courses that were required as prerequisites. The course "Mathematics" refers to various mathematics courses.

It can be noticed from Figure 4 that the course Data Structures and Algorithms was a prerequisite course for many of the other courses. In addition, apparently the courses Compilers, Distributed Systems, and Advanced Operating Systems were more advanced than the other intermediate or advanced courses because these three courses often required other courses as prerequisites.

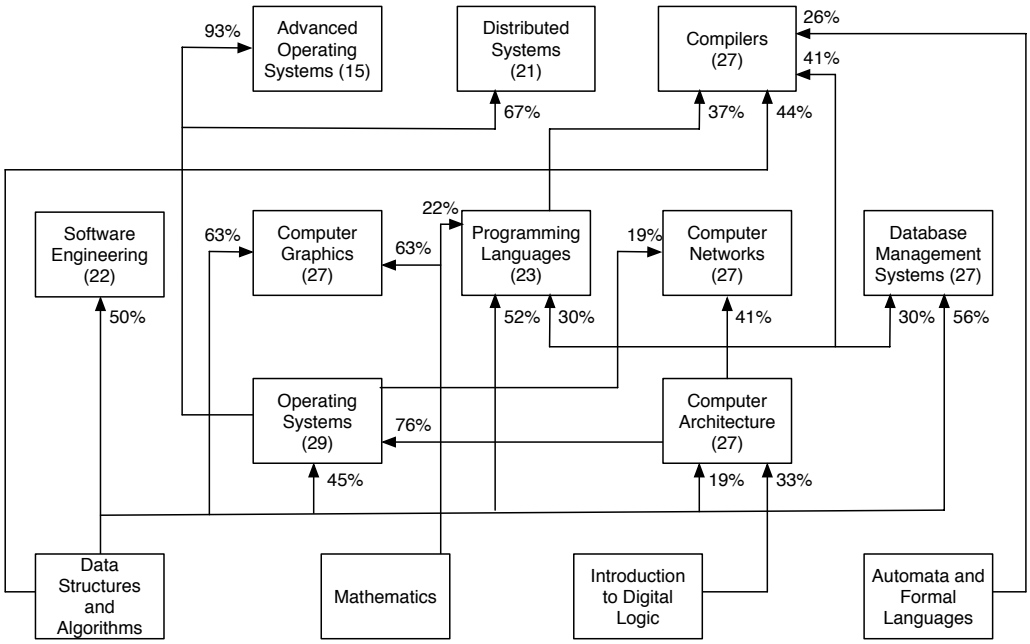

Figure 4. Prerequisites of most common courses of specializations in Software Systems.

5.3 Evaluation

The content validity of the present research is likely to be good, or at least satisfactory, because institutions are likely to require, or offer as elective, courses that are considered as important for future graduates. However, a problematic choice was the one that required courses and short lists of optional courses to be coded similarly. It would be a stronger indication of a necessary subject or skill if only required courses were coded. This principle was not applied for practical reasons. It was common that from none to only two courses were required in specializations. In addition, from two to three electives of a short list were typically required. Thus, the amount of data would be a lot smaller if only required courses were coded.

Next, the external validity is discussed:

- The present research could be classified as a benchmarking study because the sample of institutions was non-probabilistic and based on two ranking lists. Another sampling strategy would be use of accredited programs.
- The proportions of specializations presented in Table 7 would probably be considerably lower if the sample was all accredited computer science programs because specializations were apparently offered in top-level research universities more often. It is possible that the order of specializations would be different as well because more academically

oriented specializations such as Theoretical Computer Science and Artificial Intelligence might be more common in top-level research universities. Thus, the selected sampling strategy probably had some effect on the findings of the present research.

- There is no reason to assume that the course prerequisites presented in Figure 4 would be very different if different sampling were used. It is possible that the more advanced courses Advanced Operating Systems, Distributed Systems, and Compilers and more academic courses such as Programming Languages were offered in top-level research universities more often than in computer science programs on average. However, this should not have effect on the proportions because they were counted in relation to the courses that were offered.

The stability of the results over time is likely to be good, or at least satisfactory, because (a) the degree requirements of universities typically change slowly, and (b) the specialization and course names represent more abstract or broader concepts than, for example, individual skills such as Java required in job advertisements. Relationships between such abstract or broader concepts might be more stable over time.

The results of the present research are compared with the previous publications and possible differences are discussed as part of the general discussion in Section 21.1.4.

6 Triangulation: Concept analysis of "software systems" versus content analysis of degree requirements

In this section, the results of the concept analysis of "software systems" (Section 4.2) are compared with the results of the content analysis of degree requirements (Section 5.2.3). In addition, the main findings of Part II are summarized at the end of this section.

The results are presented in Table 9. The proportions of the courses that were evaluated as being the most central to the concept "software systems" are presented in the column Concept analysis. The columns Undergraduate and Graduate refer to the degree requirements of specializations in Software Systems in American universities. Some of these results were not presented previously because the proportions were so small. The rows are ordered first according to the column Concept analysis, second according to the column Undergraduate, third according to the column Graduate, and finally according to the course names. According to both sets of research results, the following courses were central or common: Operating Systems, Compilers, Distributed Systems, and Databases.

Table 9. Results of concept analysis of "software systems" versus results of content analysis of degree requirements. All results are proportions (%).

Course	Concept analysis (n = 10)	Content analysis	
		Undergraduate (n = 10)	Graduate (n = 18)
Operating Systems	80	60	44
Compilers	50	60	61
Distributed Systems	50	30	50
Programming	50	0	10
Data Structures and Algorithms	40	0	0
Databases	30	60	61
Concurrent Programming	30	10	22
Embedded Systems	30	0	0
Software Architecture	30	0	0
Computer Networks	10	70	44
Programming Languages	10	40	61
Software Engineering	10	40	22
Computer Architecture	10	40	17
Computer Graphics	0	40	17
Advanced Operating Systems	0	20	50

There were several differences as well. First, the courses where the proportions of the concept analysis were greater are discussed. According to the concept analysis, the courses "Programming" and "Data Structures and Algorithms"

were considered central but they were not commonly mentioned in the degree requirements. The course name "Programming" refers to basic programming courses. An obvious explanation is that these two courses are typically required already during basic studies; that is, before a specialization. According to the concept analysis, the courses Embedded Systems and Software Architecture were evaluated as being somewhat central but not according to the degree requirements. A possible explanation for embedded systems is that most of the respondents of the concept analysis worked at technical universities and embedded systems could be considered as being more important in engineering-oriented tasks.

A possible explanation for the difference of the course Software Architectures is that in general, various software engineering courses seemed to be less common in the sample of 31 American research universities than in three Finnish universities where the respondents of the concept analysis worked. This was not properly analyzed but was only the author's anecdotal observation during the content analysis of degree requirements.

Second, the courses where the proportions of the concept analysis were smaller are discussed. The courses Computer Networks and Programming Languages were mentioned in the degree requirements often but they were not central according to the concept analysis. A possible explanation could be that both these course subjects were used as specialization names as well. As a consequence, a respondent might classify them at the same level as the concept "software systems" when the courses might be classified as subordinate concepts for the concept "software systems." The course Advanced Operating Systems was not mentioned in the concept analysis at all but it was often mentioned in the degree requirements of the graduate programs. A probable explanation is that an open question was used in the content analysis. In that setting, it was natural if a respondent first thought about the course Operating Systems and wrote its name on the questionnaire. After that, a respondent might move on to consider whether some other common course such as Databases was central enough.

Main findings of Part II

The main findings of Part II are:

- According to the content analysis of degree requirements, Software Systems is a common specialization in American research universities.
- According to both sets of research results, the following courses were central or common for a specialization in Software Systems: Operating Systems, Compilers, Distributed Systems, and Databases.

Part III: Questionnaires

The results of the questionnaires are presented in Part III. The respondent groups were software developers, professor and lecturers, and Master's students. In addition, the triangulation of the questionnaire research is presented.

The purpose of the research presented in Sections 7–0 was to solve the main problem of the present thesis. The main problem was: What technical skills do graduates from specialization in Software Systems need in their work after graduation? Section 10 is only moderately related to the research problems of the present thesis. This is explained more at the beginning of Section 10.

The scope of this part of the thesis is presented in Figure 5 using the data sources and research methods. The figure is the same as Figure 2 presented previously in Section 3.2 but the boxes related to this part are gray.

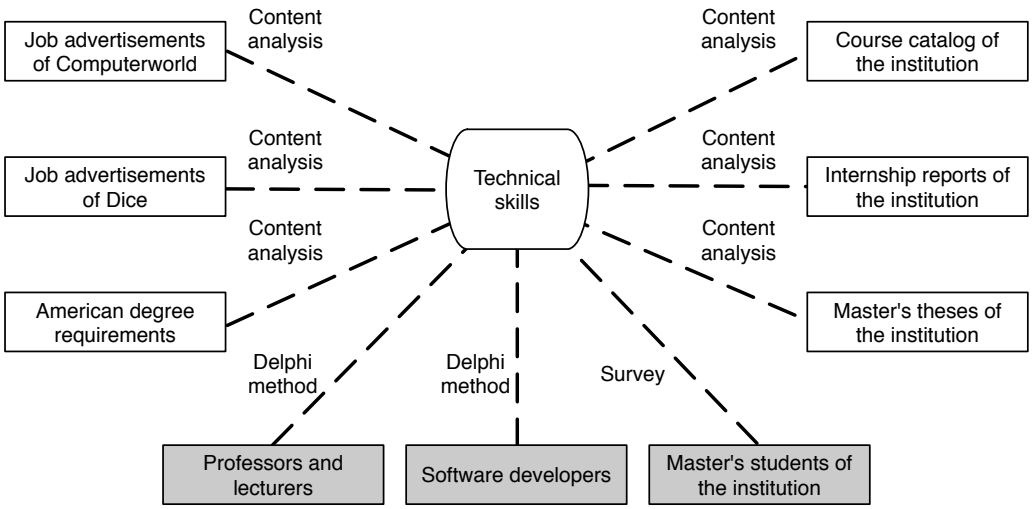

Figure 5. Scope of Part III of thesis.

7 Delphi study targeted at software developers: Technical skills

Eleven experienced Finnish software developers evaluated the importance of various subjects and skills related to software development. The purpose was to solve the main problem of the present thesis: What technical skills do graduates from specialization in Software Systems need in their work after graduation? The present research was conducted between November 2003 and January 2004 as part of a Delphi study. The results concerning 42 technical subjects or skills are presented in this section. The main findings of the present study will be published later in Communications of the ACM (Surakka, in press-b).

7.1 Research method

The Delphi method was selected for the present research because it was suitable for analyzing cognitive skills, which will be explained later in Section 10. Two questionnaire rounds were conducted. However, in this section, only some results from the first round are presented.

7.1.1 Finding respondents

The goal was to find between 10 and 20 particularly good software developers. The respondents were found using recommendations. Thus, they were not a statistically representative sample of all software developers but a focus group. Kitchenham and Pfleeger (2002, p. 19) wrote that one reason for using a non-probabilistic sample is that the target population is hard to identify. The situation was like that in the present study because it was difficult to identify the target group using properties such as age, education, and title. For example, the title and age were not enough to separate particularly good software developers from intermediate developers.

The minimum criteria were a degree, five years working experience after graduation, at least half of the time dedicated to programming during these five years, and at least 100,000 lines of self-implemented code. In addition, at least half of the respondents should have versatile software development experience. Here, versatile means projects of different kinds, for example using various programming languages. Two extra criteria were that (a) a maximum of three respondents could work for the same organization and (b) only one respondent could work full-time at the Helsinki University of Technology. The degree could be other than a computer science degree. For example, some older candidates had the degree from electrical engineering. The title of the respondent did not need to be programmer, software developer, or software

engineer, since the important issue was only that their work included enough programming.

The recommendations were gathered using three different methods simultaneously:

- Some professors and lecturers of the Helsinki University of Technology and the University of Helsinki were asked to make recommendations. These professors and lecturers were selected from the area of software systems. In some cases, a professor or lecturer gave the name of some manager or director of a company that could be asked for recommendations.
- Approximately ten Finnish companies were mailed a letter or sent an electronic mail to named persons who were asked if the company would like to participate in such a study and for recommendations for suitable candidates. The contact persons of the companies were found partly from the web pages of the companies and partly they were obtained from the Career Services of the institution. These persons were technical directors and managers or from the personnel department; that is, they were not considered as possible candidates but they were asked to help to find candidates.
- The so-called snowball method was applied. That is, the candidates who were recommended first and promised to take part were asked to recommend other possible candidates.

Altogether, 59 persons were recommended. Forty of them were not asked, for several different reasons, for example, the person had graduated less than five years ago. Thus, 19 persons were asked to participate. From these 19 persons, 11 agreed to participate.

In most cases, the criterion of any degree was checked from the student register of the institution or the personal WWW pages of the candidates. The criteria of at least five years of working experience after graduation, at least 100,000 lines of self-implemented code, and enough programming experience during the past five years were checked when a candidate was asked to take part. In other words, a candidate was simply asked to evaluate himself if he fulfilled the criteria. Some candidates declined because of these three conditions. The criterion that at least half of the respondents should have versatile software development experience was controlled with the first questionnaire. More than half of the respondents had versatile software development experience (see Section 7.2.1). Thus, no respondents were excluded because of this criterion.

7.1.2 Planning of questionnaire

Forty-two items used in Question 15 of the questionnaire were selected using group work and previous literature. These 42 items are listed later in Table 10 in Section 7.2.1. In addition, the questionnaire is available on the web page of

the institution (Surakka, 2005b). Three members of the group had a Doctoral degree in computer science and worked as professors or lecturers at the institution. The purpose was to select subjects and skills that were commonly required in computer science programs or might be important for software developers. The same 42 question items were used in all three questionnaires that are presented in Sections 7–9. The planning of Question 15 is explained in detail in Appendix D.

7.1.3 Questionnaire

Two questionnaire rounds were conducted. The questionnaires are available on the web page of the institution (Surakka, 2005b). The first questionnaire was answered between November 2003 and January 2004. During the first round, most respondents answered so that the author was present while they answered. Thus, the respondents were able to ask questions. The mean answering time was one hour and six minutes. The questionnaire was in Finnish. The main properties of the questionnaire are presented next.

The questionnaire had 14 open questions and 14 multiple-choice questions. The topics were (a) background information about the respondent, (b) the importance of various subjects and skills for software development such as discrete mathematics and concurrent programming, (c) cognitive skills, (d) problem-solving techniques, and (e) software quality. For brevity, only results about the background information and importance of various subjects and skills are presented in this section.

The questions about background information were the title, the proportion of time used in programming, the number of employees under the respondent, lines of code implemented by the respondent, the number of different groups involved, the number of different projects, personal skills in the same 42 items that were used in Question 15, skills in various programming languages, and knowledge of various operating systems.

Question 15 was as follows “How important do you think the following subjects and skills are for demanding programming tasks?” Below this question was a table where 42 items such as discrete mathematics and concurrent programming were listed. The answering options were: “Not at all important,” “A little important,” “Somewhat important,” and “Very important.”

7.1.4 Statistical analysis

The Mann-Whitney test was used. The results of the test are presented in the same table with the means for brevity. This was a problematic choice because a reader might wrongly interpret that the differences between the means were statistically significant. The confidence level $1 - \alpha = .99$ was used to avoid Type I errors because the number of items in Question 15 was so large.

7.2 Results

First, background information on the respondents is presented. Second, the results of the respondents' opinions of the importance of various subjects and skills are presented.

7.2.1 Background information of respondents

All respondents were male and the mean of respondents' ages was 37.1 years. Their degrees were as follows: one college degree in computer science and engineering (9%), five masters in computer science and engineering (45%), three masters in other engineering disciplines (27%), one doctor in applied mathematics (9%) and one doctor in computer science and engineering (9%). The respondents' positions were distributed into the following groups: senior software engineers and developers 45%, researchers 27%, and managers or directors 27%.

Each respondent was asked how many projects he had participated in and how well he could program with various programming languages. A respondent was classified to have versatile software development experience if he had good or excellent skills in at least three programming languages from different programming paradigms (typically C, C++, and Lisp). Apparently some respondents knew only one or two programming paradigms well. According to the answers, six (55%) respondents were classified as having versatile software development experience.

Each respondent was asked to give himself a grade in the same 42 subjects or skills that were used in Question 15. The means of the respondents' answers are presented in Table 10. The means are divided into the same four categories that were used in the question. Inside each category, the rows are ordered first according to the mean and then according to the name of the subject or skill.

The mean across the range of items was 2.6 ($N = 461$). The values of T_1 of the Mann-Whitney test between the answers of a single item ($N = 10$ or $N = 11$) and the answers across the range of items ($N = 461$) are presented in the column Mann-Whitney. Two asterisks (**) indicate that the difference between a single item and the answers across the range of items was statistically significant ($p < .01$).

There are three issues that are worth noticing. First, script programming skills were ranked very highly. This obviously correlates with the heavy use of the Unix environment in their work. The second observation is that functional programming was ranked much higher than the general use of functional programming languages in software production would indicate. It is possible that this is related to multiskilling. A plausible explanation is that many of the respondents have used functional programming during their careers or for

hobby programming. Based on answers to the open question about working experience (Question 8), at least four (36%) respondents had actually used Lisp in some work project.[1] Third, it is interesting that the means of embedded systems and real-time systems were quite low because most of the respondents had an engineering degree. One could expect that engineers work in engineering-oriented tasks where low-level programming skills were needed. Apparently, this was not the case for this respondent group.

[1] Nine (82%) respondents graduated from the institution where Scheme was the language of the first programming course in the Degree Program of Computer Science and Engineering (CSE) during 1989–2003. However, this was not a suitable explanation because these nine respondents were admitted before 1989 or were from degree programs other than CSE. That is, the course in question was not compulsory for them.

Table 10. Means and results of Mann-Whitney test of question "Give yourself a grade in the following subjects or skills" (scale: 1 = Poor, … , 4 = Excellent).

Subject or skill	*M*	Mann-Whitney
Mathematics, physics, and theoretical computer science:		
Other areas of theoretical computer science (e.g., automata)	2.5	-0.5
Logic (in particular, propositional and predicate logic)	2.5	-0.8
Discrete mathematics	2.4	-1.3
Physics	2.1	-2.5
Mathematics for continuous systems	1.8	-3.5**
More technical or part of the operational systems:		
Procedural programming	3.8	5.0**
Data structures and algorithms	3.5	3.6**
Script programming	3.5	3.6**
Object-oriented programming	3.4	3.0**
Operating systems	3.1	2.0
Functional programming	3.0	1.6
Internet protocols	2.8	0.9
Systems programming	2.8	0.7
Compilers	2.7	0.3
Computer/data security	2.6	-0.1
Implementing techniques of WWW systems	2.6	0.0
Software architectures	2.6	0.1
Computer architecture	2.5	-0.5
Implementing techniques of user interfaces	2.5	-0.9
Embedded systems	2.4	-1.1
Artificial intelligence and knowledge engineering	2.3	-1.5
Concurrent programming	2.3	-1.6
Database management systems	2.3	-1.6
Distributed systems	2.3	-1.6
Logic programming	2.2	-2.1
Computer graphics	2.1	-2.4
Extensible Markup Language (XML) techniques	2.1	-2.5
Telecommunications techniques other than Internet protocols	2.1	-2.3
Real-time systems	2.0	-2.9**
Software engineering (different phases of life cycle):		
Implementation	3.8	5.0**
Design	3.4	3.1**
Testing	3.1	2.0
Requirements	2.9	1.2
Concept exploration	2.6	-0.1
Approval	2.3	-1.5
Operation and maintenance	2.3	-1.4
Installation and checkout	2.1	-2.3
Packaging and delivery	1.7	-3.9**
Retirement	1.5	-4.5**
Software engineering (possible in several phases):		
Version and configuration management	3.1	2.0
Project management	2.8	0.7
Documenting	2.7	0.5

Note. $N = 11$ expect $N = 10$ for the item "Computer/data security."
**$p < .01$

7.2.2 Importance of various subjects and skills

The means of the question "How important do you think the following subjects and skills are for demanding programming tasks?" are presented in Table 11. The results were divided into the four categories that were used in the question. Within each category, the rows are first ordered according to the mean and then according to the name of the subject or skill.

The mean across the range of items was 2.9 ($N = 460$). The values of T_1 of the Mann-Whitney test between the answers of a single item ($N = 10$ or $N = 11$) and the answers across the range of items ($N = 460$) are presented in the column Mann-Whitney. Two asterisks (**) indicate that the difference between a single item and the answers across the range of items was statistically significant ($p < .01$).

As expected, the respondents evaluated (a) theoretical computer science, logic, and discrete mathematics as being more important than continuous mathematics and physics, and (b) basic skills such as procedural programming, object-oriented programming, and data structures and algorithms as being important or very important. Not surprisingly, from different phases of software development, implementation and the phases near it were evaluated as being important.

Table 11. Means and results of Mann-Whitney test of the question "How important do you think the following subjects and skills are for demanding programming tasks?" Scale: 1 = Not at all important, … , 4 = Very important.

Subject or skill	*M*	Mann-Whitney
Mathematics, physics, and theoretical computer science:		
Other areas of theoretical computer science (e.g., automata)	3.3	1.4
Logic (in particular, propositional and predicate logic)	2.8	-0.8
Discrete mathematics	2.6	-1.6
Mathematics for continuous systems	2.0	-4.2**
Physics	1.6	-5.7**
More technical or part of the operational system:		
Data structures and algorithms	3.8	4.2**
Procedural programming	3.8	4.2**
Object-oriented programming	3.6	3.3**
Software architectures	3.5	2.4
Internet protocols	3.4	1.9
Script programming	3.4	1.9
Operating systems	3.3	1.7
Computer/data security	3.2	1.0
Systems programming	3.2	1.2
Compilers	3.1	0.8
Concurrent programming	3.1	0.8
Distributed systems	3.1	0.5
Computer architecture	3.0	0.3
Database management systems	2.7	-1.3
Extensible Markup Language (XML) techniques	2.7	-1.0
Implementing techniques of user interfaces	2.7	-1.3
Implementing techniques of WWW systems	2.7	-1.3
Functional programming	2.6	-1.5
Real-time systems	2.6	-1.5
Embedded systems	2.5	-1.7
Logic programming	2.3	-3.1**
Telecommunications techniques other than Internet protocols	2.0	-4.2**
Computer graphics	1.9	-4.9**
Artificial intelligence and knowledge engineering	1.6	-5.5**
Software engineering (different phases of life cycle):		
Design	3.7	3.7**
Implementation	3.7	3.7**
Requirements	3.6	3.3**
Test	3.5	2.4
Concept exploration	3.0	0.1
Approval	2.6	-1.8
Operation and maintenance	2.5	-2.4
Installation and checkout	2.3	-3.1**
Packaging and delivery	1.9	-4.6**
Retirement	1.8	-4.8**
Software engineering (possible in several phases):		
Version and configuration management	3.6	3.3**
Project management	3.2	1.0
Documenting	3.0	0.1

Note. $N = 11$ expect $N = 10$ for the items "Discrete math." and "Math. for continuous systems."
**$p < .01$

7.3 Evaluation

The content validity of the 42 question items reported in Table 10 and Table 11 was seriously considered when the questionnaire was planned (see Appendix D). Based on the respondents' behavior during the answering, this apparently succeeded reasonably well. The respondents answered these questions fast and typically did not have problems in understanding the items. The concepts "continuous mathematics" and "discrete mathematics" were not familiar enough for one respondent, who left these items unanswered. This respondent was the only one who had an Associate Degree. Some other respondents also had problems with the item "Continuous mathematics," but they were able to answer after the author of the present thesis gave some advice.

Next, the external validity of the present study is discussed:

- Forty-five percent of the respondents worked for telecommunication companies. As a consequence, it is possible that the importance of the following subjects was emphasized: computer/data security, concurrent programming, distributed systems, Internet protocols, and telecommunications techniques other than Internet protocols. In Finland, this is not a serious problem for the generalization of the results because telecommunication companies, in particular Nokia, actually are large employers in the field of information technology. However, this is a problem for the generalization of the results to other countries.
- The original goal was to get several respondents who had graduated from institutions other than the Helsinki University of Technology. This succeeded poorly because only two respondents had graduated from other institutions. This is a problem for the generalization of the results as well. At the institution, Unix has been the dominant operating system in computer science education since 1986. Based on the answers of the respondents, most of them have continued to work in a Unix environment after graduation. This might have increased the importance of the following subjects: operating systems, script programming, and systems programming.

The stability of the results over time is probably modest, or satisfactory at best. As will be shown later in Section 12, the proportions of most common skill categories have increased during the past 15 years in job advertisements targeted at software developer positions. This gives a reason to suspect that respondents similar to those in the present study might answer differently in the year 2015, for example.

The results of the present study are compared with the previous findings and possible differences are discussed as part of the general discussion in Section 21.1.3.

8 Delphi study targeted at professors and lecturers

Nineteen Finnish professors and lectures evaluated the importance of various subjects and skills related to software development. The purpose was to solve the main problem of the present thesis: What technical skills do graduates from specialization in Software Systems need in their work after graduation? The present research was conducted in January and February 2005 as part of the Delphi study. The main findings of the present study will be published later in Communications of the ACM (Surakka, in press-b).

8.1 Research method

The results of this section were obtained from the same Delphi study that was used for the concept analysis. As was explained at the beginning of Section 4, the Delphi method was selected because it was suitable for the concept analysis. Two questionnaire rounds were conducted. The questionnaires are available on the web page of the institution (Surakka, 2005b). In this section, only a part of the results are presented. The Mann-Whitney test was used to analyze the results.

8.2 Results

First, some background information on the respondents is presented. Second, the results of the respondents' opinions on the importance of various subjects and skills are presented.

8.2.1 Background information of respondents

Background information on the respondents ($N = 19$) is presented in this subsection. Part of the background information was not asked from the respondents but found at their personal WWW pages or in various registers of the institution. Eighty-four percent of the respondents were male. On April 1, 2005, their ages varied from 32 to 63 years, the mean was 45.4 years. Fifty-three percent of them were professors and the others were lecturers. Seventy-nine percent had a Doctoral and 21% a Master's degree. Fifty-eight percent worked at the Helsinki University of Technology, 26% at the University of Helsinki, and 16% at the Tampere University of Technology.

Each respondent was asked to give himself a grade in 42 subjects or skills. The means from the respondents' answers are presented in Table 12. The rows are divided into the same four categories that were used in the question. Inside each category, the rows are ordered first according to the mean and then according to the name of the subject or skill.

The mean across the range of items was 2.4 ($N = 796$), which was smaller than the corresponding mean 2.9 of the software developers. This was as expected because the professors and lecturers were probably more often experts in only one or two relatively narrow topics such as data structures and algorithms when the goal was that the software developers should be multiskilled; that is, have good skills in several areas.

The values of T_1 of the Mann-Whitney test between the answers of a single item ($N = 18$ or $N = 19$) and the answers across the range of items ($N = 796$) are presented in the column Mann-Whitney. Two asterisks (**) indicate that the difference between the answers for a single item and the answers across the range of items was statistically significant ($p < .01$). The confidence level $1 - \alpha = .99$ was used to avoid Type I errors because the number of items was so large.

As in the results of the software developers, the professors and lecturers were evaluated to have the best skills in basic topics such as procedural programming, object-oriented programming, and data structures and algorithms. The order of different software development phases is similar to the software developers' results.

Table 12. Means and results of Mann-Whitney test to the question "Give yourself a grade in the following subjects or skills" (scale: 1 = Poor, … , 4 = Excellent).

Subject or skill	*M*	Mann-Whitney
Mathematics, physics, and theoretical computer science:		
Logic (in particular, propositional and predicate logic)	2.8	1.9
Discrete mathematics	2.7	1.7
Other areas of theoretical computer science (e.g., automata)	2.7	1.6
Mathematics for continuous systems	2.3	-1.1
Physics	1.9	-3.0**
More technical or part of the operational system:		
Procedural programming	3.6	6.2**
Data structures and algorithms	3.3	4.5**
Object-oriented programming	3.3	4.2**
Operating systems	2.9	2.5
Software architectures	2.7	1.0
Compilers	2.6	0.8
Concurrent programming	2.6	0.8
Functional programming	2.6	0.5
Computer architecture	2.5	0.0
Distributed systems	2.5	0.0
Script programming	2.5	0.2
Database management systems	2.4	-0.4
Systems programming	2.4	-0.6
Implementing techniques of user interfaces	2.3	-1.2
Logic programming	2.3	-0.9
Implementing techniques of WWW systems	2.2	-1.5
Artificial intelligence and knowledge engineering	2.0	-2.7**
Computer/data security	1.9	-3.3**
Embedded systems	1.9	-3.2**
Real-time systems	1.9	-2.9**
Computer graphics	1.8	-3.8**
Extensible Markup Language (XML) techniques	1.8	-3.6**
Internet protocols	1.8	-3.5**
Telecommunications techniques other than Internet protocols	1.5	-5.2**
Software engineering (different phases of life cycle):		
Design	3.2	3.6**
Implementation	3.1	3.6**
Concept exploration	2.8	2.0
Requirements	2.7	1.5
Test	2.6	0.9
Approval	2.2	-1.5
Operation and maintenance	2.2	-1.7
Packaging and delivery	2.0	-2.7**
Installation and checkout	1.9	-3.2**
Retirement	1.6	-5.3**
Software engineering (possible in several phases):		
Project management	2.8	1.6
Documenting	2.7	1.5
Version and configuration management	2.5	0.0

Note. $N = 19$ expect $N = 18$ for the items "Script programming" and "Retirement."
**$p < .01$

8.2.2 Importance of various subjects and skills

The means from the question "How important do you think the following subjects and skills are for graduates?" are presented in Table 13. The results were divided into the four categories that were used in the question. Within each category, the rows are first ordered according to the mean and then according to the name of the subject or skill.

The mean across the range of items was 3.0 ($N = 787$). The values of T_1 of the Mann-Whitney test between the answers of a single item ($N = 18$ or $N = 19$) and the answers across the range of items ($N = 787$) are presented in the column Mann-Whitney. Two asterisks (**) indicate that the difference between the answers for a single item and the answers across the range of items was statistically significant ($p < .01$). The confidence level $1 - \alpha = .99$ was used to avoid Type I errors because the number of items was so large.

Next, the items where the differences were statistically significant are listed. The respondents evaluated the following subjects or skills as being important and the differences were statistically significant: data structures and algorithms, object-oriented programming, operating systems, procedural programming, software architectures, distributed systems, project management, and "version and configuration management." In addition, they evaluated the software development phases implementation, design, and test as being important.

The respondents evaluated the following subjects or skills as being less important and the differences were statistically significant: mathematics for continuous systems, physics, real-time systems, logic programming, Extensible Markup Language (XML) techniques, artificial intelligence and knowledge engineering, Internet protocols, computer graphics, and telecommunications techniques other than Internet protocols. In addition, they evaluated the software development phases "packaging and delivery" and retirement as being less important.

Table 13. Means and results of Mann-Whitney test of the question "How important do you think the following subjects and skills are for graduates?" Scale: 1 = Poor, … , 4 = Excellent. $N = 18$ or $N = 19$.

Subject or skill	*M*	Mann-Whitney
Mathematics, physics, and theoretical computer science:		
Discrete mathematics	3.1	-0.4
Logic (in particular, propositional and predicate logic)	2.9	-0.9
Other areas of theoretical computer science (e.g., automata)	2.9	-1.4
Mathematics for continuous systems	1.7	-6.4**
Physics	1.5	-7.5**
More technical or part of the operational system:		
Data structures and algorithms	3.9	5.5**
Object-oriented programming	3.9	5.5**
Operating systems	3.7	3.8**
Procedural programming	3.7	3.8**
Software architectures	3.6	3.4**
Concurrent programming	3.5	2.4
Distributed systems	3.5	2.6**
Database management systems	3.3	1.0
Computer architecture	3.2	0.3
Compilers	3.1	-0.3
Implementing techniques of user interfaces	3.1	0.0
Computer/data security	3.0	-0.5
Functional programming	2.9	-1.3
Systems programming	2.9	-1.1
Script programming	2.8	-1.7
Implementing techniques of WWW systems	2.8	-2.1
Embedded systems	2.7	-2.3
Real-time systems	2.7	-2.7**
Logic programming	2.6	-3.1**
Extensible Markup Language (XML) techniques	2.5	-3.5**
Artificial intelligence and knowledge engineering	2.5	-3.8**
Internet protocols	2.3	-4.6**
Computer graphics	2.3	-4.8**
Telecommunications techniques other than Internet protocols	2.0	-5.7**
Software engineering (different phases of life cycle):		
Implementation	3.9	4.9**
Design	3.8	4.6**
Test	3.8	4.6**
Requirements	3.4	1.9
Concept exploration	3.4	2.2
Approval	2.9	-0.5
Operation and maintenance	2.9	-1.1
Installation and checkout	2.8	-1.4
Packaging and delivery	2.3	-3.8**
Retirement	2.3	-3.8**
Software engineering (possible in several phases):		
Project management	3.6	3.4**
Version and configuration management	3.6	3.1**
Documenting	3.4	1.6

**$p < .01$

8.2.3 Changes in the future

During the second questionnaire round, the respondents were asked how the significance of various subjects and skills would change during the next 20 years in education. The answering options were: Decrease a lot, Decrease somewhat, Stay at the same level, Increase somewhat, and Increase a lot.

The results are presented in Table 14. The results were divided into the four categories that were used in the question. Within each category, the rows are first ordered according to the mean and then according to the name of the subject or skill. The mean across the range of items was 3.3 ($N = 752$) when the scale was: 1 = Decrease a lot, … , 5 = Increase a lot. The values of T_1 of the Mann-Whitney test between the answers of a single item ($N = 16, 17, 18$) and the answers across the range of items ($N = 752$) are presented in the column Mann-Whitney. Two asterisks (**) indicate that the difference between the answers of a single item and the answers across the range of items was statistically significant ($p < .01$). The confidence level $1 - \alpha = .99$ was used to avoid Type I errors because the number of items was so large.

Next, some statistically significant differences are listed. The respondents estimated that the significance of physics and mathematics for continuous systems would decrease somewhat in the future. In addition, inside the category "More technical or part of the operational system," the mean of systems programming was quite low.

The items that were estimated to increase somewhat in the future were: computer/data security, concurrent programming, distributed systems, embedded systems, script programming, and software architectures. The means of these items were greater than or near 4.0.

In addition, the means of several items of the category "More technical or part of the operational system" and one item of the category "Software engineering (different phases of life cycle)" were less than 3.0 and the differences were statistically significant. However, these means were so near 3.0 that it would be better to interpret that generally the respondents evaluated that the significance would stay at the same level.

Table 14. Means and results of Mann-Whitney test of the question "Will the significance of the following subjects or skills decrease, increase, or stay at the same level during the next 20 years in education?"
Scale: 1 = Decrease a lot, … , 5 = Increase a lot. N = 16, 17, 18.

Subject or skill	***M***	**Mann-Whitney**
Mathematics, physics, and theoretical computer science:		
Logic (in particular, propositional and predicate logic)	3.5	0.9
Other areas of theoretical computer science (e.g., automata)	3.5	1.0
Discrete mathematics	3.4	-0.2
Mathematics for continuous systems	2.2	-6.5**
Physics	2.1	-6.5**
More technical or part of the operational system:		
Distributed systems	4.3	5.9**
Computer/data security	4.1	4.5**
Concurrent programming	4.1	4.7**
Embedded systems	3.9	3.4**
Script programming	3.8	2.9**
Software architectures	3.8	2.9**
Implementing techniques of user interfaces	3.7	2.5
Database management systems	3.6	1.5
Real-time systems	3.6	1.5
Computer graphics	3.4	0.4
Implementing techniques of WWW systems	3.4	1.0
Extensible Markup Language (XML) techniques	3.3	0.4
Artificial intelligence and knowledge engineering	3.2	-0.6
Data structures and algorithms	3.1	-1.6
Logic programming	3.1	-1.6
Object-oriented programming	3.1	-1.5
Telecommunications techniques other than Internet protocols	3.1	-1.2
Functional programming	2.9	-2.3
Operating systems	2.9	-2.6**
Procedural programming	2.9	-3.0**
Compilers	2.8	-2.9**
Computer architecture	2.7	-4.0**
Internet protocols	2.7	-3.7**
Systems programming	2.4	-5.4**
Software engineering (different phases of life cycle):		
Test	3.7	2.4
Concept exploration	3.6	1.6
Requirements	3.6	1.7
Approval	3.4	0.2
Design	3.4	0.3
Operation and maintenance	3.3	0.1
Retirement	3.3	-0.5
Installation and checkout	3.2	-0.6
Packaging and delivery	3.0	-2.0
Implementation	2.8	-3.4**
Software engineering (possible in several phases):		
Version and configuration management	3.6	1.2
Project management	3.4	0.3
Documenting	3.1	-1.8

**$p < .01$

8.3 Evaluation

First, the content validity of the present study is discussed. As was explained previously in Section 7.3, the content validity of the 42 question items presented in Table 12, Table 13, and Table 14 was seriously considered when the questionnaire was planned (see Appendix D). Based on the respondents' behavior during answering, this apparently succeeded well. The respondents answered these questions fast and typically did not have problems in understanding the items. However, one respondent left all ten items of the category "Software engineering (different phases of life cycle)" unanswered in Question 12 because the area was not familiar enough for him. In Question 11 (Table 12), this respondent answered that he had excellent knowledge in procedural programming, operating systems, and computer architecture. The other respondents did not report major difficulties in answering.

Second, the external validity of the present study is discussed:

- All respondents were Finnish. This could have caused that the items related to telecommunications were evaluated as being important because telecommunication companies are important employers in Finland. However, based on the results, the respondents did not answer this way. The means of Internet protocols and the item "Telecommunications techniques other than Internet protocols" were 2.3 and 2.0, respectively, which are quite low.
- Seventy-four percent of the respondents worked at technical universities. It is possible that graduates from technical universities worked more often in positions where low-level programming skills were necessary, which might have an effect on the answers. As a consequence, the means of embedded systems and real-time systems might be greater than otherwise. However, both these means were 2.7, which was not very high.
- The respondents might have emphasized the more academic items; that is, typical subjects for universities but not for colleges or polytechnics. As a consequence, the means of the following subjects and skills might be greater: artificial intelligence and knowledge engineering, discrete mathematics, functional programming, logic (in particular, propositional and predicate logic), logic programming, and other areas of theoretical computer science (e.g., automata). The means of these items were from 2.5 to 3.1.

The stability of the results over time is evaluated in exactly same manner as previously in Section 7.3. The stability is probably modest or satisfactory at best. As will be shown later in Section 12, the proportions of most common skill categories have increased during the past 15 years in job advertisements targeted at software developer positions. This gives a reason to suspect that

similar respondents as in the present study might answer differently in the year 2015, for example.

The results of the present study are compared with the previous findings; possible differences are discussed as part of the general discussion in Section 21.1.4.

9 Survey targeted at Master's students

A small-scale survey was conducted of the Master's students who were in the process of graduating from the specialization in Software Systems at the institution. The purpose was to solve the main problem of the present thesis: What technical skills do graduates from specialization in Software Systems need in their work after graduation? The students ($N = 25$) answered the survey between January 2004 and March 2005. The main findings of the present survey will be published later in Communications of the ACM (Surakka, in press-b).

9.1 Research method

The survey was used in the present research instead of, for example, interviews in order to be able to compare the results on the importance of various subjects and skills with the results of the Delphi studies targeted at software developers and professors and lecturers. The Delphi method was not used because organizing the second questionnaire round would be difficult. The data were collected during a period that was longer than one year in order to get a large enough sample. The period was so long because the students graduated randomly during the period, not just at the end of terms.

The goal was to collect data from all members of the target population. Sampling was not used because the target population was so small. The research is a descriptive survey.

First, the questionnaire used is presented. Second, it is explained how the data were gathered.

9.1.1 Questionnaire

The questionnaire is available on the web page of the institution (Surakka, 2005b). The questionnaire had compulsory and voluntary parts. Altogether, it had 28 open questions and 19 multiple-choice questions. The topics were (a) background information of the respondent, (b) evaluating the contents of the specialization as a whole, (c) the importance of various subjects and skills for thesis project and future work, (d) Master's thesis project, and (e) evaluating the respondent's own skills and knowledge on various skills and subjects. In the present thesis, only the results of some questions of background information, the importance of various subjects and skills, and Master's thesis project are reported.

In Question 20, the importance of subjects and skills was asked from two viewpoints, for the Master's thesis and for the work during the next 12

months. It was assumed that the respondents were able to reliably evaluate their future work because (a) most respondents continued to work in the same company where they completed the thesis and (b) most respondents continued in similar duties.

Three recent graduates or students near graduation pre-tested the questionnaire. It took 1–2 hours to answer the compulsory part and 0.5–1 hour the voluntary part. It was not separated how much time it took to answer only Question 20. Some minor changes were made after the pre-testing. For example, one question was changed so that it should take considerably less time to answer.

9.1.2 How were data gathered?

The questionnaire was distributed in 2004 to all students specializing in Software Systems at the institution who were in process of graduation. The students belonged to the Degree Program of Computer Science and Engineering at the institution. The sampling was targeted at only 10–20% of the total number of graduates of the degree program because the questionnaire was not distributed to the students of other specializations. However, these respondents were the most suitable group from the institution for this survey because the specialization in Software Systems was more oriented to software development than the other specializations of the program.

Answering was compulsory for most respondents. According to the degree regulations of the institution, each student had to give a presentation of his or her Master's thesis. Typically, the questionnaire was given to a respondent on paper when he reserved time for his thesis presentation from a secretary. In some cases, the paper questionnaire was mailed to the respondent after the thesis presentation or even after the graduation. The respondents answered unsupervised and mailed the answered questionnaires to the author of the present thesis. The questionnaire had the author's contact information for asking questions but the respondents did not ask anything. The questionnaire was distributed to 30 students and 25 of them answered. Thus, the response rate was 83%.

9.2 Results

First, the background information of the respondents is presented. Second, the results of the respondents' opinions on the importance of various subjects and skills are presented. Third, the results of the respondents' opinions on courses and Master's thesis project are presented.

9.2.1 Background information of respondents

All persons of the target group were male. Some other properties of the target group and of the respondents are presented in Table 15. The data were collected from the student register of the institution. As can be noticed, the differences were very slight. According to the Mann-Whitney test, the differences were statistically not significant ($p \geq .05$). Thus, the sample was representative relative to the target group.

Table 15. Means of some properties of target group and respondents.

Property	**Target group** ($N = 30$)	**Respondents** ($N = 25$)
Age in years	28.3	27.5
Credits (minimum 180.0)[a]	189.5	189.1
Overall grade point average (scale: 1–5)	3.3	3.4
Grade point average of specialization (scale: 1–5)	3.7	3.8
Grade of Master's thesis (scale: 1, 2, … , 5)	4.2	4.3

[a]At the institution, one credit equals 40 studying hours.

The respondents were asked their employer's name and their job title after graduation. The given employer's names were classified using the WWW pages of a company. The distribution of the employers was as follows ($N = 25$): software house 44%, software services and consulting company 24%, telecommunications company 20%, and other or not classified 12%. The distribution of job titles was as follows ($n = 20$): software developers 60%, experts 35%, and project managers 5% where the category "experts" means consultants, researchers, or software specialists.

The distribution of answers to the question "How many lines of code have you programmed?" was ($n = 20$): 11–50 thousand lines 15%, 51–100 thousand lines 50%, and 100 thousand lines or more 35%. This question included programming both as part of studies and outside the university, for example, during internships. In addition, the respondents were asked how many lines of code they programmed as part of the Master's thesis project. Nineteen out of 25 respondents programmed as part of the thesis project. The mean of these 19 programs was approximately 7,500 lines of code when the maximum was 1,000 and the minimum 23,000 lines of code. This was not asked from the respondents but the author of the present thesis estimated course-by-course that the respondents typically programmed 15–20 thousand lines of code in required course assignments; that is, during their studies before the Master's thesis. Thus, most respondents had considerably programming experience outside their studies, for example during the internships and part-time jobs.

9.2.2 Importance of various subjects and skills

In this subsection, the results of the importance of various subjects and skills are reported. The means from Question 20 are presented in Table 16. In the question, the respondents had to evaluate how important various subjects and skills were for their future work during the next 12 months after graduation. The results were divided into the four categories that were used in the question. Within each category, the rows are first ordered according to the mean and then according to the name of the subject or skill.

The mean across the range of items was 2.6 (N = 1,003). The values of T_1 of the Mann-Whitney test between the answers of a single item (N = 22…24) and the answers across the range of items (N = 1,003) are presented in the column Mann-Whitney. Two asterisks (**) indicate that the difference between the answers of a single item and the answers across the range of items was statistically significant ($p < .01$). The confidence level $1 - \alpha = .99$ was used to avoid Type I errors because the number of items was so large.

It can be noticed that some means are very high or low. In other words, for some questionnaire items the students had high agreement whether it was necessary or not.

Table 16. Importance of subjects and skills for respondents' work during 12 months after graduation. Means and results of Mann-Whitney test.
Scale: 1 = Not at all important, … , 4 = Very important.

Subject or skill	*M*	Mann-Whitney
Mathematics, physics, and theoretical computer science:		
Other areas of theoretical computer science (e.g., automata)	2.1	-3.1**
Discrete mathematics	1.7	-5.1**
Logic (in particular, propositional and predicate logic)	1.7	-5.1**
Mathematics for continuous systems	1.3	-7.4**
Physics	1.1	-7.8**
More technical or part of the operational system:		
Object-oriented programming	3.8	5.6**
Data structures and algorithms	3.6	4.8**
Software architectures	3.5	4.4**
Extensible Markup Language (XML) techniques	3.2	2.6**
Internet protocols	3.2	2.6**
Procedural programming	3.2	2.6
Concurrent programming	3.1	2.0
Database management systems	3.1	2.2
Script programming	3.1	1.9
Computer/data security	3.0	1.4
Implementing techniques of user interfaces	3.0	1.4
Operating systems	3.0	1.7
Implementing techniques of WWW systems	2.8	0.3
Distributed systems	2.6	-0.6
Systems programming	2.6	-0.5
Computer architecture	2.5	-1.0
Telecommunications techniques other than Internet protocols	2.5	-1.2
Compilers	2.3	-2.2
Functional programming	2.3	-1.9
Embedded systems	2.0	-3.5**
Real-time systems	2.0	-3.7**
Artificial intelligence and knowledge engineering	1.7	-5.1**
Logic programming	1.7	-5.0**
Computer graphics	1.5	-5.8**
Software engineering (different phases of life cycle):		
Design	3.9	6.5**
Implementation	3.7	5.3**
Requirements	3.3	2.8**
Testing	3.3	3.2**
Concept exploration	3.0	1.7
Approval	2.5	-1.1
Operation and maintenance	2.4	-1.5
Installation and checkout	2.3	-2.0
Packaging and delivery	1.8	-4.6**
Retirement	1.7	-5.2**
Software engineering (possible in several phases):		
Version and configuration management	3.5	3.9**
Documenting	3.2	2.5
Project management	3.1	1.9

Note. $N = 24$ expect $N = 22$ for "Math…," and $N = 23$ for "Logic prog." and "Embedded…"
**$p < .01$

9.2.3 Evaluation of courses

Next, it is reported how the respondents evaluated the required courses of the specialization. The means and sample sizes of the question "Give the following courses a general grade based on how well their contents fit as part of the specialization as a required course" are presented in Table 17. The rows are ordered first according to the mean and then according to the course name. The sample size of the course Project in Software Technology was small because it was a new course and an alternative for the course Capstone Project. The sample size of the course Human-Computer Interaction was small because this course was discontinued in approximately 1996 and most respondents studied the newer course User Interfaces.

It can be noticed that the respondents generally gave good grades to the courses. The mean of the course Discrete Mathematics was the lowest. However, this course was no longer required in the specialization during the academic year 2004–2005. Instead, it was required in the basic studies that were common for all students of the degree program.

Table 17. Means and sample sizes of the question "Give the following courses a general grade based on how well their contents fit as part of the specialization as required courses." Scale: 1 = Poor, … , 4 = Excellent.

Course name	***M***	***n***
Capstone Project[a]	3.7	22
Operating Systems	3.6	25
Compilers	3.5	23
Operating Systems Project	3.5	17
Design and Analysis of Algorithms	3.4	25
Database Management	3.3	24
Project in Software Technology[b]	3.3	3
Concurrent Programming	3.2	21
Introduction to Knowledge Engineering	3.2	25
Introduction to Software Engineering	3.1	24
Human-Computer Interaction	3.0	2
Telecommunications Architectures	3.0	23
Computer Networks	2.9	19
Logic in Computer Science: Foundations	2.8	25
User Interfaces	2.8	20
Discrete Mathematics	2.4	23

[a]The official English name is Individual Project but Capstone Project is used here because it is probably easier to understand for most readers.

[b]This is a capstone project as well but more technically oriented than Individual Project.

9.2.4 Evaluation of Master's thesis projects

The results of the evaluation of Master's thesis projects are presented in the present subsection. This subsection is not a needs assessment but these results

were included because Master's thesis was such a significant part of the advanced studies at the institution. As will be explained later in Section 15.1, Master's thesis was 29% of the extent of studies that were selected as the target area of the present thesis.

In Table 18 are presented the distribution of answers to the questions that concerned (a) how interesting the Master's thesis was, (b) how useful the thesis was, (c) how good was the respondent's motivation towards the thesis, (d) how satisfied a respondent was with the supervisor's work, and (e) how satisfied a respondent was with the instructor's work. In the question about the usefulness of the thesis, it was not specified if this meant usefulness to the student, to the company or other organization where the student worked, or usefulness in general. Each thesis project had to have a supervisor who was typically a professor of the institution. An instructor worked typically in the same company or other organization where the thesis work was actually conducted. Apparently, the instructor was often student's immediate superior as well. More than 90% of the respondents had both a supervisor and an instructor.

Based on the results of Table 18, the respondents were typically satisfied with the thesis project. However, four respondents did not evaluate the supervisor's work and it is possible that these respondents were dissatisfied. Four respondents did not evaluate the instructor's work. One of these four respondents did not have an instructor and three did not answer for some other reason.

Table 18. Distribution of answers to various questions concerning students' satisfaction in Master's thesis project ($N = 25$).

Question	Topic	Poor	Satis-factory	Good	Excel-lent	Did not answer
27	How interesting?	0	4	11	10	0
28	Usefulness	0	2	16	7	0
29	Student's motivation	1	3	16	5	0
33	Supervisor's work	0	1	14	6	4
34	Instructor's work	2	0	13	6	4
	Sum:	**3**	**10**	**70**	**34**	**8**

Question 32 was "What was the worst or the most difficult issue during the Master's thesis project? Could this problem be avoided somehow?" The respondents were also advised not to comment on his supervisor's or instructor's work in this question because there were questions on these topics later. Twenty-two respondents answered the question. Only one respondent answered that he had no problems. The most common problems in the other answers were the following:

- various problems during the literature survey (6 respondents)

- problems in time management (5 respondents). In the worst case, the thesis project took 18 months because the respondent was able to conduct it only in the evenings and weekends beside a normal, full-time day-job.
- problems in deciding the limitations of the thesis (3 respondents).

9.3 Evaluation

First, the content validity of the present survey is discussed. As was mentioned previously, content validity of 42 questionnaire items was seriously considered when the questionnaire was planned (see Appendix D). Based on the respondents' answers, this apparently succeeded well. It is somewhat difficult to ascertain whether the respondents had problems during the answering because the survey was unsupervised. However, out of 25 respondents, 19 (76%) answered all the items reported in Table 16, five (20%) left only one item unanswered, and one (4%) left the part reported in Table 16 completely unanswered. This indicates that most respondents did not have problems in understanding the question items. In addition, out of 25 respondents, 68% answered completely the voluntary part, 20% answered it partly, and 12% did not answer the voluntary part. Based on this and the good quality of the answers, most if not all respondents were well motivated to answer the survey. This was contrary to what was expected because the respondents were required or asked to answer several other questionnaires as part of the graduating process.

In addition, the background of the respondents is evaluated as a content validity question. Here, the purpose is to ascertain whether the respondents had enough suitable working experience to even be able to give reasonable answers to the latter part of Question 20 (Table 16):

- In Finland, engineering students are older when they graduate than in many other countries. One of the reasons is that many students work at least part-time during the terms. As mentioned previously, the mean age of the respondents was 27.5 years when they graduated. It would be more suitable for characterizing the respondents as Bachelors who were graduate students in a part-time Master's program and had typically from one to three years of full-time work experience when they answered the questionnaire. This characterization was not used in the beginning of this section because the institution did not offer a Bachelor degree. Anyhow, these properties made the respondents' answers more reliable because the answers were typically based on several months working experience in software development.
- In some institutions, a student may choose if he or she conducts a Master's thesis or studies advanced courses. At the institution, Master's thesis was required for all students and a principal, such as a company or a research institution, typically provided a topic for a thesis. A principal probably

thought carefully about the topic because normally the principal paid a fair salary to the student for the thesis project. It was not directly asked where the respondents conducted their theses or if they got a salary for the thesis project. However, it was asked what was respondent's employer at the time of answering and after the graduation. In most cases, the employer at the time of answering was apparently the same as the employer during the thesis project. Eighty-four percent of the respondents ($N = 25$) worked at the time of answering in companies. After the graduation, the proportion was 92% ($N = 25$). Based on the answers, two respondents conducted their thesis at a university and got a salary, and apparently three respondents conducted their thesis at least partly without a salary or in their free time. Thus, it is safe to conclude that majority of the respondents conducted their thesis in a company.

- Based on the answers to Question 25, many respondents continued after graduation with the same project as in the Master's thesis project. This probably helped them to evaluate what kind of subjects and skills would be necessary during the next twelve months after graduation.

Second, the external validity of the present survey is discussed:

- It was shown previously that the sample was representative relative to the target group. However, it is not claimed that the target group would be representative of graduates in other institutions.
- The problems on the generalization of the results are similar as with the software developers. Twenty percent of the respondents worked for telecommunication companies. This is less than in the case of the software developers but still, as a consequence, it is possible that the importance of the following subjects was emphasized: computer/data security, concurrent programming, distributed systems, Internet protocols, and telecommunications techniques other than Internet protocols.
- At the institution, Unix has been the dominant operating system in computer science education since 1986. Based on the answers to Question 41, most respondents preferred Unix in their work as well. This might increase the evaluated importance of the following subjects: operating systems, script programming, and systems programming.

The stability of the results over time is evaluated in exactly same manner as previously in Sections 7.3 and 8.3. The stability is probably modest, or satisfactory at best. As will be shown later in Section 12, the proportions of most common skill categories have increased during the past 15 years in job advertisements targeted at software developer positions. This gives a reason to suspect that similar respondents as in the present study might answer differently in the year 2015, for example.

The results of the present survey were not compared against the previous findings because the previous publications did not include suitable results, as was explained at the end of Section 2.5.3.

10 Delphi study targeted at software developers: Cognitive skills

In the present research, the goal has been to identify cognitive skills that are important for expert software developers' work. The present research was not a needs assessment because the purpose was to evaluate the level of difficulty of cognitive skills, not their usefulness. However, this section was included in the thesis as well because the results are strongly related to the software development phases design, implementation, and test. Based on the results of three previous sections, these three phases are—obviously—important for software developers' work.

The results were published earlier in the Informatics in Education journal (Surakka & Malmi, 2005a).

The structure of this section is the following. First, the details of the research method are presented. Second, the results are presented and analyzed. Third, the research is discussed.

10.1 Research method

The Delphi method was selected for the present research because the topic "cognitive skills" was rather vague. When the Delphi method was used, it was possible to ask refined questions on this topic during the second questionnaire round.

The respondents were the same as explained previously in Section 7 where some of the results of the first questionnaire round were reported. The second questionnaire round was targeted at refining the results of an interesting part of the first questionnaire; that is, cognitive skills. The first questionnaire had three open questions about cognitive skills required by a software specialist. Based on the answers in total 36 different skills were identified. In the second round, the respondents defined the level of these skills; that is, how much learning and experience is needed before such a skill is mastered. The questionnaires are available on the web page of the institution (Surakka, 2005b).

The decision of limiting the second questionnaire to only one area of interest was based on several reasons: (a) The results from the other areas of the first questionnaire were sufficiently satisfactory. Thus, the need to conduct a second questionnaire round for the sake of the other areas was slight, (b) The respondents thought that the questions about cognitive skills were the most difficult to answer. The author of the present thesis interpreted this as a hint to further explore this area, (c) Regardless of the answering difficulties, some respondents thought that cognitive skills were an interesting or promising area

for research of this kind. This was the author's own opinion as well, and finally, (d) At the beginning of the research, it was promised to the respondents that participating would take no more than 1–3 hours.

After the cognitive skills were chosen as the topic for the second questionnaire round, the goal was to evaluate how demanding or difficult the different cognitive skills that were mentioned during the first round are.

10.1.1 Questionnaire rounds

The second questionnaire was answered between January and February 2004. The author of the present thesis was not present during answering on the second round. The mean answering time for the second round was 54 minutes. The main properties of the questionnaires are presented in the following two subsections.

First questionnaire

Only the questions about cognitive skills and problem-solving techniques of the first questionnaire are presented in the present subsection. Instead of cognitive skills, the concept "tacit knowledge" was used because it was assumed that it would be easier to understand for the respondents. An explanation of the concept including initial division into cognitive skills and technical skills was given before the questions. Three questions were:

- "What are important mental models, beliefs, and understanding for a top-level software developer that belong to the cognitive element of tacit knowledge?"
- "What topics and skills belong to the technical element of tacit knowledge for a top-level software developer? These can also be called skills that are located in the fingertips."
- "Do you believe that some area of tacit knowledge will be more important in the future?"

Second questionnaire

The second questionnaire was based on the respondents' answers to the first questionnaire. These were analyzed to identify and separate different skills mentioned in the comments. Comments clearly denoting the same skill were combined. Typing skill was included in the list, based on the author's observations, even though the respondents did not mention it. Finally, the list for the second questionnaire round had 36 comments each identifying at least one skill. In the second questionnaire, the respondents had to evaluate the level of these comments according to the following categories:

1. Very low-level skill that even novices can learn quickly (during a 1–4 credits basic course)
2. Somewhat low-level skill that requires working experience of 3–6 months to be learned, for example
3. Somewhat high-level skill that starts to differentiate good programmers from less good programmers
4. Very high-level skill that takes usually several years to learn and typically only top-level programmers have this skill.

In the question about problem-solving techniques, the respondents were asked to read or browse three pages of the book (Détienne, 2002, pp. 26–28). The text was about a strategy-centered approach to software design. After this, the respondent had to answer Question 3 that contained the following questions:

- "During the designing of software, have you used these techniques or strategies (Top-down vs. Bottom-up, Forward vs. Backward Development, Breadth-first vs. Depth-first, Procedural vs. Declarative, Mental Simulation)? How often? Do you think they are good?"
- "Has the use of these techniques or strategies changed when you have gained more experience in software development (as is described in the book)?"
- "Do you think these skills are tacit or explicit knowledge?"
- "What do you think the level of these skills is (the scale is the same as previously: 1 = Very low-level skill, …, 4 = Very high-level skill)?"

The second questionnaire had questions about typing skills and the use of an editor as well. These questions are not presented here because they are so simple that it is sufficient to just present the results.

10.2 Results

The results on cognitive skills, problem solving, and typing skills are presented in this subsection. The background information of the respondents was presented previously in Section 7.

10.2.1 Respondents opinions on cognitive skills

This subsection presents the results of the respondents' opinions on cognitive skills. In the second questionnaire, the statements of skills were divided according to the division used in the first questionnaire. However, for the present research the results were reclassified into two categories: composition and comprehension. Some comments were combined as well. Two comments are not presented in the tables because they are not related only to software development. These two comments and their means were Being systematic

(2.1) and Ability to type using ten fingers (2.1). Thus, Table 19 and Table 20 contain together only 30 (17 and 13, respectively) items whereas the second questionnaire contained 36 items. First, the results related to composition are presented in Table 19. The comments are ordered first according to the means and then according to the comments. The numbers in the leftmost column are used for commenting on the items.

Even though statistical analysis was not the main purpose of the present research, the Mann-Whitney test was used to test whether the observed differences were significant or not. The mean across the range of items was 2.8 ($N = 219$). The values of T_1 of the Mann-Whitney test between the answers of a single item ($N = 9$, 10, or 20) and the answers across the range of items ($N = 219$) are presented in the column Mann-Whitney. The sample size of Item 7b was 20 because the data of two questionnaire items were pooled. Two asterisks (**) indicate that the difference between the answers of a single item and the answers across the range of items was statistically significant ($p < .01$). The confidence level $1 - \alpha = .99$ was used to avoid Type I errors and because the same confidence level was used in the other research of the present thesis.

There are a few observations that need to be commented on in Table 19. First, the high mean of item "2a Automating one's own work using scripts, keyboard macros, etc." obviously does not indicate the time needed to learn such skills. Instead, it indicates the time needed to use them efficiently as one's personal tools, when necessary. It is author's assumption that this skill is analogous to bottom-up software design, where the programmer recognizes the need for general-purpose procedures and data structures. Thus, it has a role in differentiating excellent developers from others. Second, the items "Design of interfaces" and "Isolating the implementation behind well-defined (and documented) interfaces" are kept separate. The first one is more associated with designing and the latter with using interfaces. It is obviously easier to learn to use ready-made interfaces properly than to actually design interfaces that support good software architecture. Third, comments 2b and 7b are similar but one could consider that 2b is broader than 7b. Comment 2b includes also low-level knowledge, for example knowing language's keywords by heart. Fourth, one could consider that the low ranked items 15a and 17 are not really cognitive skills, but other kind skills or knowledge. However, these items were not omitted from the table because they were related to composition.

Table 19. Comments classified into the category "Composition": Means and results of Mann-Whitney test to the question "What do you think the level of this skill is?" Scale: 1 = Very low-level skill, …, 4 = Very high-level skill.

Number	Comment	*M*	Mann-Whitney
1	A good programmer has always a model. The code itself comes from the spine and brains only operate the model.	3.6	3.1**
2a	Automating one's own work using scripts, keyboard macros, etc.	3.5	2.9**
2b	The mastery of a certain programming language or a certain environment	3.5	3.0**
4	Writing code so well that it is not even necessary to comment	3.4	2.5
5	Design of interfaces	3.3	2.1
6	Choosing as optimal data structures and algorithms as possible	3.1	1.2
7a	Ability to find right abstractions	3.0	0.8
7b	Mastery of the structures and idioms that are characteristic of each language or environment	3.0	1.1
9	Ability to write code clearly and briefly	2.9	0.5
10a	Choice of the programming language	2.8	-0.1
10b	Implementing programs as independent of the operating environment as possible	2.8	-0.1
12	Isolating the implementation behind well-defined (and documented) interfaces	2.7	-0.6
13	Changing lower level cognitive models/design patterns to code. For example, table field in C/C++ object and its memory management get/set/constr/destr	2.6	-1.1
14	Identifying concepts	2.4	-1.5
15a	Ability to find existing Open Source solutions from the Net and being familiar with libraries	2.3	-2.2
15b	Procedural or object-oriented way of thinking about programming	2.3	-2.2
17	Documenting code	1.9	-3.3**

Note. $n = 9$ or $n = 10$ expect $n = 20$ for Item 7b.
**$p < .01$.

The results related to the category Comprehension are presented in Table 20. The mean across the range of items was 3.1 ($N = 138$). The values of T_1 of the Mann-Whitney test between the answers of a single item ($N = 9$, 10, or 20) and the answers across the range of items ($N = 138$) are presented in the column Mann-Whitney. The sample size of Item 8a was 20 because the data of two questionnaire items were pooled. Two asterisks (**) indicate that the difference between the answers to a single item and the answers across the range of items was statistically significant ($p < .01$). The confidence level $1 - \alpha = .99$ was used to avoid Type I errors and because the same confidence level was used in the other research of the present thesis.

As a general note, it is interesting that the respondents have often used words like "see" and "notice" to describe these skills. One might consider that item "13 Understanding the function of programming languages and computer (e.g., parameter passing, the order of execution, and concurrency)" is explicit rather than tacit knowledge.

Table 20. Comments classified into the category "Comprehension": Means and results of Mann-Whitney test to the question "What do you think the level of this skill is?" Scale: 1 = Very low-level skill, …, 4 = Very high-level skill.

Number	Comment	*M*	Mann-Whitney
1	Ability to see all possible alternatives from the source code (this comment was related to debugging)	3.9	3.6**
2	Ability to notice isomorphisms with some known problem	3.6	2.1
3	Ability to evaluate how the system will operate even before its implementation has begun	3.5	1.4
4a	Ability to see esthetic values in solutions	3.4	1.2
4b	Ability to see the big picture. What is the core of the problem and how it is connected to the environment around it?	3.4	0.9
6a	Ability to distinguish essential matters	3.2	0.0
6b	Interpreting the program as the whole	3.2	-0.1
8a	Ability to change fluently • abstraction level (e.g., single line of code vs. procedure or big picture vs. details) • perspective (e.g., is the control flow or the data flow of the program examined) • concepts (e.g., are the concepts of program or the concepts of application domain considered) • view (e.g., users needs vs. maintenance vs. development speed)	3.1	-1.1
8b	Ability to debug	3.1	-0.7
10	Ability to see symmetries	3.0	-0.9
11	Exploring the architecture of the existing systems	2.9	-1.2
12	Ability to see a big problem as several partial problems	2.7	-2.2
13	Understanding the functioning of programming languages and computer (e.g., parameter passing, the order of execution, and concurrency)	1.8	-4.6**

Note. $n = 9$ or $n = 10$ expect $n = 20$ for Item 8b.
**$p < .01$.

10.2.2 Respondents opinions on problem solving

The means to the question where the difficulty level of different development strategies was asked are presented in Table 21. The strategies are presented in Appendix B. Some respondents did not answer this question or did not give the level for all strategies. Some respondents gave different values for single

strategies, for example, one respondent gave 2 for top-down and 3 for bottom-up. In this case, the mean 2.5 was used for the strategy "Top-down vs. bottom-up." It can be noticed that the differences between the means are small. No statistical tests were conducted to analyze the results because the differences and the subsamples were so small.

Table 21. Sample sizes and means to the question "What do you think the level of these skills is?" Scale: 1 = Very low-level skill, …, 4 = Very high-level skill.

Strategy	***n***	***M***
Breadth-first vs. depth-first	6	3.3
Procedural vs. declarative	6	3.3
Top-down vs. bottom-up	8	3.1
Forward vs. backward	4	3.1
Mental simulation	4	2.9

The question had other parts as well. These answers were so mixed that no statistics are presented. Most respondents commented that they have used all or most of the strategies and they have often used a combination of strategies, for example, "top-down + verification by mental bottom-up simulation" or "bottom-up + declarative." Some respondents commented that single strategies are explicit knowledge but choosing a suitable strategy and changing fluently between the strategies is tacit knowledge that takes years to achieve. One respondent commented that the list of strategies did not mention iterative technique and another that "design by aesthetics" was missing. He explained that design by aesthetics is a kind of extreme bottom-up situation where first, central parts are programmed as the developer wants them to be, and then, it is considered what has to be done so it is really possible to use these central parts as they were coded.

10.2.3 Typing skills and use of editor

To get some data on the lower level practical skills of the respondents, the questionnaire included a few questions on their typing skills and the editors they use in practical work. Four (40%) respondents have taken a typing course. Five (50%) respondents could type with ten fingers, four (40%) used less than ten fingers but did not look at the keyboard during programming, and only one (10%) respondent had to look at the keyboard while typing.

The author wanted to compare the previous results against some other group. However, no records or previous publications were found on how common typing skills were among general population or among computer science graduates in particular. Therefore, similar question was asked from the students of a basic programming course. The questions were presented as part of the normal feedback questionnaire at the end of the course. Thirteen percent of the students ($N = 216$) had taken a typing course and 25% could type using

ten fingers. Eighteen percent answered that he or she did not have to look at the keyboard during one minute of constant typing, 47% had to look 1–5 times, and 33% more than five times.

According to the *z*-test for proportions (Milton & Arnold, 2003, p. 324), the difference between the experienced software developers and the students is statistically not significant ($p \geq .05$) for the proportions of ability to type using ten fingers (50% and 25%, respectively). However, one could assume that looking at the keyboard was more important than using ten fingers because looking at the keyboard might interrupt thinking. The difference would be statistically very significant ($p < .001$) if the answers were interpreted so that only 10% of experienced software developers had to look at the keyboard and the corresponding proportion for the students was 82%. Thus, there was some evidence that the typing skills of the experienced software developers were better than those of the students—as one would expect.

An editor is the basic tool in programming. Good knowledge of the versatile features allowed by advanced editors can significantly improve the coding speed. Based on the background of the respondents, it was deduced (even though this was not asked) that most if not all had used mainly Emacs in a Unix environment in their student days. According to the answers, most respondents have also continued to use Emacs after graduation, which is easily understandable considering its wide variety of available operations and support for editing different languages. Six respondents used mainly Emacs, one respondent used Epsilon, which is an Emacs clone, and one respondent used vi. One respondent had changed from Emacs to Source-Navigator because he thought that Source-Navigator was more suitable for editing and browsing large programs. Five respondents answered that they had programmed macros for Emacs, and two answered that they knew the basic commands of Emacs but had not programmed macros. At the time of answering, one respondent programmed mainly in a Windows environment and used Visual Studio. Two respondents answered that they did not work in a Windows environment. Others answered that they have installed Emacs in Windows when necessary.

10.3 Evaluation

First, the present research is discussed at a general level before validity and reliability are considered:

- The present research would have been very different if the original main goal was to gather information about the cognitive skills of software developers. Questionnaires are seldom used in the psychology of programming whereas experimental research setting is dominant. One source of criticism is that questionnaires measure opinions, not observable behavior. However, in the

present research the purpose was to measure especially the opinions of experts.

- The Delphi method was suitable for this type of research because the follow-up questionnaire round was necessary to investigate more explicitly tacit knowledge, which is a vague concept. During the first questionnaire round, most respondents commented that the questions about the tacit knowledge were the most difficult to answer. A possible interpretation could be that the research method used was not suitable or that the questions were poorly designed. However, the author interpreted that the answering difficulties were mainly due to the topic itself; that is, the topic is genuinely difficult.
- An alternative respondent group could be the researchers in the area of psychology of programming. The results might be very different from those of the present research.

Second, the content validity of the present research is discussed. It is possible that the respondents do not remember or cannot describe skills that have already been automated for several years. For example, adults often have difficulties in describing how a bicycle is ridden or a car is driven. An attempt was made to minimize this problem by dividing the questions into two parts and adding an explanatory text before the questions.

The stability of the results over time is likely to be good, or at least satisfactory, because they mostly represent more abstract concepts or skills than typically mentioned in job advertisements, for example.

The results of the present study are not compared with the previous publications because no similar research was found, as explained previously, in Section 2.8.

11 Triangulation of questionnaires

The results of the three questionnaires targeted at the software developers, the professors and lecturers, and the Master's students are compared in this section. Finally, the main findings of Part III are listed.

11.1 Results of three questionnaires

The means across the range of all items from the three respondent groups were: the software developers 2.9 ($N = 460$), the professors and lecturers 3.0 ($N = 787$), and the students 2.6 ($N = 1{,}003$). According to the Mann-Whitney test, all differences between the groups were statistically significant ($p < .01$). However, only the differences between the students and the two other groups were large enough to be of some practical relevance. That is, generally the students evaluated the questionnaire items as being a little less important than the other two groups.

The distributions of the three groups are presented in Figure 6. Also these results were across the range of all items. The software developers,' and the professors and lecturers' distributions were similar when the most common answer was "3 = Somewhat important." In all three distributions, the answers "3 = Somewhat important" and "4 = Very important" were more common than the answers "1 = Not at all important" and "2 = A little important." The students' distribution was more even, and they answered "1 = Not at all important" considerably more often than the software developers, and the professors and lecturers did.

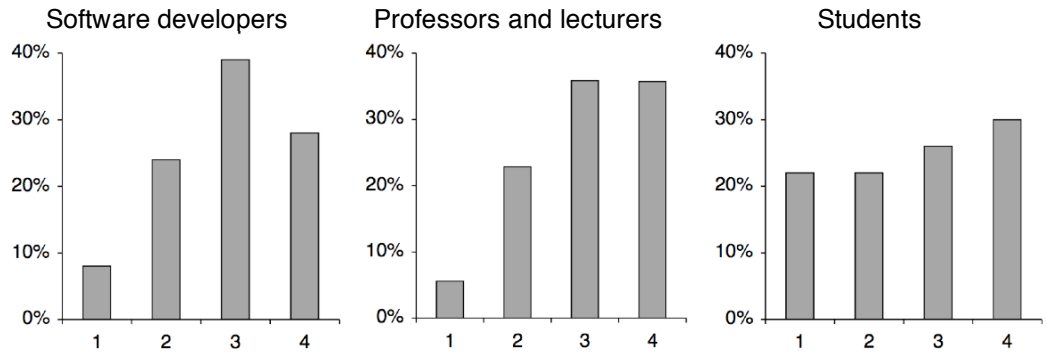

Figure 6. Distributions of answers across the range of all items from the three respondent groups. Scale: 1 = Not at all important, … , 4 = Very important.

The means of the individual subjects and skills are presented in Table 22. The column title "Developers" refers to the software developers and "Professors" to the professors and lecturers. Inside each category, the rows are ordered first

according to the software developers' results, then according to the professors and lecturers' results, and finally according to the students' results. On each row, a letter pair indicates that the difference between the groups in the question was statistically significant ($p < .01$) according to the Mann-Whitney test. For example, two letters "b" in the row "Other areas of theoretical CS (e.g., automata)" indicate that the difference between the software developers and the students was statistically significant.

Based on the results, the following subjects or skills were evaluated as being important by all three respondent groups. The means of these items were at least 3.0 for all three groups: computer/data security, concurrent programming, data structures and algorithms, documenting, object-oriented programming, operating systems, procedural programming, project management, software architectures, and version and configuration management. In addition, the following software development phases were evaluated as being important: concept exploration, requirements, design, implementation, and test.

Mathematics for continuous systems and physics were evaluated as being less important. These were only two questionnaire items where the means of all respondent groups were 2.0 or less.

Table 22. Importance of various subjects and skills. Means of three respondent groups. Scale: 1 = Not at all important, … , 4 = Very important.

Subject or skill	Developers (N = 10, 11)	Professors (N = 18, 19)	Students (N = 22…24)
Mathematics, physics, and theoretical CS:			
Other areas of theoretical CS (e.g., automata)	3.3[b]	2.9[c]	2.1[b,c]
Logic (in particular, propositional and predicate l.)	2.8[b]	2.9[c]	1.7[b,c]
Discrete mathematics	2.6[b]	3.1[c]	1.7[b,c]
Mathematics for continuous systems	2.0[b]	1.7	1.3[b]
Physics	1.6[b]	1.5	1.1[b]
More technical or part of operational system:			
Data structures and algorithms	3.8	3.9	3.6
Procedural programming	3.8	3.7	3.2
Object-oriented programming	3.6	3.9	3.8
Software architectures	3.5	3.6	3.5
Script programming	3.4	2.8	3.1
Internet protocols	3.4[a]	2.3[a,c]	3.2[c]
Operating systems	3.3	3.7	3.0
Computer/data security	3.2	3.0	3.0
Systems programming	3.2	2.9	2.6
Concurrent programming	3.1	3.5	3.1
Distributed systems	3.1	3.5[c]	2.6[c]
Compilers	3.1	3.1[c]	2.3[c]
Computer architecture	3.0	3.2	2.5
Database management systems	2.7	3.3	3.1
Implementing techniques of user interfaces	2.7	3.1	3.0
Implementing techniques of WWW systems	2.7	2.8	2.8
Extensible Markup Language (XML) techniques	2.7	2.5[c]	3.2[c]
Functional programming	2.6	2.9	2.3
Real-time systems	2.6	2.7	2.0
Embedded systems	2.5	2.7	2.0
Logic programming	2.3	2.6[c]	1.7[c]
Telecommunications tech. other than Internet prot.	2.0	2.0	2.5
Computer graphics	1.9	2.3[c]	1.5[c]
Artificial intelligence and knowledge engineering	1.6[a]	2.5[a,c]	1.7[c]
Software eng. (different phases of life cycle):			
Implementation	3.7	3.9	3.7
Design	3.7	3.8	3.9
Requirements	3.6	3.4	3.3
Test	3.5	3.8	3.3
Concept exploration	3.0	3.4	3.0
Approval	2.6	2.9	2.5
Operation and maintenance	2.5	2.9	2.4
Installation and checkout	2.3	2.8	2.3
Packaging and delivery	1.9	2.3	1.8
Retirement	1.8	2.3	1.7
Software eng. (possible in several phases):			
Version and configuration management	3.6	3.6	3.5
Project management	3.2	3.6	3.1
Documenting	3.0	3.4	3.2

[a,b,c]On each row, the letter pairs indicate that the difference of the corresponding ranks (not means) is statistically significant ($p < .01$) according to the Mann-Whitney test.

11.2 Professors and lecturers' explanations for differences

Next, explanations for the differences in Table 22 are considered. These explanations were asked from the professors and lecturers who took part in the Delphi study. The software developers and the students were not asked for explanations for practical reasons. The Delphi study targeted at the professors and lecturers was conducted last and, therefore, it was convenient to ask for explanations from them.

During the second questionnaire round, the respondents were asked explanations for some of these differences. Four questions were asked and 17 respondents answered them. Question 5 was: "Why the students evaluated discrete mathematics, logic, and theoretical computer science as being less important than the professors and lecturers, and the software developers?" Typical answers were as follows:

- "These topics are abstract and difficult. Students do not consider these topics as concrete tools in work."
- "Students do not understand the relationship between theory and practice."
- "Professors and lecturers, and apparently software developers as well have a longer perspective. There are some situations where the problems can be solved considerably better using formal methods than using traditional, even *ad hoc* programming."

Question 6 was: "Why the professors and lecturers evaluated artificial intelligence and knowledge engineering as being more important than the software developers and the students?" Typical answers were as follows:

- "This is a good research area from the viewpoint of an educator. Students have not had a chance to apply these in 'real life.'"
- "Professors and lecturers believe that artificial intelligence has prospects in the future. Software developers and students think of what is important now."

Question 7 was: "Why the professors and lecturers evaluated Internet protocols as being less important than the software developers and the students?" Typical answers were as follows:

- "Professors and lecturers do not consider it so important because the Internet is just one telecommunications technique."
- "Students consider knowledge of the Internet as important because it improves their likelihood of getting work."
- "These are important at the moment in basic software development, too."

11.3 Other explanations for differences

In addition to these explanations from the professors and lecturers, the background of the students and the software developers partly explained the differences in the item "Internet protocols." As mentioned previously, approximately 45% of the software developers and 20% of the students worked for telecommunication companies.

11.4 Correlation of results

Next, it is calculated whether the results of the three respondent groups correlate with each other. As an example, the means of the software developers, and the professors and lecturers that were presented previously in Table 22 are presented in Figure 7. It can be noticed from the figure that generally the means seem to correlate quite well. However, there are two points where the mean of the professors and lecturers is clearly different from the mean of the software developers. These points are circled and marked with letters A and B, respectively. The leftmost differing point A refers to the item "Artificial intelligence and knowledge engineering," for which the means were 2.5 for the professors and lecturers and 1.6 for the software developers. The differing point B at the right side refers to the item "Internet protocols," for which the means were 2.3 for the professors and lecturers and 3.4 for the software developers. As was previously reported in Table 22, these differences were also statistically significant ($p < .01$).

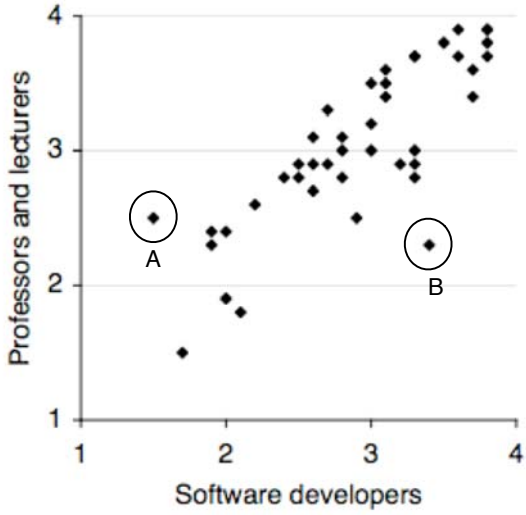

Figure 7. Means of software developers, and professors and lecturers.
Scale: 1 = Not at all important, … , 4 = Very important.

The Spearman rank correlation coefficients r_s between the groups were as follows: the software developers versus the professors and lecturers 0.80, the software developers versus the students 0.86, and the professors and lecturers versus the students 0.74. All these coefficients indicated that the means correlated positively. The greatest correlation was between the means of the software developers and the students. In addition, it was calculated whether the correlations were statistically significant. From three options, the upper-tailed test for positive correlation was selected. For all three correlations, the confidence level used was $1 - \alpha = .999$ and the sample size was 42. The value of $w_{0.999}$ was 0.58 for all three correlations (Conover, 1999, p. 317). This was smaller than the corresponding Spearman rank correlation coefficients. Thus, the positive correlations were statistically very significant ($p < .001$).

11.5 Main findings of Part III

The main findings of Part III are:

- From various subjects and skills, computer/data security, concurrent programming, data structures and algorithms, documenting, object-oriented programming, operating systems, procedural programming, project management, software architectures, version and configuration management were evaluated as being important.
- From various software development phases, concept exploration, requirements, design, implementation, and test were evaluated as being important. The results concerning cognitive skills corresponded well with this finding because most cognitive skills were related to these phases.
- Mathematics for continuous systems and physics were evaluated as being less important.

Part IV: Job advertisement analyses

The results of two job advertisement analyses are presented in this part. The purpose of these analyses was to solve the main problem and several subproblems of the present thesis. These problems are listed later at the beginning of Sections 12 and 13.

As explained previously in Section 3.2, job advertisement analyses were used in the present thesis instead of asking this question from managers and directors who hire software developers. The first research is a trend analysis from the year 1990 to the year 2004 and the second research is a more detailed cross-sectional analysis of the year 2004.

The scope of this part of the thesis is presented in Figure 8 using the data sources and research methods. The figure is the same as Figure 2 presented previously in Section 3.2 but the boxes related to this part are gray.

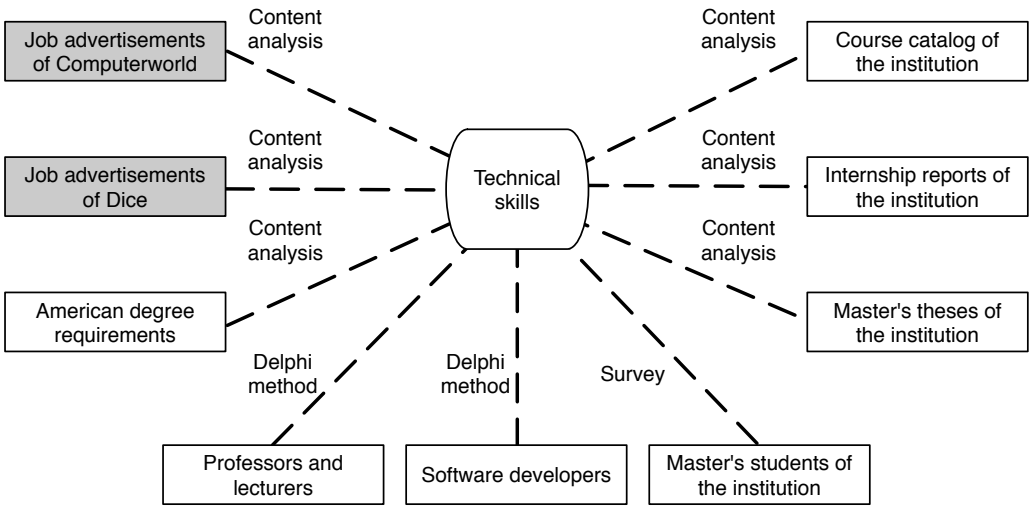

Figure 8. Scope of Part IV of thesis.

12 Trend analysis of job advertisements

The purpose of the present trend analysis was to solve the following subproblems that were presented previously in Section 1.2:

- Has the number of required technical skills increased during the past 15 years in job advertisements targeted at software developers? Todd et al. (1995) reported that the number of technical phrases in job advertisements for programmer positions increased from the mean of 2.2 in 1970 to 4.2 in 1990. Has this increase continued after the year 1990?
- In particular, how has the number of required distributed technology skills increased? World Wide Web technology was released in 1993. After this, the number of web sites has increased rapidly. As a consequence, skills related to distributed systems should now be required more often than ten years ago.

The details of the research method are described in Section 12.1 and the results are presented in Section 12.2. For brevity, this section presents only some general results. The results of this section were published earlier in the Informatics in Education journal (Surakka, 2005a, pp. 102–110). More detailed results about individual technical skills such as the most common programming languages can be found in Surakka (2005c).

12.1 Research method

Trend analysis and content analysis were selected as the methods of the present research because they were obvious choices. Other methods were not even considered. Trend analysis is highly suitable for solving the subproblems presented above. Content analysis was an obvious choice when job advertisements were used as the data source; as mentioned previously, content analysis is also known as document analysis. In this subsection, the details of the research method are explained.

12.1.1 Choice of data source

In previous longitudinal research, newspapers or professional magazines were used as data sources but web recruiting services were not used. The number of newspapers and magazines varied from two to ten. The justification for several data sources was to reduce the effect of possible regional differences. Use of newspapers was more common than the use of professional magazines. Gallivan et al. (2004) was the only previous research that used a professional magazine (Computerworld) as a data source. The commercial analysis (Salary

Services, 2004a, p. 286) used three professional magazines, six newspapers, and six web recruiting services.

In the present research, web recruiting services were not selected because they had not operated long enough for the purpose of the present research and data from the past few years were not publicly available. This was a very problematic situation because according to Salary Services (2004a, p. 2), the web services have dominated the IT job advertising market at least since the year 2000 and therefore, it was not certain if newspapers and professional magazines were still representative data sources. However, it was the author's opinion that (a) it was better to conduct a trend analysis using a magazine or a newspaper than not to conduct a trend analysis at all because data from web services were not available, and (b) the selected data source was the most suitable that was available.

Only one data source was used to keep the amount of work to a reasonable level. Newspapers such as the New York Times were not selected because they were too regional. From various magazines targeted at IT professionals, Computerworld was chosen. Other possible professional magazines would be, for example, Communications of the ACM, IEEE Computer, and IEEE Software. The circulations of these four magazines were approximately as follows (Bowker, 2004): Computerworld 250,000, IEEE Computer 97,000, Communications of the ACM 85,000, and IEEE Software 23,000. Computerworld was published weekly and the other three magazines six or 12 times per year. The main reasons to choose Computerworld were that it is a national magazine and might be more attractive to advertisers because it has the biggest circulation and is published weekly.

12.1.2 Sampling

In the previous longitudinal analyses, the periods varied from six to 20 years and the intervals from one to seven years. Athey and Plotnicki (1998) and Gallivan et al. (2004) used unequal intervals whereas the others used equal intervals. In the present research, the year 1990 was chosen as the starting year because WWW technology was released in 1993 and it was considered as interesting to get some results before that year. Every second year was chosen as the interval to get more detailed results from possible trends. Every fifth year was not used because it was assumed that changes in distributed technologies, in particular, might be so fast and so great during the period that it would be beneficial to use a shorter interval.

Gallivan et al. (2004) and Maier et al. (1998) collected data from four issues per year. Todd et al. (1995) collected data from each month of the year in order to avoid seasonal or cyclical effects on data. However, in these three papers, it was not reported if there were any seasonal or cyclical effects. Athey and Plotnicki (1998) and Trower (1995) collected data from one issue per year.

In the present research, one issue per year was chosen as the sampling strategy. This decision was based on an assumption that there were no significant seasonal or cyclical differences. Normally, the sampling was issue number 36. However, in 2002 and 2004 the sampling was three issues (36–38) to get sufficiently large subsamples from each year. The limit was set to at least 100 positions per year.

12.1.3 Coding

The selected 12 issues had a total of 1,004 job advertisements that were read and coded manually. It is possible that one advertisement contained several job titles and one job title contained several positions. In most previous longitudinal job advertisement analyses, apparently one advertisement was used as the unit of analysis. Gallivan et al. (2004, p. 71) and Salary Services (2004a, p. 285) were the only previous analyses that reported that they used one position as the unit of analysis. In the present research, one position was used as the unit of analysis.

Gallivan et al. (2004, p. 86) wrote: "We initially coded ads in both the regional and national *Computerworld* editions; however, based on additional information provided to us by the *Computerworld*'s Director of Classified Advertising, we combined the data sets." In the present research, the advertisements were not separated by regions.

An advertisement was included if it contained at least one suitable job title such as "Programmer," "Programmer Analyst," "Software developer," or "Software engineer." For example, the job titles "C++ developer" and "Web developer" were classified as software developer positions and were included. A systems analyst or programmer position was included if software development tasks were mentioned as well. A systems analyst/programmer position was excluded if it was rather a systems administrator position or the proportion of software development tasks was unclear. The following job titles, for example, were always excluded: "Business analyst," "Consultant," "Database administrator," "Project manager," "Quality assurance engineer," and "Systems administrator." A position was excluded if no job title was given or if it was too general, for example "IT professional," the required degree was a Doctoral degree, or the field of study was not suitable for a computer science graduate, for example, Masters in Electrical Engineering was required. In other words, the position should be suitable for a Bachelor's or Master's degree graduate from a computer science program. Contractor positions were included and part-time positions excluded.

Each suitable job title described one or more open positions. The exact number of positions was used if it was given, for example, "5 programmers." The job title was coded as two positions if the number of positions was not given but the text indicated several positions, for example, "Programmers." It

was coded as one position if the text indicated one position, for example, "Programmer." Overall, these job advertisements contained 1,291 suitable positions, which is the sample of the present research (i.e., $N = 1{,}291$). The number of suitable positions for each year varied from 112 to 265. These are the subsamples of the present research (i.e., $n = 112, \ldots, 265$). The numbers of included advertisements, excluded advertisements, and excluded positions were not counted.

Technical skills such as Cobol, Java, SQL, and Windows were sought from advertisements of these suitable positions. These phrases were typically names or abbreviations of different programming languages, operating systems, database vendors, and protocols. In some advertisements, the required and desired skills were separated using phrases like "must have the following skills," "is required," "is preferred," and "is desirable." Only required skills were included, and desired skills were excluded. A skill was coded as required if it was not stated if it was required or only desired.

12.1.4 Statistical analysis

A proportion was counted for each individual skill for each year. For example, the sample of the year 1990 had 189 positions and Cobol was required in 77 of these positions. Thus, the proportion of Cobol was 41%. In addition to these proportions for individual skills, several other figures were counted. These coding principles are explained later before the corresponding results.

The Student's t test and the Smith-Satterthwaite procedure were used to test if the difference between two means was statistically significant. The z-test for proportions was used to test if the difference between two proportions was statistically significant.

In the previous longitudinal analyses, Gallivan et al. (2004, p. 72) was the only one that reported use of statistical tests. In the other analyses, statistical tests were not used or at least were not reported. However, even Gallivan et al. used a statistical test only to make sure that they could combine two datasets into a single dataset, but they did not analyze the results using statistical tests.

12.2 Results

The results of the number of required technical skills and five main skill categories are presented in this subsection.

12.2.1 Number of required technical skills

The number of required individual technical skills was counted for each position. For example, if Cobol and DB2 were required, the number of skills

would be two. The minimum number was used if alternatives were given. For example, for the text "C++ or Java" the number of skills was one. The mean value of these numbers was counted for each year. The means of three categories All, Programmers, and Others are presented in Figure 9. The software developer, software engineer, and systems analyst positions were combined as the category Others because some of the subsamples were too small to be presented alone. It can be noticed that the means increased during the period. For example, the mean of the category All increased from 3.6 in 1990 to 7.7 in 2004. Based on the Student's *t* test and the Smith-Satterthwaite procedure, the differences between the means of the years 1990 and 2004 are statistically very significant ($p < .001$) for the categories "All" and "Programmers," and significant ($p < .01$) for the sample "Others."

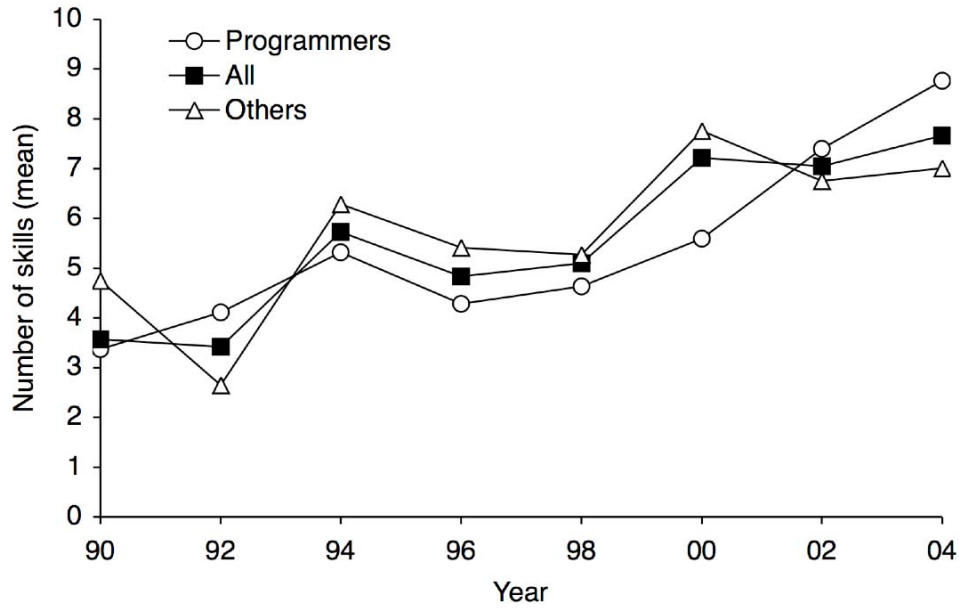

Figure 9. Means of number of required technical skills in the years 1990–2004.

12.2.2 Results for five main skill categories

During this part of analysis, the following criteria were used: (a) at least one common programming language skill (C, C++, Cobol, Java, or Visual Basic), (b) at least one operating systems skill such as AS/400 or Windows NT, (c) at least one database skill such as Oracle or SQL, (d) at least one networking skill such as LAN or TCP/IP, and (e) at least one distributed technology skill such as client/server or ASP. These five categories are called Programming language, Operating systems, Database, Networking, and Distributed technology. The

proportion was counted for each category for each year. For example, in the 1990 subsample were 189 positions and in 92 of these at least one common programming language was required. Thus, the proportion was 49%. Similar results for each year and for all five categories are presented in Figure 10.

It can be noticed that the proportions for every category increased. However, the increase of the category Networking was small and no trend could be noticed. Based on the z-test for proportions, the differences between the proportions of the years 1990 and 2004 are statistically not significant ($p \geq .05$) for the category Networking and very significant ($p < .001$) for the other categories.

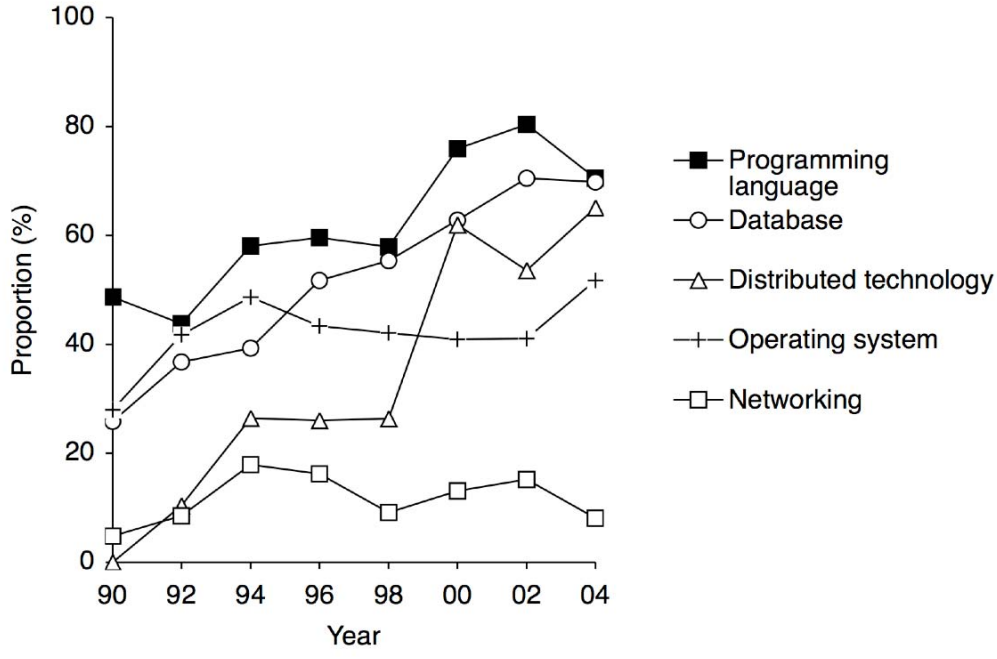

Figure 10. Proportions (%) of at least one skill in five skill categories in the years 1990–2004.

12.3 Evaluation

First, the content validity of the present research is discussed. One could find out from job advertisements that some skills were probably important, but determining whether a particular skill or subject was not important could be much more difficult, or impossible. This is discussed more in Section 13.3.

Second, the external validity of the present research is discussed. There are some problems with the representativeness of the sample:

- The use of web recruiting services increased strongly during the past five years and as a consequence, the proportion of newspaper advertisements decreased. For example, the author of the present thesis estimated that the web recruiting service Dice published approximately 0.3–0.8 million job advertisements and Computerworld approximately 3,000 job advertisements in 2004. In other words, currently the web recruiting services appear to dominate the IT job advertisement market in the USA. Thus, it is possible that Computerworld is not a representative source for the period 2000–2004. This problem is considered later in Section 14 when the results of the present research are compared with the results of the cross-sectional analysis. However, Computerworld should be a representative source for the period 1990–1998.
- Only one issue per year was chosen as the sampling strategy.
- It is possible that some well-known companies such as HP, IBM, Microsoft, and Sun Microsystems do not advertise in Computerworld at all or only a little because interested job seekers search job advertisements directly from the web site of the company. As will be reported later in Section 13.3, it was found that Microsoft announced only a very small proportion of open positions in Dice.

Stability over time is not a very relevant question for the present trend analysis because changes were expected. Based on the actual results of the present research, the stability of the results over time is likely to be modest, or satisfactory at best. Based on the results for the period 2000–2004 presented in Figure 9, it is not possible to conclude whether the number of required skills has stagnated or will probably increase in the future. Therefore, it is not possible to evaluate how stable these results are if the research were to be repeated later for the period 2005–2020, for example. The results presented in Figure 10 were for broader skill categories such as Database. The changes for such categories are likely to be slower and less random than for individual skills. In addition, the results for the category Networking are likely to be more stable because no trend could be noticed.

A big problem for the present research is the increased use of web recruiting services in general. Fortunately, this problem can be partly solved by comparing the results of the present research with the results of the previous job advertisement analyses. In particular, the report of Salary Services Ltd. (2004a) is useful because its sample was large and included recent data from several web recruiting services as well. This comparison is presented later in Section 21.1.2.

13 Cross-sectional content analysis of job advertisements

In this section, the cross-sectional content analysis of job advertisements is presented. The present research is a single cross-sectional analysis; that is, only one data source is considered at one point in time. The results were published previously in the Informatics in Education journal (Surakka, 2005a, pp. 110–119).

The purpose of the present research was to solve the main problem of the present thesis using job advertisement analysis. The main problem "What technical skills do graduates from specialization in Software Systems need in their work after graduation?" was changed to the following problem when it was assumed that these positions are typical for the graduates from specialization in Software Systems:

1. What technical skills were needed most in positions for programmers, software developers, and software engineers? In particular, distributed technology skills were analyzed thoroughly because, as a consequence of World Wide Web technology, these skills should now be required more often than they were ten years ago.

In addition, the purpose was to solve the following subproblems that were mentioned previously in Section 1.2:

2. What are the differences, if any, between the required skills of programmers, software engineers, and software developers?
3. What are the differences between entry-level and senior-level software developer positions?
4. How well do entry-level job requirements for software developers correspond with the requirements of a typical undergraduate program in computer science? This subproblem can be classified as a planning problem as well, rather than as a research problem.

13.1 Research method

Content analysis was an obvious choice as the method of the present research because job advertisements were used as the data source; as mentioned previously, content analysis is also known as document analysis. Next, the details of the research method are explained.

In the USA, web recruiting services were dominant in the information technology job advertising market in 2004. The biggest service was Dice (http://www.dice.com), which was selected for the present research because a

large number of advertisements were necessary for some parts of the analysis, for example, in order to find a large enough sample of entry-level positions.

The author of the present thesis used the service as a normal user; that is, a copy of the database was not requested for research purposes. Advertisements that had the job titles Programmer, Software developer, or Software engineer were searched for at Dice in February and July 2004. The searches produced 9,680 advertisements. Technical skills were searched for using phrases such as Java, SQL, TCP/IP, and Windows from these 9,680 advertisements. These phrases were typically names or abbreviations of different programming languages, operating systems, database vendors, and protocols. Some more general phrases such as "embedded," "object-oriented," or "relational" were used as well but this was not common. Note that during this part of analysis the advertisements were not read but only Dice's search function was used. However, some of the results were counted with five smaller samples ($N = 41…334$) that were read and coded manually.

The two-sided confidence intervals for proportions were calculated using an equation from Milton and Arnold (2003, p. 315) and $1 - \alpha = .99$. The z-test for proportions was used to test if the difference between two proportions was statistically significant.

13.2 Results

The results are divided into three subsections. First, the most common platform, programming language, and database skills are presented in the subsection "Updating previous results." Second, the results from topics that are more characteristic of the approach used in the present research are presented in the subsection "Results characteristic of the present research." Based on the literature survey, the topics of the second subsection have been studied only a little or not at all previously. Third, the present research is discussed.

13.2.1 Updating previous results

The top five platforms, programming languages, and databases are presented in Table 23. For example, the proportion of Java was 35% as Java was mentioned in 3,359 advertisements and the number of advertisements was 9,680. In each column, the sum of the proportions can be greater than 100 because one position could be classified in more than one category. The confidence intervals for the worst cases inside each category are presented below the table.

Only the coding principles of the column Platforms are explained because they are less obvious than the coding principles of the two other columns. The following categories were used: Macintosh, Mainframe/midrange, Unix, Windows, and Cross-platform. For example,

Windows refers to those positions where some Windows operating system or Windows based software such as Visual Basic or SQL Server was mentioned. Products that were available for both Windows and Macintosh, for example, Word and Excel, were classified as Windows if Macintosh was not explicitly mentioned. The category Cross-platform refers to positions where only cross-platform products such as Oracle were mentioned.

Table 23. Top five platforms, programming languages, and databases.

Rank	Platform	Programming language	Database
1	Windows 42%	Java 35%	Oracle 22%
2	Unix 29%	C++ 31%[a]	SQL Server 11%
3	Mainframe/midrange 23%	C 23%[a]	DB2 7%
4	Cross-platform 17%	Visual Basic 15%[a]	Sybase 5%
5	Macintosh 0%	C# 9%	Access 4%

Note. The confidence interval is ±5% ($N = 224$) for platforms, and ±1% ($N = 9{,}680$) for programming languages and databases (except ±5% and $N = 224$ for Access) when the confidence level is $1 - \alpha = .99$.

[a]These proportions were corrected because of false hits or other small problems with Dice's automatic search function. This is explained later in Section 13.2.3.

13.2.2 Results characteristic of the present research

Results about distributed technologies, differences between job titles, differences between entry-level and senior-level positions, and comparing requirements in job advertisements with the degree requirements are presented in the following subsections.

Distributed technologies

For distributed technologies, three categories were used: Microsoft, Sun, and Other. A position was classified in a certain category if at least one skill of the category was mentioned. One position might be classified in several categories. The skills of each category are presented in the following lists:

- Microsoft: .NET, Active X, ASP, DCOM, IIS, and MTS
- Sun: EJB, J2EE, JSP, RMI, and Servlets
- Other: technologies that do not belong in the previous two categories, for example, CORBA, Tuxedo, Tibco, WebLogic, WebSphere, client-server, or applications server.

At least one distributed technology skill was required or desired in 40% of the positions. The proportions of categories were Sun 20%, Microsoft 17%, and Other 8%. These results were counted with the smaller sample ($N = 224$), manual coding was used, and the confidence interval was ±5%. The difference between Sun and Microsoft is not statistically significant and therefore, Sun's

and Microsoft's technologies appear to have held an equally strong position. The most common individual distributed technology skills and their proportions are presented in Table 24.

Table 24. Most common distributed technology skills.

Skill	Abbreviation	Company	Proportion (%)
.NET		Microsoft	19
Active Server Pages	ASP	Microsoft	18
Java 2 Enterprise Edition	J2EE	Sun	13
Java Server Pages	JSP	Sun	8
WebLogic		BEA	5
WebSphere		IBM	5
Enterprise Java Beans	EJB	Sun	4
Java Servlets		Sun	4
Internet Information Server	IIS	Microsoft	3
Common Object Request Broker Architecture	CORBA		2
Distributed Component Object Model	DCOM	Microsoft	2
Microsoft Transaction Server	MTS	Microsoft	1

Note. N = 9,680. Confidence interval is ±1% when 1 - α = .99.

Differences between job titles

It is possible that employers often use the job titles Programmer, Software developer, and Software engineer as synonyms. However, it is reasonable to expect that requirements for software engineers *on average* would emphasize low-level programming skills more than requirements for programmers and software developers. For example, according to Salary Services (2004a, p. 296), software engineers (a) have "experience of real time or embedded software and associated hardware," and (b) are "employed mainly in the electronics, computer, aviation, & defense industries." During this part of the research, the purpose was to analyze whether low-level programming skills would be more common in software engineering positions.

Some results for the subsamples of programmers (n = 5,418), software developers (n = 924), and software engineers (n = 3,338) are presented in Table 25. Only those results that best show the differences concerning the low-level programming skills are presented. It can be noticed that assembler, C, C++, and the phrase "embedded" were more common for software engineering positions. The z-test for proportions was used to test whether the differences between the job titles were statistically significant ($p < .01$). The pairs are marked with small letters if the difference was statistically significant. For example, two letters "b" in the row Assembler mean that the difference between the proportions of programmers and software engineers was statistically significant. Thus, there was some evidence that low-level programming skills were more common in software engineer positions.

Table 25. Some differences in required or desired skills between job titles (n = 5,418, 924, 3,338, respectively).

Skill	Programmer (%)	Software developer (%)	Software engineer (%)
Assembler	1[b]	4	14[b]
C	16[a,b]	30[a,c]	40[b,c]
C++	22[a,b]	48[a]	50[b]
"embedded"	1[b]	5	15[b]

Note. Confidence intervals are ±1, 2, … , 4% when 1 - α = .99.

[a,b,c]In each row, the letter pairs indicate that the difference is statistically significant ($p < .01$) according to the z-test for proportions.

Entry-level versus senior-level positions

Two groups were compared: (1) Entry-level positions that had no word "senior" in the job title and the number of required working years was 0–1 ($N = 41$). These positions had the word "junior" often as part of the job title. This sample was collected from Dice mainly in March 2004 using phrases like "junior" and "jr." (2) Senior-level positions that had the word "advanced," "lead," "principal," or "senior" in the job title or at least five years work experience was required ($N = 73$). These two samples were coded manually. During this analysis, desired skills were excluded and only the required skills were compared. In addition, only broader skill categories were used and individual skills such as Java were not compared because the sample sizes were so small. Only the following skill categories were used: (a) at least one common programming language (C, C++, Cobol, Java, or Visual Basic), (b) at least one common database skill (Access, DB2, "database," Oracle, SQL, SQL Server, or Sybase), and (c) at least one distributed technology skill.

The mean of the number of required skills was greater for the senior-level group. The mean was 3.7 for the entry-level group and 5.2 for the senior-level group. According to the Mann-Whitney test, the difference between the two groups was statistically significant ($p < .01$). Thus, as expected, more technical skills were required in the senior-level positions than in the entry-level positions. The proportions of the different skill categories are presented in Table 26. According to the z-test for proportions, the difference in distributed technology skills was statistically very significant ($p < .001$) but the differences in the categories Programming language and Database were not significant ($p \geq .05$).

Table 26. Proportions (%) of different skills categories for entry-level ($N = 41$) and senior-level positions ($N = 73$).

Group	Programming language (%)	Database (%)	Distributed technology (%)
Entry-level	68	38	27**
Senior-level	73	48	59**

**$p < .01$

In addition, the mean of the numbers of software development life cycle phases were counted for the entry-level and senior-level positions. The phases presented in the IEEE standard (Institute of Electrical and Electronics Engineers, 1990, p. 186) were used for analysis. There were no major differences. Even entry-level software developers were typically required to take part in more phases than just implementation. Tutoring younger software developers and leading small groups of software developers were mentioned often for senior-level positions but obviously not for entry-level positions. The proportions of these duties were not counted because they were non-technical skills.

Undergraduate programs versus required skills

Next, it is considered how well current curricula in the USA correspond to the job market. McCauley and Manaris (2002) reported that in ABET/CAC accredited undergraduate programs the three most common programming languages that were taught first during the academic year 2001–2002 were Java (49%), C++ (40%), and C (11%). In this respect, the correspondence between curricula and the job market was good because these three languages were exactly the same as the three most common programming languages in the job advertisements. This comparison does not imply that all degree programs should use these three languages. There can be other reasons than the popularity of language in industry to choose the programming language used in education—especially the first one. For example, some institutions might use Scheme as the first language because its syntax is simple.

In addition, McCauley and Manaris reported how often various upper-level courses were required. Related skills are presented in Table 27 that combines the results of their survey with those of the present research. The column Proportion is based on their survey and refers to the number of times a course was required in accredited programs.

The author's estimations of how often the related skills were mentioned in Dice's advertisements are presented in the column "Required in advertisements (estimation)." For this analysis, the exact proportions of phrases were counted but they are not presented because the table would become too complex. For example, for the course Database Management Systems the phrases "SQL," "database," "relational," and "query" were searched for. The

respective proportions were 7–32%. Similarly, related phrases from job advertisements for the other courses were searched for as well. The text "Hardly ever" refers to the proportions 0–1%, "Sometimes" to 2–19%, and "Often" to at least 20%.

Table 27. Most common upper-level courses, their proportions in accredited programs, and estimation of how often related skills were required in job advertisements.

Course name	**Proportion (%)**[a]	**Required in advertisements (estimation)**[b]
Operating Systems	96	Sometimes
Programming Languages	87	Hardly ever
Software Engineering	76	Sometimes
Architecture	69	Sometimes
Analysis of Algorithms	67	Sometimes
Theory of Computation	49	Hardly ever
Database Management Systems	31	Often
Networks	18	Sometimes
Compiler Construction	16	Sometimes
Artificial Intelligence	9	Hardly ever
Human-computer Interaction	4	Sometimes

[a]Source: McCauley and Manaris (2002).

[b]See the body text for the explanation.

This part of the present research was the most problematic that is considered in the next subsection. However, Table 27 was not omitted because the results showed that topics for at least eight out of 11 courses were required sometimes or often.

13.2.3 Automatic search function versus manual coding

Next, some results obtained using Dice's automatic search function are compared with the smaller sample ($N = 224$) that was coded manually. The smaller sample was originally gathered in order to find the search phrases that were used for the automatic search function. The smaller sample was gathered in January 2004 whereas the automatic search function was used in February and July 2004. The smaller sample was coded in a similar manner to the way in which it was assumed that the automatic search functioned. In particular, both required and desired skills were included, not just the required skills.

The proportions of the five most common individual skills of each skill category according to the larger sample and the corresponding results of the smaller sample are presented in Table 28. However, the six most common distributed technology skills are presented because the fifth place was tied. In addition, the proportions of Access are presented because it was noticed as a

consequence of comparisons that the phrase "Access" produced several false hits in the automatic search. The differences of the proportions are presented in the column Difference. Two asterisks indicate that the difference was statistically significant ($p < .01$) according to the z-test for proportions. Inside each skill category, the rows are ordered according to the differences.

It can be noticed from the table that some differences were statistically significant. However, the difference of the means of proportions presented at the lowest row is so small that is has little practical relevance. As one could expect for random changes, some differences are positive and some negative. That is, no systematic differences between the two samples were noticed. The random changes of skill requirements in newspaper advertisements are discussed later in Section 14.3.

Table 28. Proportions (%) of various skills using Dice's automatic search function (N = 9,680) and manual coding (N = 224).

Skill	**Automatic search function (%)**	**Manual coding (%)**	**Difference (%)**
Programming languages:			
C	26	16	10**
C++	34	25	9**
C#	9	8	1
Java	35	40	-5
Visual Basic	10	17	-7**
Database skills:			
Access	12	4	8**
SQL	34	27	7
DB2	7	7	0
SQL Server	11	13	-2
Oracle	22	24	-2
Sybase	5	8	-3
Distributed technology skills:			
ASP	18	11	7**
.NET	19	16	3
JSP	8	8	0
WebSphere	5	7	-2
J2EE	13	16	-3
WebLogic	5	8	-3
Mean	**16.1**	**15.0**	**1.1**

**$p < .01$

Next, it is explained how some proportions of the automatic search function were rejected or corrected because systematic errors were found. Previously in Table 23, the corrections were marked with the letter "a." False hits in the

automatic search and other explanations were investigated for the most common skills and for which the differences presented in Table 28 were statistically significant. In addition, SQL was investigated because its difference was almost significant ($p \leq .05$), it was a commonly required skill, and it might be confused with the skills SQL/PL and SQL Server.

As was mentioned previously, it was found that the search phrase "Access" produced several false hits. The reason was that the automatic search function accepted the phrase "access" as well, for example, in the text "… modifying existing programs that access data …" Therefore, the result of the automatic search function was rejected and the proportion of manual coding was used for Access.

One might expect that C was confused with C++ and C#. However, this was not the case. The search phrase "C" produced approximately 10% of false hits such as "C ++" and "Claims P&C" but it did not systematically find the advertisements where C++ or C# was mentioned and C not. Here, the result of the automatic search function was not rejected but just corrected a little because the confidence intervals of the smaller hand-coded sample were quite poor and there was a good basis on which to decide the level of correction necessary. The following corrected proportion of C was used: $25.6\% \cdot 0.9 \approx 23\%$ where 25.6% was the original proportion without correction.

Similarly, the search phrase "C++" resulted in approximately 10% of false hits such as "(not just C++)" and "Visual C++". Thus, the following corrected proportion of C++ was used: $34.2\% \cdot 0.9 \approx 31\%$ where 34.2% was the original proportion without correction.

For the automatic search of Visual Basic, the search phrase used in the automatic search was "Visual Basic." However, it was later noticed that often only the abbreviation "VB" was used in job advertisements, which explained why the proportion of the automatic search function was lower than the proportion of the manual coding (10% and 17%, respectively). Based on the sample that was coded manually and additional automatic searches conducted in Dice, the author estimated that the search "Visual Basic <OR> VB" would produce approximately 50% more hits than just "Visual Basic." Thus, the following corrected proportion of Visual Basic was used: $9.9\% \cdot 1.5 \approx 15\%$ where 9.9% was the original proportion without correction.

The proportions of ASP and SQL were not corrected because false hits or other systematic errors were not found for these skills.

13.3 Evaluation

The discussion about the present research is detailed because a web recruiting service is a new type of data source for job advertisement analysis. First, the content validity of the present research is discussed:

- One could find out from job advertisements that some skills were probably important but determining if some particular skill or subject was *not important* could be much more difficult. Analysis appeared to work well for language and product names such as Java and WebSphere. For search phrases of other kinds, which were typically more general concepts such as "theory of computation" or "data structures and algorithms," the situation was much more difficult. In particular, these problems were evident in analysis of those results that are presented in Table 27. For example, if it was assumed that the course Programming Languages deepened the understanding of programming principles and was thus relevant to any advertisement that mentioned programming or a programming language, the result in the column "Required in advertisements (estimation)" would be "Often" instead of "Hardly ever."
- It was only possible to search for individual skills such as Java but was not possible to get results of broader skill categories such as "at least one programming language." This is a major problem because broader skill categories are very interesting for educational planning, for example. This problem could be solved if a copy of the database was provided for research purposes.

Second, the external validity of the present research is discussed. One problem with the sample used was the possibility that some well-known companies do not advertise in Dice at all or only a little because interested job seekers searched for job advertisements directly from the web site of the company. The author of the present thesis investigated if this was the case for the following companies: HP, IBM, Microsoft, and Sun Microsystems. From these four companies, it appeared that Microsoft was advertising a very small proportion of open positions in Dice but the other three companies advertised in Dice. On October 14, 2004, Microsoft offered approximately 930 software development and testing positions on its own web site but only 20 positions in Dice. However, the total number of software developer positions in Dice was approximately 19,000. Compared with these 19,000 positions, the proportion of Microsoft's positions was only approximately 5%. Thus, correcting the sample by including some Microsoft's advertisements from the web site of the company would have only a moderate effect on the results. Obviously, correcting the sample would probably increase the proportions of skills that were related to Microsoft's products.

Third, the stability of the results over time is discussed. The stability is likely to be modest. As was shown in Section 12, the proportions of the most common skill categories have increased in job advertisements targeted at software developer positions during the past 15 years. This indicates that the proportions of at least some individual skills have also changed. It is even more likely that the changes for individual skills are greater than for broader skill

categories. In addition, how common for individual skills random changes are will be discussed later, in Section 14.3.

Fourth, some practical problems are considered. Here, the following problems are not classified as the properties of validity or reliability; rather, they are viewed as practical problems related to the use of a web recruiting service:

- It was not possible to use the "first code all, categorize and analyze later" approach that was used in the trend analysis of the present thesis. In other words, one has to use traditional content analysis where the search phrases are decided before the coding phase[2]. This would be a lot less of a problem if a copy of the database was provided for research purposes.
- It was not possible to search only for the required skills and exclude the desired skills. As a consequence, the proportions of the large sample ($N = 9{,}680$) of the present research were somewhat greater than if only required skills were used.
- Some search phrases resulted in several false hits. Fortunately, this problem was a lot less serious than expected. It concerned only a few phrases when "SQL" and "access" were the biggest problems. In such cases, it is probably better to use a smaller sample that is coded manually.
- A typical coding problem for job advertisement analyses is known as a fishing expedition advertisement where almost every common skill is mentioned. In some previous research (e.g., Litecky & Arnett, 2001, p. 7), these advertisements have been excluded. In the present research, fishing expedition advertisements were included because it was not possible to exclude them when Dice's automatic search function was used. This problem probably resulted in a kind of constant background humming for the most common skills; that is, the proportions were a little greater than if fishing expedition advertisements had been excluded.
- Generally, the results obtained using Dice's automatic search function corresponded reasonably well with the manually coded sample or the systematic errors found in the automatic search explained the major differences. This is a good property when the reliability of the present research is considered.

Fifth, some probable "blind spots" of job advertisement analysis are considered; that is, these subjects or skills are hard to analyze using job advertisements. Other research methods could be used to gather information about the importance of such subjects and skills. For example, Lethbridge (2000, p. 46) found in his survey that the respondents considered data structures and software architectures as being important. A third example could be concurrent

[2] In the present thesis, a sample of advertisements was coded manually before using the automatic search function for the main analysis in order to find all suitable search phrases and to notice search phrases that might cause false hits.

programming because, for instance, Java could be used for both object-oriented and concurrent programming but it was usually not possible to determine from the job advertisements if concurrent programming was necessary.

The results of the present research are compared with the previous findings and possible differences are discussed as part of the general discussion in Section 21.1.6.

14 Triangulation of job advertisement analyses

In this section, the results of the trend analysis (Section 12) and the cross-sectional analysis (Section 13) of job advertisements are compared. Mainly, the results of the last year of the trend analysis were used because these came from the approximately the same time period as the results of the cross-sectional analysis. The trend analysis data came from September 2004, and the cross-sectional analysis data from February and July 2004.

In addition, the main findings of Part IV are summarized at the end of this section.

14.1 Number of required skills

The mean number of required skills in Computerworld in 2004 was 7.7 whereas Dice's mean was 6.0. Dice's mean was for the sample that was coded manually ($N = 223$). As mentioned previously, it was not possible to count the mean using Dice's automatic search function. According to the Student t test and the Smith-Satterthwaite procedure, the difference is statistically significant ($p < .01$) when Computerworld's standard deviation was 5.7 ($n = 145$) and Dice's 3.8 ($N = 223$).

However, even Dice's mean in 2004 is considerably greater than Computerword's mean in 1990. The means were 6.0 and 3.6, respectively. According to the Student's t test and the Smith-Satterthwaite procedure, the difference is statistically very significant ($p < .001$) when Computerworld's standard deviation was 3.6 in 1990 ($n = 189$) and Dice's 3.8 in 2004 ($N = 223$).

Thus, this comparison indicated as well that the number of required skills has increased during the last 14 years but according to Dice's result in 2004 the increase is less dramatic than according to the results of Computerworld alone.

14.2 Proportions of various skills

The proportions of both analyses and the difference of proportions are presented in the following tables. The proportions of five skill categories that were presented previously in Section 12 are shown in Table 29. The proportions of the five most common skills of these categories based on Dice's results are presented in the other tables. In Table 29 and Table 30, the Dice's results are from the manually coded sample whereas Dice's results of the other tables are completely or mainly obtained using the automatic search function.

The fifth position was decided according to Computerworld's results if it was a tie according to Dice's results. The rows are ordered according to the differences. Two asterisks indicate that the difference was statistically significant ($p < .01$) according to the z-test for proportions.

The details of each table are not discussed but the results are compared only at a more general level. It can be noticed from the tables that the differences of approximately half of the items were statistically significant. As one can expect according to the comparison of the means, often the proportions of Dice were smaller than in Computerworld. As a consequence, the increased trends presented in Section 12 would probably be less sharp if Dice was used as a data source for the trend analysis.[3]

Table 29. Proportions (%) of at least one skill in five skill categories in Dice ($N = 223$) and Computerworld ($n = 149$) in 2004 and difference of these proportions.

Skill category	Dice	Computerworld	Difference
Distributed technology	31	65	34**
Database	37	70	33**
Programming language	54	71	17**
Operating systems	42	52	10
Networking	6	8	2

**$p < .01$

Table 30. Proportions (%) of most common platforms in Dice ($N = 223$) and Computerworld ($n = 149$) in 2004 and difference of these proportions.

Platform	Dice	Computerworld	Difference
DOS or Windows	37	65	28**
Unix	25	41	16**
Mainframe/midrange	21	21	0
Macintosh	0	0	0
Cross-platform	23	10	-13**

**$p < .01$

Table 31. Proportions (%) of most common programming languages in Dice ($N = 9{,}680$) and Computerworld ($n = 149$) in 2004 and difference of these proportions.

Programming language	Dice	Computerworld	Difference
Visual Basic	15	32	17**
Java	35	49	14**
C	23	30	7
C++	31	31	0
C#	9	7	-2

**$p < .01$

[3] As was explained previously, Dice was not used because the past data were not publicly available.

Table 32. Proportions (%) of most common database skills in Dice and Computerworld (n = 149) in 2004 and difference of these proportions.

Skill	Dice[a]	Computerworld	Difference
Oracle	22	51	29**
SQL Server	11	25	14**
DB2	7	13	6**
PL/SQL	6	7	1
SQL	19	11	-8

**p < .01

[a]N = 223 for SQL and N = 9,680 for the other skills.

Table 33. Proportions (%) of most common distributed technology skills in Dice (N = 9,680) and Computerworld (n = 149) in 2004 and difference of these proportions.

Skill	Dice	Computerworld	Difference
ASP	18	28	10**
J2EE	13	14	1
.NET	19	11	-8
JSP	8	11	3
WebLogic	5	7	2

**p < .01

14.3 Random changes

So, there were several statistically significant differences between Dice and Computerworld in 2004. In addition, some statistically significant differences were found between Dice's samples in 2004 when the automatic search function and the manual coding were compared (Section 13.2.3), and between Computerworld's samples from January 2004 and August 2004 (Surakka, 2005c, pp. 24–25). It will also be presented later in Section 21.1.6 that Athey and Plotnicki (1998) found large differences between ten American cities. They did not report if the differences were statistically significant but some differences were so large that they apparently were significant. Based on the comparison of the results of the present thesis with the results of Salary Services Ltd. (2004a), there are some statistically significant differences between the USA and the UK (later Section 21.1.6).

Therefore, the author became gradually more and more convinced during the thesis project that the random changes of skill requirements in job advertisements are large enough and sufficiently common to make it easy to find at least some statistically significant differences between almost *any* two data sources, time periods, cities, or countries.

Next, some ways to manage this problem are listed:

- The results of different research can be compared, which is conducted later in Sections 21.1.2 and 21.1.6. In this way it is possible to find at least some greater changes or the most common skills that are reported by several sets of research. For example, it can be noticed by comparing different research results that the need for Oracle has increased strongly during the past 20 years.
- The use of broader skill categories as well might be beneficial because one could assume that the random changes are more evident in individual skills than in skill categories.
- One should consider if the statistically significant differences are so large that they have some practical meaning as well. For example, even a one percent difference of the operating system Macintosh might be statistically significant but it has little relevance to curriculum planning because both proportions 0% and 1% would show that the need is low.
- Obviously, one should consider if the differences make sense. For example, it was reasonable to expect that the need for distributed skills would increase during the past ten years as a consequence of WWW technology. Unfortunately, it is easy and tempting to figure out some more or less plausible explanation for almost every change *afterwards*. This problem might be less serious if a researcher states some expectations and records these *before* the data are coded. In the present job advertisement analyses, the only stated expectation was the increase of distributed technology skills.

Main findings of Part IV

The main findings of Part IV are:

- The mean of the number of required technical skills increased from 3.6 in 1990 to 7.7 in 2004. The technical requirements have changed and have required greater versatility; in particular, distributed technology skills were required in 2004 considerably more often than in 1990.
- The top five skills in 2004 were Windows, Java, C++, SQL, and Unix.

Part V: Viewpoint of basic studies

The results of three different content analyses are presented in the present part. These results were published earlier in the Computer Science Education journal (Surakka & Malmi, 2005b). The present part is a case study when the case example is the Degree Program of Computer Science and Engineering at the institution.

In the other parts of the present thesis, the scope is in advanced or upper-level studies (study years 3–5) when in the present part the viewpoint is in the introductory or lower-level studies (study years 1–2). Another difference is that the present part is not targeted only at software developer positions but rather at IT positions of all kinds. For example, all Master's theses from various specializations were analyzed, not just theses of specialization in Software Systems. The present part was included in the thesis because some documents used in it were suitable for analyzing the importance of physics and mathematics, in particular. This was not possible using job advertisements.

The scope of this part of the thesis is presented in Figure 11 using the data sources and research methods. The figure is the same as Figure 2 presented previously in Section 3.2 but the boxes related to this part are gray.

First, the case example is described. After this, the different documents are analyzed in the following order: Master's theses, internship reports, and course catalog of the institution. Finally, the results of the three content analyses are compared with each other.

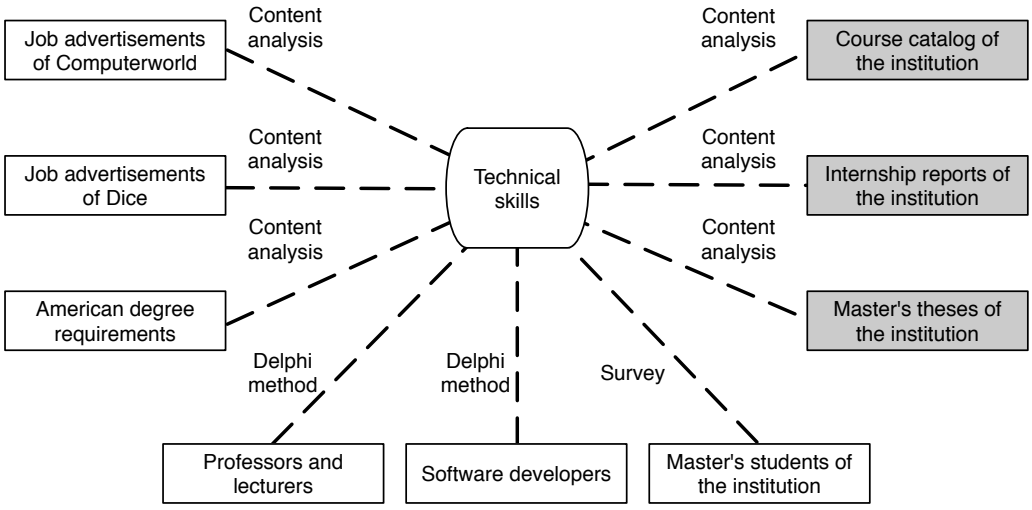

Figure 11. Scope of Part V of thesis.

15 Description of case example

The Helsinki University of Technology is located in Finland, which is a member of the European Union, located in Northern Europe, and has a population of approximately five million people. Finland has 20 institutions that have university status. Ten of them are specialized institutions such as the universities of technology or the institutions of arts. The Helsinki University of Technology is one of these specialized institutions. Higher education in Finland is free of charge. The detailed description of the Finnish higher education system can be found, for example, from Kuikka (1992) or in Ministry of Education (2005).

According to the Carnegie Classification of Institutions of Higher Education (Carnegie Foundation, 2005), the institution could be classified as "Doctoral/Research Universities—Extensive" or at least as "Doctoral/Research Universities—Intensive." More information about the institution can be found, for example, from the annual report (Helsinki University of Technology, 2003).

During the academic year 2004–2005, the institution offered 17 degree programs and the total intake was approximately 1,500 undergraduate students. All degree programs applied *numerus clausus*, in which entrance examinations were a key element. The specialization in Software Systems was part of the Degree Program of Computer Science and Engineering the intake of which was approximately 140 students.

The institution changed its degree structure during the thesis project. The old degree structure is presented in the next subsection because the results of this case study are mainly from the years 2000–2003 when the old structure was still used. The new degree structure will be presented later in Sections 23.1 and 23.2.

15.1 Old degree structure

Before the academic year 2005–2006, a Bachelor's degree was not offered, the nominal duration of a Master's degree was five years, and the degree was in two parts. The old structure of the Master's degree is presented in Figure 12. The first part of the degree, which a student should complete during the first and the second years of study, was the same for all students of the program. During the advanced studies (years 3–5) a student had to: (a) choose one area of study called an "option[4]" and (b) choose one "major" that belonged to that option. The program offered five options: Telecommunications Software and

[4] The word "specialization" is normally used instead of "option" or "major" in the other parts of the thesis.

Applications, Software Techniques, Neural Networks and Signal Processing[5], Software Business and Engineering, and Theoretical Computer Science. Typically one option contained 3–5 majors. Altogether, these five options contained 17 majors. The majors varied from the mathematically more demanding "Formal Methods in Computer Science and Engineering" to softer majors such as "Venturing in Digital Economy" and "User-centered Product Development."

In Figure 12, the numbers refer to European Credit Transfer System (ECTS) credits. The institution used Finnish credits before the 2005–2006 academic year. However, these credits were converted to ECTS credits because the Finnish credit system is not widely known internationally. According to the Helsinki University of Technology (2005a), 60 ECTS credits equals 1,600 studying hours.

The present thesis was targeted at the option Software Techniques, the major Software Systems, and the Master's thesis of those students who belonged to the major Software Systems. The extent of the target area was approximately 100 ECTS credits. This would correspond on the one hand to 20 courses if it was assumed that the extent of one course was five ECTS credits or on the other hand to approximately one and a half study years if a student studied 60 ECTS credits per year. In addition, it can be noticed that the Master's thesis is a substantial part of the target area. By credits, its proportion was 30%.

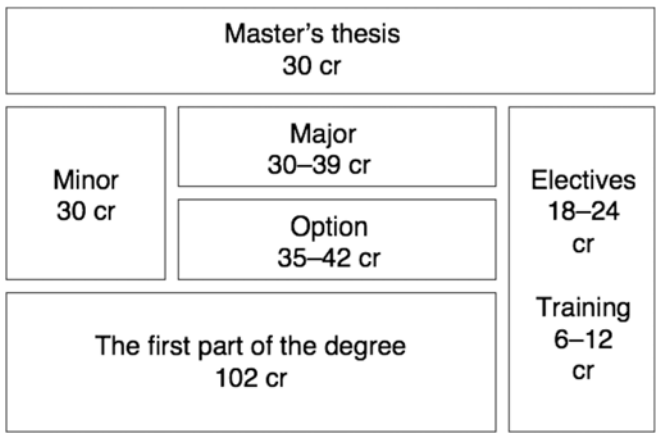

Figure 12. Structure of old Master's degree (Kerola, Knuuttila, & Kujanpää, 2004, p. 21). Abbreviation "cr" means ECTS credits.

[5] The official name is "Computer and Information Science" but "Neural Networks and Signal Processing" is used here because it is probably easier to understand for most readers.

15.2 Generality of degree program

For brevity, the degree program of the institution was compared only with programs in the USA. In particular, the survey by McCauley and Manaris (2002) of undergraduate programs that were accredited by the Computer Accreditation Commission (CAC) of the Accreditation Board for Engineering and Technology is referred to. As a metric for extent, McCauley and Manaris used semester-credits according to the system in the USA. The author converted semester-credits to ECTS credits. According to Engels and Roberts (2001, p. 16), typical 3-credit semester course in the USA has 40 hours of lecture time and (p. 15) "As a general guideline, the amount of out-of-class work is approximately three times the in-class time." Thus, one semester-credit equals 53.33 studying hours ($40 \cdot 4 / 3 \approx 53.33$). According to the Helsinki University of Technology (2005a), 60 ECTS credits equals 1,600 studying hours. Thus, one ECTS credit equals 26.67 studying hours ($1600 / 60 \approx 26.67$) and one semester-credit equals two ECTS credits.

The breakdown by subjects of the minimum requirements for CAC programs and the program of the Helsinki University of Technology (TKK[6]) is presented in Table 34. For the TKK program, only the credits in the first part of the degree (105 credits) and in the option Telecommunications Software and Applications (35 credits) were counted. This option was selected for the analysis because it is the most popular. Thus, the number of credits was 140. This was considerably smaller than the average number of credits of the undergraduate degree in the CAC programs (250 credits), but this part of the studies has most in common with the CAC degree studies. For example, it would not be reasonable to count how many credits of mathematics or computer science were required in a minor because this could vary from 0 to 30. From these 140 credits in the TKK program, 18 credits were excluded from Table 34 because they were general studies, elective studies, or electrical engineering and therefore not relevant to this comparison. It seems that the TKK program was quite typical with respect to the value of mathematics, physics, and computer science. The biggest difference was that the TKK program had no compulsory other sciences such as chemistry whereas the average for the CAC programs was 18 credits.

[6] TKK is the Finnish abbreviation of Teknillinen korkeakoulu. According to the administrative instructions of the institution, TKK should be used instead of the English abbreviation HUT.

Table 34. Breakdown by subjects of minimum requirements of CAC accredited programs ($N = 40$) and Helsinki University of Technology program.

Program(s)	ECTS credits			
	Computer science	**Mathematics**	**Physics**	**Other sciences**
CAC accredited programs[a]	94	34	10	18
Helsinki University of Technology	74	36	12	0

[a]Source: McCauley & Manaris (2002, p. 3). American credits were converted to ECTS credits.

Some options had considerably more students than others. This is important for the needs assessment because if, for example, few students chose the mathematically demanding option and the other options did not require a great deal of mathematics, the average or typical need for mathematics during the advanced studies would be low. The year 2002 graduates ($N = 78$) were distributed among the different options as follows: Telecommunications Software and Applications (55%), Software Techniques (23%), Neural Networks and Signal Processing (14%), Software Business and Engineering (6%), and Theoretical Computer Science (1%). It can be noticed that over 50% of the students graduated from the option Telecommunications Software and Applications. Thus, it is likely that the program was biased in this respect when compared with a typical computer science program.

16 Content analysis of Master's theses

The present research concerned topics needed in Master's theses. In some institutions, a student may choose if he or she conducts a thesis or studies advanced courses. At the institution, Master's thesis was required for all students and a principal, such as a company or a research institution, typically provided a topic for a thesis. A principal probably thought carefully about the topic, because normally the principal paid a salary or gave a scholarship to the student who did the work and wrote the thesis. Therefore, it was possible to find out something about the needs of the employers by analyzing the theses of the degree program.

16.1 Research method

The program started in 1984 and the first students graduated in 1989. The theses that were completed during the period 1989–91, in 1996, between January and June in 1999, and January and September in 2002 were analyzed. From the given periods, the samples were complete or almost complete (97–100%). A typical thesis had 50–100 pages. Approximately half of the theses were written in Finnish and approximately half in English. From each Master's thesis, the author of the present thesis read the abstract, browsed the thesis, and frequently read the list of contents. The main purpose of browsing was to find mathematical or other equations from a thesis. At most 15 minutes was used to analyze a thesis. The subjects were classified into one of the following three categories that relate to the different subjects:

1. A subject was unnecessary for the thesis or its necessity was insignificant. Necessity was classified into this category if, for example, mathematics was used only to count some simple metrics such as percentages.
2. A subject was occasionally or moderately needed. The necessity for programming was classified as occasional or moderate if, for example, a student programmed only some small Matlab macros.
3. A subject was important for the thesis. Mathematics was classified as important to a thesis if, for example, a mathematical model was the basis for the simulation program.

16.2 Results

The proportions of the third category "important" are presented in Table 35. The subjects are ordered by the results of the column 2002. The proportions of computer technology were counted for the year 2002 only. Note that the necessity of programming has remained high and steady over the years, but the

necessity of data communications and networking has arisen from the level of 1989–91. This is evidently a result of the increasing importance of the telecommunications industry in Finland during the 1990s. The necessity of mathematics has varied a little but remained, roughly, the same during the years.

Table 35. Necessity of different subjects in Master's theses.

Subject	Proportion (%)			
	1989–91 (n = 49)	1996 (n = 48)	1999 (n = 66)	2002 (n = 60)
Programming	55	48	62	67
Data communications and networking	29	44	59	58
Mathematics	16	19	11	15
Computer technology	—	—	—	13
Signal processing	6	10	17	5
Theoretical computer science	4	6	5	5
Physics	2	2	3	2

Note. Dash (—) means that the proportion was not counted.

Based on the z-test for proportions, the difference between the proportions of the years 1990–91 and 2002 is statistically significant ($p < .01$) for data communications and networking and not significant ($p \geq .05$) for the other subjects.

16.3 Evaluation

The content validity of the present research is likely to be good or satisfactory. Since most students' Master's theses involve a project in industry or in another organization than the institution where the present thesis was carried out, this analysis provided a fair view of the subjects needed in practical work. During the analysis, it was noticed that the theses were probably a valid data source because they were so detailed.

The external validity is less good, at least for part of the results. As was mentioned in Section 15.2, over 50% of the students graduated from the option Telecommunications Software and Applications. Thus, it is likely that the program was biased in this respect when compared with a typical computer science program. As a consequence, the proportions of "Data communications and networking" in Table 35 might be a lot smaller if the research were repeated in other computer science programs. For the other items of Table 35, the external validity is likely to be satisfactory or good.

Based on the results of the present research, the stability of results concerning programming and "data communications and networking" over time is likely to be modest, or satisfactory at best. As was shown in Table 35, the proportions of these subjects have changed strongly during the period 1989–

2002. This gives a reason to assume that the proportions might change in the future also. However, based on the results, the stability of mathematics, signal processing, theoretical computer science, and physics might be good, because no trend could be noticed. Based on the results of the present research, it is not possible to evaluate how stable the results of computer technology are.

The results of the present research were not compared with any in previous publications because no similar research was found.

17 Content analysis of internship reports

The present analysis relates to the internships of the students. It provides information about the practical needs of subjects studied in the first two years.

17.1 Research method

In the computer science program of the institution, students were required to take 4–8 credits internships known as professional training (Ylihеljo, Mulari, & Hettula, 2002, p. 92). The degree regulations of the institution (Helsinki University of Technology, 2001a, para. 17) stated: "A three-week training period is equivalent to one credit. Practical training may consist of periods of working environment experience and professional training." The working environment experience is work that does not demand professional studies in computer science and engineering. The professional training is based on professional studies, and normally supervised by a B.Sc. (Eng.) or a M.Sc. (Eng.) (Ylihеljo et al., 2002, p. 92).

A student had to attach an employment report to the application for internship credits as an appendix. As a minimum, the employment report had to contain information about the student's duties and the duration of the internship (ibid., p. 94). A typical report was written on a single page. The applications for the working environment experience were excluded and only the applications for internships were analyzed. Ninety-five applications were accepted between January and June 2000. There was no time limit; that is, the internship could have been conducted several years before the application was submitted. However, most internship periods were quite recent: 96% were from 1995–2000 and only 4% before 1995. It is possible that an application had more than one employment report attached as an appendix because the students were allowed to combine internship periods. However, each application was classified as one unit of the sample, regardless of the number of reports from which it was composed.

Based on the applications, it was analyzed which subjects were necessary for each internship position. A yes/no principle was used: a subject was either needed or not. Usually, programming and data communications and networking were the only subjects that were clearly mentioned in the reports. For the other subjects, the author of the present thesis made his own conclusions about their necessity. For example, it was judged that mathematics was needed if it was not mentioned in the report but the report suggested that it probably was needed. Typically, such suggestions were the words "simulation," "model," and "modeling." It was classified that knowledge about computer technology was needed if the student's task was system administration or operation.

17.2 Results

The necessity of different subjects in internships is presented in Table 36. As might be anticipated, professional subjects were needed more than theoretical subjects.

Table 36. Necessity of different subjects in internships ($N = 95$).

Subject	Proportion (%)
Programming	84
Data communications and networking	21
Computer technology	15
Mathematics	4
Signal processing	3
Theoretical computer science	2
Physics	0

17.3 Evaluation

The content validity of the present research is modest, or satisfactory at best. The applications for internship credits were often so brief that the results may not be valid, particularly in the case of non-professional subjects. It may be that most students and employers classified general studies, mathematics, and physics as non-professional subjects and, therefore, did not mention them in the applications and employment reports. However, the present research was not removed from the thesis because the results indicate that at least some skill categories were required.

The external validity is problematic as well. As mentioned in Section 15.2, over 50% of the students graduated from the Telecommunications Software and Applications option. Thus, it is likely that the program was biased in this respect when compared with a typical computer science program. As a consequence, the proportion of "Data communications and networking" in Table 36 might be smaller if the research were repeated with respect to another computer science program.

The stability of the results over time is likely to be modest, or satisfactory at best. As was shown in Section 12, the proportions of the most common skill categories have increased during the past 15 years in job advertisements targeted at software developer positions. This gives reason to suspect that the requirements for internships have also gradually changed and might continue to change. However, most skill categories considered in the present research were quite broad. As a consequence, the changes over time are probably more moderate than for individual skills such as Java.

The results of the present research were not compared with any in previous publications because no similar research was found.

18 Content analysis of course prerequisites

The purpose of the present content analysis was to discover how important the basic courses covered in the first two years were for advanced studies. Which topics were needed later and which remain isolated from the succeeding studies?

18.1 Research method

The research method involved the following steps: (a) The total number of credits of the compulsory courses of each option were counted. The minimum number of credits was used if a course had the varying number of credits. (b) It was checked which prerequisites the compulsory courses had. (c) For each subject (programming, data communications, mathematics, physics, signal processing, theoretical computer science, and computer technology), it was counted the number of credits of compulsory courses that had the subject as a prerequisite. (d) This value was divided by the total number of credits of the option. This result was called as the prerequisite proportion of the subject. In the following, the previous steps are applied to the option Software Techniques as an example where only the prerequisite proportion of programming is counted. (a) There were eight compulsory courses in the option with a value of 37.5 ECTS credits[7]. (b) From eight courses, seven courses had programming as a prerequisite. (c) The value of these seven courses was 34.5 ECTS credits. (d) Therefore, the prerequisite proportion of programming for the option Software Techniques was 92% ($100 \cdot 34.5 / 37.5 = 92$).

The course catalog of the institution (Helsinki University of Technology, 2002a) defined the prerequisites for each course; normally the course codes of the prerequisites courses were given. Often, some prerequisite courses were not listed in course descriptions, but only those courses were listed that were the nearest in the chain of prerequisites. During the analysis, the chain of prerequisites was followed to the end of the chain. For example, the course Software Project Management had the course Introduction to Software Engineering as a prerequisite but no programming course was mentioned as a prerequisite. However, Introduction to Software Engineering had a

[7] The credits of the institution were converted into European Credit Transfer System (ECTS) credits because the Finnish credit system is not widely known internationally. According to the Helsinki University of Technology (2005a), 60 ECTS credits equals 1,600 studying hours.

programming course as a prerequisite. In this case, programming was classified as a prerequisite for the course Software Project Management. In some cases, the definition of prerequisites was missing entirely from the course catalog. This could mean that the course had no prerequisites or there was a mistake in the course catalog. Prerequisites were added if it was thought that the missing definition was a mistake. For example, the Automation and Control Technology course had no prerequisites defined in the course catalog. However, based on the author's knowledge of the topic, it was considered that mathematics should be a prerequisite and the situation was classified as such. Finally, it was always judged that the course had no prerequisites if there was an explicit statement that a course had no prerequisites.

18.2 Results

How often a subject was mentioned as a prerequisite for the compulsory courses of the options it is presented in Table 37. The rows are ordered according to the relative need of programming because it was needed most often. The columns are ordered so that the subjects having typically greater proportions are on the left. There is no column for physics because physics was not a prerequisite for any of the compulsory courses in question.

Based on the results, programming was very necessary (over 50%) in four options and somewhat necessary in one option. Data communications and networking was somewhat (10–49%) necessary in all options. Mathematics was somewhat necessary in two options but not needed in other three options. As one might expect, data communications and networking were needed most in the option Telecommunications Software and Applications, and theoretical computer science in the option Theoretical Computer Science. Signal processing was somewhat necessary only in the option Neural Networks and Signal Processing. Computer technology was needed a little (0–9%) only in one option.

Table 37. Necessity of different subjects based on prerequisites in options.

Option		Proportion as prerequisite (%):					
	ECTS credits	Prog-ramming	Data commu-nicat.	Mathe-matics	Signal pro-cessing	Theore-tical CS	Compu-ter tech-nology
Software Techniques	37.5	92	16	0	0	0	0
Telecommunications Software and Applications	30.0	80	45	0	0	0	0
Theoretical Computer Science	42.0	71	14	21	0	21	0
Software Business and Engineering	42.0	64	14	0	0	0	0
Neural Networks and Signal Processing(a	33.0	32	18	45	32	0	9

[a)]The official name is "Computer and Information Science" but "Neural Networks and Signal Processing" is used here because it is probably easier to understand for most readers.

18.3 Evaluation

First, the content validity of the present research is discussed. It is possible that in some cases the prerequisites stated in the course catalog were not really needs of the students but more like the wishes of the teacher. On the other hand, it is possible that a teacher did not mention prerequisites that he or she thought were obvious. For example, all students of the program should have basic knowledge of high-school-level mathematics because a mathematics exam was a compulsory part of the entrance exams of the program. However, it was assumed that the prerequisites stated in the course catalog of the institution were valid.

Second, the external validity of the present research is discussed. As was mentioned in Section 15.1, the program also offered the options "Neural Networks and Signal Processing" and "Telecommunications Software and Applications." It is possible that these options are not typically offered in computer science programs, and therefore, it is likely that the program was not representative in this respect when compared with a typical computer science program.

Third, the stability of the results over time is discussed. The results will probably be quite consistent over time because (a) the degree requirements of universities typically change slowly, and (b) they represent relationships between specializations and courses, which represent more abstract or broader concepts. Relationships between such concepts are likely to be more stable over time than results for individual skills such as Java.

The results of the present research were not compared with any in previous publications because no similar research was found.

19 Triangulation for basic studies

The summative results of three content analyses are presented in Table 38. Each subject was given a rank based on proportions. Tied rank was used if the proportions were equal. From the analysis of the Master's theses, only the results of the year 2002 were included. The sum of ranks is presented in the column Sum. The rows are ordered according to the sum of ranks. It can be noticed that based on all three analyses, programming was the most necessary, data communications and networking was the second most necessary, and physics was the least necessary subject.

The necessity of mathematics was lower based on the analysis of internship reports than it was in the other two analyses. The difference is clearer if the proportions are compared (4% vs. 13% and 15%). A possible explanation for the difference was that the necessity of mathematics could be detected more easily from the course catalog and the Master's theses because they were more detailed than the applications for internship credits.

The necessity of computer technology was lower based on the analysis of course prerequisites than in the other two analyses. The proportions were 2% versus 13% and 15%. A possible explanation is case related because the computer science program of the institution did not offer a specialization or other advanced studies in the area of computer technology.

Table 38. Ranks of three content analyses and sum of ranks.

Subject	**Course prere-quisites**	**Intern-ships**	**Master's theses in 2002**	**Sum**
Programming	1	1	1	3
Data communications and networking	2	2	2	6
Mathematics	3	4	3	10
Computer technology	6	3	4	13
Signal processing	4	5	5	14
Theoretical computer science	5	6	5	16
Physics	7	7	7	21

Part VI: Putting it all together

The results presented previously in Parts II–IV are summarized, conclusions are drawn, recommendations are presented, and the thesis is discussed in this part.

20 Summative triangulation

In this section, the results presented previously in Parts II-IV are summarized into one table. The results presented in Part V (Viewpoint of basic studies) were not included because they were less accurate. In addition, the results presented in Section 10 regarding cognitive skills were not included because the research was not a needs assessment and the results were not suitable for summarization. However, additionally, the results of Part V and Section 10 are compared with the other results but this comparison is more general.

The purpose of this summarization was to determine which subjects or skills were evaluated as being important. The same 42 items were used that were used in the Delphi study targeted at software developers, for example. For each subject or skill, a sum of points was counted so that from a single research project it was possible to get zero or one point. One point means that a subject or skill was important according to the research in question. Two different types of job advertisement analyses were classified as one research project. Thus, the maximum number of points was six because six separate sets of research or research types were used. The research and criteria used are presented in Table 39.

Table 39. Criteria used for summarization of results.

Research(es)	**Criterion**
Content analysis of degree requirements	The courses that were mentioned most often (Table 8 in Section 5.2.3).
Concept analysis	The courses that were estimated as being the most central (Table 4 in Section 4.2.3).
Delphi study targeted at the software developers	The mean of evaluated importance was at least 3.0 (Table 11 in Section 7.2.2).
Delphi study targeted at the professors and lecturers	The mean of evaluated importance was at least 3.0 (Table 13 in Section 8.2.2).
Survey targeted at the Master's students	The mean of evaluated importance was at least 3.0 (Table 16 in Section 9.2.2).
Job advertisement analyses	The proportion of a skill category was at least 50% in 2004 in trend analysis (Figure 10 in Section 12.2.2) OR the proportion of an individual skill was at least 25% in the cross-sectional analysis of job advertisements (e.g., Java for object-oriented programming in Table 23 in Section 13.2.1).

Next, some issues related to the selected criteria are discussed briefly:

- One might wonder, for example, if the results of the Delphi study targeted at software developers should be weighted more than the results of the survey targeted at the students. However, no such weights were used because it was assumed that the student's opinions were relevant for entry-level positions, in particular.
- The limits used for the means and proportions are more or less artificial but nevertheless, the author of the present thesis had to select some limits. For three questionnaires, the limit of 3.0 was selected because it matches with the answering option "Somewhat important" of the questions used. For the trend analysis of job advertisement, the limit of 50% was used for skill categories because one can consider that a skill category is worth of studying if the probability of using those skills in the future is more than 50%. For the cross-sectional job advertisement analysis, the limit of 25% was used for individual skills because such high proportions are rare and indicate a strong position among alternative technologies of a skill category.

The results are presented in Table 40. The columns from "Concept analysis" to "Job advert." refer to the different sets of research of the present thesis. The rows are ordered first according to the column Sum and then according to the names of the subject or skill.

Table 40. Summarized results of present thesis.

Subject or skill	Concept analysis	Delphi study (developers)	Delphi study (professors)	Survey (students)	Job advertisements	Degree requirements	Sum
Mathematics, physics, and theoretical CS:							
Discrete mathematics			1				1
Other areas of theoretical CS (e.g., automata)		1					1
Logic (in particular, propositional and predicate l.)							0
Mathematics for continuous systems							0
Physics							0
More technical or part of the operational system:							
Operating systems	1	1	1	1	1	1	6
Database managements systems	1		1	1	1	1	5
Distributed systems	1	1	1		1	1	5
Compilers	1	1	1			1	4
Concurrent programming	1	1	1	1			4
Data structures and algorithms	1	1	1	1			4
Object-oriented programming		1	1	1	1		4
Procedural programming		1	1	1	1		4
Software architectures	1	1	1	1			4
Computer architecture		1	1			1	3
Computer/data security		1	1	1			3
Internet protocols		1		1		1	3
Implementing techniques of user interfaces			1	1			2
Script programming		1		1			2
Computer graphics						1	1
Embedded systems	1						1
Extensible Markup Language (XML) techniques				1			1
Functional programming		1					1
Systems programming		1					1
Artificial intelligence and knowledge engineering							0
Implementing techniques of WWW systems							0
Logic programming							0
Telecommunications techniques other than Internet pr.							0
Real-time systems							0
Software eng. (different phases of life cycle):							
Concept exploration		1	1	1			3
Design		1	1	1			3
Implementation		1	1	1			3
Requirements		1	1	1			3
Test		1	1	1			3
Approval							0
Installation and checkout							0
Operation and maintenance							0
Packaging and delivery							0
Retirement							0
Software engineering (possible in several phases):							
Documenting		1	1	1			3
Project management		1	1	1			3
Version and configuration management		1	1	1			3

Based on the summarization, the following subjects were evaluated as being important. These items received from four to six points and are presented in alphabetical order:

> compilers, concurrent programming, data structures and algorithms, database management systems, distributed systems, object-oriented programming, operating systems, procedural programming, and software architectures.

Based on the summarization, the following subjects or skills were not important because they received no points. The items are presented in alphabetical order:

> artificial intelligence and knowledge engineering, implementing techniques of WWW systems, logic (in particular, propositional and predicate logic), logic programming, mathematics for continuous systems, telecommunications techniques other than Internet protocols, physics, and real-time systems.

However, all these items, except mathematics for continuous systems and physics, were evaluated as somewhat important (mean greater than 2.4) by at least one respondent group of questionnaires:

- The mean of the professors and lecturers was 2.5 for artificial intelligence and knowledge engineering.
- The means of all three respondent groups were 2.7 or 2.8 for the implementing techniques of WWW systems.
- The means of the software developers and the professors and lecturers were 2.8 and 2.9, respectively, for logic (in particular, propositional and predicate logic).
- The mean of the professors and lecturers was 2.6 for logic programming.
- The mean of the students was 2.5 for telecommunications techniques other than Internet protocols.
- The means of the software developers and professors and lecturers were 2.6 and 2.7, respectively, for real-time systems.

Thus, the only subjects that were not important according to the results of the present thesis were mathematics for continuous systems and physics.

In addition, the following phases of life cycle were less important according to the summarization: approval, installation and checkout, operation and maintenance, packaging and delivery, and retirement. Similarly, the phases approval, installation and checkout, and operation and maintenance were evaluated as being somewhat important (mean greater than 2.4) by at least one respondent group. Thus, only the phases packaging and delivery and retirement were deemed not to be important according to the results of the present thesis.

The ranks of subjects according to the results of Part V (Viewpoint of basic studies) and the summarization presented in Table 40 are presented in Table 41. Comparison of signal processing was not possible because it was not used in the questionnaires nor analyzed in the job advertisement analyses. The results correspond quite well but there are two differences: (a) mathematics was more important according to Part V than according to Table 40 and (b) theoretical computer science was more important according to Table 40 than according to Part V.

Table 41. Ranks of subjects according to results of Part V and Table 40.

Subject	**Part V**	**Table** 40
Programming	1	1
Data communications and networking	2	2
Mathematics	3	4
Computer technology	4	3
Signal processing	5	—
Theoretical computer science	6	4
Physics	7	7

Note. Dash (—) means that no relevant result was obtained.

The results of Section 10 are so different to the items of Table 40 that comparison is less useful. The results correspond on a very general level because the cognitive skills listed in Section 10 were related to the software development phases design, implementation, and test that are among the most important life cycle phases according to the results of Table 40. However, this is an obvious finding.

21 Discussion

The present thesis is discussed in this section. First, the results of the present thesis are compared with those of previous publications. Second, the present thesis is evaluated as a whole. Third, some conclusions are drawn. Fourth, some recommendations targeted at all computer science programs are presented. The case-specific comparison and recommendations will be presented later in Part VIII. Fifth, academically and practically oriented curricula, and admission procedures are discussed. Finally, future research is considered.

21.1 Comparison with previous research

The results of the present thesis are compared with the results of previous publications in this section. The triangulations, the trend analyses, and the questionnaires targeted at software developers are compared first because they were the most important for the conclusions of the present thesis. After this, the rest of the comparisons are presented according to the general structure of the present thesis.

The concept analysis of "software systems" is not compared because no similar previous publications were found. In all subsections, previous publications not deemed relevant are not compared with the present thesis, and not even mentioned. Explanation was given previously in Section 2 as to whether a publication was relevant or not.

21.1.1 Triangulations

In the present subsection, the comparison is limited to the level of agreement between different respondent groups, not to differences between skill categories or individual skills.

In the research by Mawhinney et al. (1995), the students and the employers evaluated the importance of 162 questionnaire items. They wrote (ibid., p. 234): "Perhaps the most important result was the consistent positive correlation between the rank-ordered means for two sample groups. Correlation values in the range found here (.55 to .99) are exceptionally high and indicate that when comparing items against each other within the same knowledge group the students are in remarkable agreement with employers." Apparently the text ".55 to .99" means that they divided 162 questionnaire items into more than one category and calculated the Spearman rank-correlation coefficients for each category whereas in the present thesis only one coefficient was calculated

for all 42 items. Anyhow, strong correlation was found in the present thesis as well when the Spearman rank-correlation coefficient was 0.86.

In addition, Mawhinney et al. (1995, p. 234) wrote: "Overall, the responses to more than 75% of the total 162 questionnaire items were found to be significantly different when comparing the students to employers." In the present thesis, the equivalent proportion was only 12% (5 out of 42 items, $p < .01$, see Table 22 in Section 11). Thus, perhaps in the present thesis the agreement was considerably greater than in their research. However, this comparison and conclusion would be unfair to their research because their sample sizes were so much greater than in the present thesis. It is possible and even likely that more significant differences between the respondent groups would be found if the sample sizes of the present thesis were greater. In addition, they did not report whether the significance level $p < .01$ or $p < .05$ was used. The greater than 75% proportion would make even more sense if they used the significance level $p < .05$.

Mawhinney et al. (1995, p. 234) wrote as well: "However, what is particularly interesting here is that in all cases where the MWW [Mann-Whitney-Wilcoxon] test was significant, the average student response was *higher* than the average employer response." That is, the student evaluated the items as being more important than the employers did. In the present thesis, the results were the reverse. In all five items where the difference between the students and the software developers was statistically significant, the means of the students were *smaller*. That is, the students evaluated them as being less important than the software developers did. These five items belonged to the category "Mathematics, physics, and theoretical computer science."

Kim et al. (1999, p. 513) reported that IS professionals and professors and lecturers perceived the importance of 12 items differently when the number of items was 30. However, this result was for the significance level $p < .05$ (p. 516). The number of non-agreed items decreased from 12 to four when the significance level $p < .01$ was used (p. 516). Thus, the proportion of non-agreed items was 13% whereas the equivalent proportion between the software developers and the professors and lecturers was 5% in the present thesis (2 out of 42 items, $p < .01$). They did not report correlation coefficients between the means but based on the low proportion of non-agreed items, their respondent groups apparently agreed quite well on the importance of various items. In that respect, their research was similar to the results of the present thesis.

Based on the results of the present thesis, the correspondence between typical degree requirements in computer science programs and required skills in software developer positions was quite good. Based on Knapp's (1993) results as well, the correspondence between industry and education was good. Thus, no relevant triangulation was found where the correspondence between education and the importance of skills was poor.

21.1.2 Trend analysis of job advertisements

The results of the trend analysis of job advertisements are compared with those of previous longitudinal research. The comparison is divided into subsections according to the type of results that are compared. The order is as follows: the number of skills, programming languages, operating systems, database skills, networking skills, and distributed skills. Finally, it is assessed if the results correspond in general or not.

Number of skills

The means of the number of required skills from the year 1970 to 2004 according to the previous longitudinal research and the present thesis are presented in Table 42. The results are not exactly from the years in question but nearby results were used as well in order to reasonable compare the results. The results are divided into two parts: the research targeted at software developer positions and the research targeted at all IT positions.

It can be noticed from the table that generally the results correspond well. One can assume that technical skills are required in software developer positions more often than in IT positions on average. The results are in line with this assumption. Moreover, all of the research shows increasing trends.

It was not reported in the previous research whether the differences of the means were statistically significant. It was not possible to calculate this either because the standard deviations were not reported. However, the sizes of the subsamples of Maier et al. (1998) and Gallivan et al. (2004) were so large (n = 424, … , 2,045) that it is reasonable to assume that the differences were statistically significant. The subsamples of Todd et al. (1995) were smaller (n = 48, … , 171) but even in this case, the difference between the years 1970 and 1990 is so great that it is likely to be statistically significant.

Table 42. Means of number of required skills from 1970 to 2004 according to various bodies of research.

Source	70	75	80	85	90	95	00	04
Targeted at developer positions:								
Todd et al. (1995, p. 6)	2.2	3.4	2.8	3.7	4.2	—	—	—
The present thesis, Computerworld	—	—	—	—	3.6	5.3	7.2	7.7
The present thesis, Dice	—	—	—	—	—	—	—	6.0
Targeted at all IT positions:								
Maier et al. (1998, p. 38)	—	—	2.6	3.1	3.3	3.5	—	—
Gallivan et al. (2004, p. 75)	—	—	—	—	3.0	3.5	4.2	—

Note. Dash (—) indicates that the mean was not obtained. Some means are approximately from the year in question, estimated from a figure and less accurate, or pooled from two means of the nearby years. For example, the mean 3.0 of Gallivan et al. in 1990 is from the year 1988.

Programming languages

Gallivan et al. (2004) analyzed job advertisements from the period 1988–2001. They (p. 76) reported that the proportions of the category "Programming languages" were 43% in 1988, 36% in 1995, and 34% in 2001. Thus, there was a decreasing trend. This is exactly the opposite of the results of the present thesis where the proportions increased from 49% in 1990 to 71% in 2004. A partial explanation is that the Gallivan et al. sample included all types of IT positions and the proportion of the category "Programmer/Analyst" decreased from 64% in 1988 to 24% in 2001 (p. 74). During the same period, the proportion of the combined categories "Software Engineer" and "Web developer" increased approximately 19% but this does not compensate for the strong decrease of 40% in programmer/analyst positions. Another partial explanation is that apparently Gallivan et al. counted proportions in a different manner than the author of the present thesis. This explanation is considered later in subsection "General comparison of trend analyses."

The other previous research results were not suitable for comparing the category "Programming language" because previously, for example, the categories "2GL" and "3GL" were used.

The report by Salary Services (2004a, pp. 224–229) presented the numbers of positions for each individual skill but no proportions. It was not possible to count the proportions because the subsample sizes were not reported for the years 1999–2002, but the author of the present thesis counted the ranks for 150 individual skills for the years 1999–2003 from the results that were presented in the report. In addition, the ranks for the ten most common skills from the third quarter of 2004 that were presented on the web page (Salary Services, 2004b) were used. The ranks are presented in Table 43. The columns of skills are ordered according to the ranks so that the smallest rank of 2004 is on the left. One should note that in 2004 the maximum rank is only ten.

Table 43. Ranks of most common programming languages in the UK.

Year	Java	C++	C	Visual Basic	C#	Perl	Cobol
1999	8	1	7	5	—	—	10
2000	4	1	13	5	150	17	46
2001	6	1	7	5	128	23	34
2002	8	2	5	7	67	29	55
2003	5	3	6	9	27	30	75
2004	2	5	7	8	—	—	—

Note. Sources: Salary Services (2004a; 2004b). Dash (—) means that the rank was not reported.

It can be noticed from Table 43 that C++ lost its position as the most often required programming language to Java but C and C++ are still popular languages. The need for C# has increased and the need for Cobol has decreased. These results correspond well with the results of the present research because

according to the supplementary report (Surakka, 2005c, p. 16), Java, Visual Basic, C, and C++ were important languages in 2000–2004, the proportions of Cobol decreased strongly, and (ibid., p. A/2) the proportions of C# increased and the proportion of Perl stayed the same.

Operating systems

Todd et al. (1995, p. 9) used the skill category Operating systems. They did not report proportion for this category, but they reported the numbers of phrases and the sample sizes. Based on these results, the proportions of the category "Operating systems" have apparently increased during the period 1970–1990. It was not possible to count proportions in the same way as in the present research but however, this increase is in line with the results of the present thesis.

Gallivan et al. (2004, p. 76) reported that the proportions of the category Operating Systems were 26% in 1988, 23% in 1995, and 14% in 2001. This decreasing trend is contrary to that in the present thesis where the proportions increased from 28% in 1990 to 52% in 2004. As was explained previously, an explanation could be that in their study the sample included all IS positions, the proportions of programmer/analyst decreased strongly, and they counted proportions in a different manner to that used in the present thesis. The last explanation is considered later in the subsection "General comparison of trend analyses."

According to the results of Cheney et al. (1990), the importance of Operating Systems was evaluated or forecasted to decrease from 1978 to 1987 and from 1987 to 1995. Thus, their result was conflicting with the results of Todd et al. and with the results of the present thesis. However, the Cheney et al. research was not a job advertisement analysis but a focus group study.

Similarly as for programming languages, in Table 44 are presented ranks from the analyses of Salary Services Ltd. Based on these results, the relative need for various Unix and Windows operating systems has stayed at the same level whereas the need for mainframe and midrange operating systems MVS and OS/400 has decreased. Macintosh was not among the TOP150 skills.

Table 44. Ranks of most common operating systems in the UK.

Year	Unix	Windows NT	Windows 2000	Solaris	Linux	AIX	Windows	HP-UX	MVS	OS/400
1999	4	3	—	32	—	34	9	—	18	—
2000	2	8	90	15	35	56	34	73	79	135
2001	2	8	33	14	32	43	40	70	69	135
2002	3	6	20	17	28	50	53	82	90	141
2003	2	8	13	21	24	50	59	91	128	146
2004	4	—	—	—	—	—	—	—	—	—

Note. Sources: Salary Services (2004a; 2004b). Dash (—) means that the rank was not reported.

The Salary Services results are partly difficult to compare with the results of the present thesis because the results of Table 44 are mostly for individual skills, not for similar categories that were used in the present thesis. The results correspond well with the present thesis in the respect that, according to the supplementary report (Surakka, 2005c, p. A/2), the proportion of the category "Mainframe or midrange" decreased from 38% to 21% and the proportion of Macintosh remained at 0% during the 2000–2004 period. The comparison of the categories "DOS or Windows" and "Unix" of the present research was not reasonable because the relevant results are divided into several columns in Table 44. For example, the table has several items for different Unix vendors.

Database skills

Todd et al. (1995, p. 9) did not report proportions of the category Database, but they reported the numbers of phrases and the sample sizes. Based on these results, the proportions of the category Database have apparently increased during the period 1970–1990. It was not possible to count proportions in the same way as in the present research but however, this increase is in line with the results of the present thesis.

Athey and Plotnicki (1998, p. 76) reported that over 70% of all job opportunities required some knowledge of relational database technology. However, they did not report from which year this proportion was; their period was 1989–1996. They reported increasing trends in individual database skills.

Also Trower (1995, p. 599) reported an increasing trend in the category Relational DB. According to Cheney et al. (1990), the importance of DBMS was forecasted to increase from 1987 to 1995.

The results of Salary Services Ltd. are presented in Table 45. It can be noticed from the table that database skills are generally important because SQL and Oracle have remained among the TOP10 skills during the whole period. Based on these results, the order among database vendors is similar to that of the present research. However, it can be noticed that SQL has apparently been required more often than according to the results of the present research.

Table 45. Ranks of most common database skills in the UK.

Year	SQL	Oracle	SQL Server	Access	Sybase	DB2
1999	6	2	—	23	11	16
2000	3	6	10	21	20	52
2001	3	4	12	18	20	39
2002	1	4	10	18	23	48
2003	1	4	10	20	29	51
2004	1	6	9	—	—	—

Note. Sources: Salary Services (2004a; 2004b). Dash (—) means that the rank was not reported.

Networking skills

Gallivan et al. (2004, p. 76) reported that the proportion of the category Networks/Communications were 20% in 1988, 22% in 1995, and 34% in 2001. These results do not correspond with the result of the present thesis because according to the results of the present thesis there was some increase during the first half of the 1990s but after 1996 the proportions have stayed at the same level or decreased. This difference cannot be explained easily by the differences in job positions because the Gallivan et al. proportions of the category Network Design were only 1% in 1988 and 5% in 2001. A more plausible explanation is that they counted proportions in a different manner to that the author of the present thesis. This is considered later in the subsection "General comparison of trend analyses."

Trower (1995, p. 599) reported that the number of advertisements of the category Network increased during the period 1990–95 faster than the index did. This corresponded with the results of the present research.

The other previous research studies had no suitable categories or results for comparison. However, Athey and Plotnicki (1998, p. 84) wrote: "Certainly networking is becoming more widespread. However, the number of advertised job opportunities in this area are surprisingly small." According to Cheney et al. (1990), the importance of "Telecommunications concepts" was forecasted to increase from 1987 to 1995.

The results of Salary Services Ltd. are presented in Table 46. No general trend could be observed because the ranks of some networking skills increased, some stayed at the same level, and some decreased. It can also be noted that networking skills were rare among the TOP10 skills. This corresponds well with the results of the present thesis. According to the ranks, the most common networking skill was TCP/IP and CISCO was the second. The ranks of other networking skills varied so much or were so close to each other that it was hard to find the third common skill. Internet and phrases such as "network" were not apparently used as searched items in the analyses of Salary Services because they were not among the top 150 skills.

Table 46. Ranks of most common networking skills in the UK.

Year	TCP/IP	GSM	Novell	LAN	CISCO	WAN	Intranet
1999	14	—	45	22	—	24	—
2000	9	50	36	42	14	48	39
2001	9	35	36	54	15	66	42
2002	13	30	47	32	26	35	62
2003	11	48	43	35	25	36	60
2004	—	—	—	—	—	—	—

Note. Sources: Salary Services (2004a; 2004b). Dash (—) means that the rank was not reported.

Distributed technology skills

From the previous research studies, Athey and Plotnicki (1998), and Trower (1995) are the most suitable for comparing the category "Distributed technology." Trower (p. 599) reported that the number of advertisements in the category Client/Server increased from approximately 40 in 1990 to 220 in 1995. It was not possible to count the proportions because the sizes of the subsamples were not reported. However, Trower reported an index of all job advertisements as well. The numbers of the category Client/Server increased a lot faster than the index did. In any case, a more serious problem for comparison was that his coding principles for the client/server skills were problematic. He wrote (p. 598): "Finally, the leading client/server skills mentioned in the 1995 want ads are Windows (88 ads) and OS/2 (37 ads) for the GUI front-ends to client/server applications." It is author's opinion that Windows and OS/2 should be classified as operating systems skills, not as distributed technology skills.[8]

Athey and Plotnicki (1998) reported that the proportion of Client Server was 6.9% in 1993 and 7.4% in 1996. The proportion of 1993 corresponded well enough with the results of the present research but their result for the year 1996 was a lot less than the result of the present research. They did not report proportions of Client Server for the subsamples of the years 1989 and 1992.

The results of Salary Services Ltd. are presented in Table 47. Based on the ranks, these skills were required more often than previously, which corresponds well with the results of the present thesis. One difference was that the Salary Services (2004a) report did not include any results about J2EE. Probably this item was missing from the coding scheme because it is hard to believe that J2EE was not among top 150 skills in 2003. Anyhow, the rank of .NET has decreased so strongly that it is reasonable to assume that Microsoft's technologies for distributed systems are required more commonly in the UK than Sun's technologies.

Table 47. Ranks of most common distributed technology skills in the UK.

Year	.NET	ASP	JSP	IIS	EJB	CORBA
1999	—	—	—	—	—	—
2000	149	19	66	53	44	24
2001	99	21	48	62	37	41
2002	31	16	45	46	54	65
2003	17	14	37	39	47	69
2004	10	—	—	—	—	—

Note. Sources: Salary Services (2004a; 2004b). Dash (—) means that the rank was not reported.

[8] It is understandable if Trower had problems in classifying distributed technologies. Even nine years later, the author of the present thesis had most problems with this category. The data about distributed technology skills were recoded four or five times because progressive changes to the coding principles were made.

General comparison of trend analyses

Generally, the results of the trend analyses other than those of Gallivan et al. (2004) corresponded satisfactorily or well with the present thesis. The results of Gallivan et al. were conflicting with those of the present thesis in three major categories: programming languages, operating systems, and networking skills. However, even they (p. 75) reported that the mean of technical skills increased between the years 1988 and 2001.

A partial explanation is that Gallivan et al. counted proportions in a different manner to that than in the present thesis. They wrote (p. 75): "For instance, if C++ and C were mentioned in an ad for one job position, we counted two programming languages for this position." In the present thesis, the equivalent criterion was "at least one programming language" that was counted separately for each position. In addition, they counted the proportions relative to *the number of all skills*, not relative to *the number of positions* as in the present thesis. This was not explained in the body text but one can deduce it from Table 6 of their paper. For example, the proportion of programming languages in 1988 was counted as the relation 1,397 / 3,250, which equals 43%, where 1,397 is the number of programming language skills and 3,250 is the number of all skills (p. 76).

21.1.3 Questionnaires targeted at software developers

First, the results of the Delphi study targeted at the software developers are compared with Lethbridge's (2000) results because Lethbridge's research was the most relevant. Second, the other questionnaires targeted at software developers are compared with the present thesis.

Lethbridge's research

Lethbridge's results were compared only with the results of the Delphi study targeted at software developers (Section 7) because these two respondent groups were similar. The means are presented in Table 48. Lethbridge's scale of 0–5 was converted to a scale of 1–4 to enable comparison. In some cases, two or three of Lethbridge's items corresponded to one item of the present research. In these cases, Lethbridge's means were pooled. The details of the conversion are presented in Appendix E. In Table 48, a dash (—) indicates that Lethbridge's survey did not include a corresponding item.

The values of T_1 of the Mann-Whitney test between two sets of research results are presented in the column Mann-Whitney. Two asterisks (**) indicate that the difference between two sets of research results was statistically significant ($p < .01$). The confidence level $1 - \alpha = .99$ was used to avoid Type I errors because the number of items was so large. The rows were divided into

the four categories that were used in the questionnaire of the present research. Within each category, the rows are first ordered according to the result of the Mann-Whitney test and then according to the name of the subject or skill.

In general, Lethbridge's means were a little smaller than the means of the present research across the range of items. Lethbridge's converted mean was 2.6 ($N = 9{,}854$) and the mean of the present research 2.9 ($N = 460$). A possible explanation is that different answering scales and questions were used. The question of the present research was "How important do you think the following subjects and skills are for demanding programming tasks?" whereas Lethbridge asked about the usefulness of specific material in a respondent's career; that is, during their entire career.

Perhaps the most important recommendation in Lethbridge's paper concerned natural science and continuous mathematics. He did not use statistical tests to analyze the results but the means for the usefulness of the following items were as follows (scale: 0 = Completely Useless, … , 5 = Essential): Physics 1.6, Differential and Integral Calculus 1.3, Laplace and Fourier Transforms 1.3, Differential Equations 1.1, and Chemistry 0.9 (Lethbridge, 1999, p. 34). These means are so low that the differences would likely be statistically significant if they were compared with answers across all the 75 items of his questionnaire. The respondents of the present research evaluated the importance of physics and continuous mathematics as being very or quite low as well and the differences were statistically significant ($p < .01$) when compared with the answers across all the items (Table 11 in Section 7.2.2). Thus, the present research confirmed Lethbridge's results on physics and continuous mathematics. However, the present thesis cannot confirm the result on chemistry because it was not included in the questionnaire.

Table 48. Importance of various subjects and skills. Means of present research, converted Lethbridge's means, and results of Mann-Whitney test. Scale: 1 = Not at all important, … , 4 = Very important.

Subject or skill	Present research	Lethbridge's research	Mann-Whitney
Mathematics, physics, and theoretical comp. science:			
Other areas of theoretical comp. science (e.g., automata)	3.3	2.3	3.5**
Discrete mathematics	2.6	1.9	2.9**
Logic (in particular, propositional and predicate logic)	2.8	2.3	1.9
Mathematics for continuous systems	2.0	1.7	1.4
Physics	1.6	2.0	-1.7
More technical or part of the operational system:			
Data structures and algorithms	3.8	3.1	3.4**
Computer/data security	3.2	2.3	3.1**
Distributed systems	3.1	2.4	2.9**
Object-oriented programming	3.6	3.0	2.6**
Compilers	3.1	2.4	2.4
Internet protocols	3.4	2.9	2.0
Computer architecture	3.0	2.6	1.9
Systems programming	3.2	2.8	1.9
Operating systems	3.3	3.0	1.6
Software architectures	3.5	3.1	1.6
Real-time systems	2.6	2.6	0.3
Artificial intelligence and knowledge engineering	1.6	1.8	-0.1
Implementing techniques of user interfaces	2.7	3.0	-1.0
Computer graphics	1.9	2.2	-1.1
Database management systems	2.7	3.0	-1.1
Telecommunications techniques other than Internet prot.	2.0	2.5	-2.1
Concurrent programming	3.1	—	—
Embedded systems	2.5	—	—
Extensible Markup Language (XML) techniques	2.7	—	—
Functional programming	2.6	—	—
Implementing techniques of WWW systems	2.7	—	—
Logic programming	2.3	—	—
Procedural programming	3.8	—	—
Script programming	3.4	—	—
Software engineering (different phases of life cycle):			
Design	3.7	3.1	2.7**
Requirements	3.6	3.1	2.7**
Test	3.5	3.0	2.4
Operation and maintenance	2.5	2.7	-1.1
Approval	2.6	—	—
Concept exploration	3.0	—	—
Implementation	3.7	—	—
Installation and checkout	2.3	—	—
Packaging and delivery	1.9	—	—
Retirement	1.8	—	—
Software engineering (possible in several phases):			
Version and configuration management	3.6	3.0	3.0**
Project management	3.2	3.0	1.0
Documenting	3.0	3.1	-0.4

Note. Dash (—) means that no corresponding Lethbridge's item was available. **$p < .01$.

Next, the items where the differences are statistically significant ($p < .01$) are discussed. The same difference in the questions could also explain the statistically significant differences between the following items: data structures and algorithms, design, discrete mathematics, other areas of theoretical computer science (e.g., automata), requirements, and "Version and configuration management." In all these items the respondents of the present research evaluated them as being more important than Lethbridge's respondents. It is possible that the respondents of the present research evaluated discrete mathematics and theoretical computer science as being more important because they thought that theoretical tasks were often more demanding as well. The importance of the items design, requirements, and "Version and configuration management" is probably greater in larger or in some other way more demanding projects. The author of the present thesis does not have a clear explanation for the difference in data structures and algorithms. However, his assumption is that in more demanding projects it is important to, on the one hand, be aware of several different data structures and algorithms, and on the other understand the tradeoffs arising from efficiency.

Lethbridge's survey was conducted in 1998. After 1998, the use of the WWW has increased. The number of web sites was approximately three million in 1998 and nine million in 2002 (OCLC Online Computer Library Center, n.d.). The statistically significant difference in computer/data security was probably related to the increased use of the WWW. In this item, Lethbridge's mean was smaller than the mean of the present research. One could argue that the greater importance of computer/data security was partly a consequence of the terrorist attacks in the USA on September 11, 2001. This could be a possible explanation in the USA but in Finland the most likely explanation was the increased use of the WWW. Another explanation for the difference was that in Finland, telecommunications companies were such big employers that this could have an effect on the answers. Forty-five percent of the respondents of the present research worked for telecommunications companies when in Lethbridge's research the proportion was 14%.

In addition, the respondents of the present research evaluated object-oriented programming as being more important than Lethbridge's respondents did. Based on Lethbridge's results and the content analysis of job advertisements (e.g., Gallivan et al., 2004, p. 77), object-oriented programming was already very or at least relatively important approximately five years ago. The following are probably explanations for the increased importance of object-oriented programming: (a) the complexity of modern software systems, (b) the evolution of object-oriented languages and tools, and (c) the growth of the WWW and the use of Java in WWW applications.

Finally, it is evaluated whether Lethbridge's means and the means of the present research correlated. The same means as in the previous table are presented in Figure 13. Only those means are presented where Lethbridge's

value was available as well. It can be noticed from the figure that no Lethbridge's mean was greater than 3.25 but in the present research there were several such means. The Spearman rank correlation coefficient r_s of the means was 0.72, which indicates that the means correlated positively. In addition, it was calculated if the correlation was statistically significant. From three options, the upper-tailed test for positive correlation was selected. The confidence level used was $1 - \alpha = .999$ and the sample size was 28. The value of $w_{0.999}$ was 0.57 (Conover, 1999, p. 542), which was smaller than the Spearman rank correlation coefficient 0.72. Thus, the positive correlation was statistically very significant ($p < .001$).

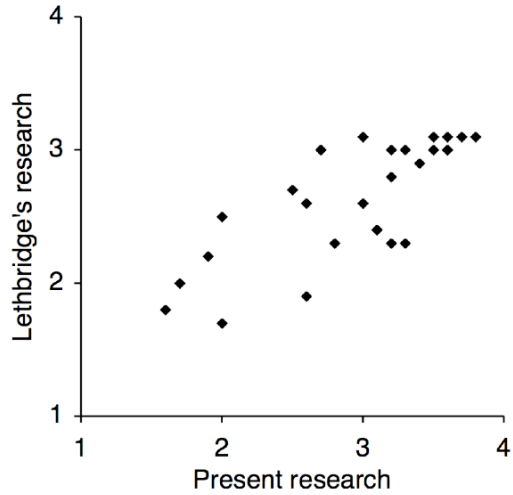

Figure 13. Lethbridge's converted means and means of present research. Scale: 1 = Not at all important, … , 4 = Very important.

Other questionnaires targeted at software developers

According to Bailey and Stefanik's (2001) results, the three most important technical skills were "Ability to read, understand and modify programs written by others", "Ability to code programs," and "Ability to debug software." The item "Ability to code programs" corresponds well with the results of the present thesis because the items Object-oriented programming, Procedural programming, and Implementation were evaluated as being important. Their items "Ability to read, understand and modify programs written by others" and "Ability to debug software" were so different that comparison was not possible. The least important technical skill in their research was "Knowledge of RPG." This item was too different for comparison purposes.

According to the results of Beise et al. (1991, p. 20) and those of Haywood and Madden (2000), programming skills were important. This corresponds well with the results of the present thesis as well.

21.1.4 Content analysis of degree requirements

Next, some results of the content analysis of degree requirements are compared with that of U.S. News (2004), the findings of McCauley and Manaris (2002), and recommendations of Computing Curricula 2001 (Engel & Roberts, 2001). Based on the results of the present research, the four most common specializations were Computer Systems, Theoretical Computer Science, Software Systems, and Artificial Intelligence. U.S. News (2004, p. 73) presented the separate TOP10 ranking lists of graduate programs for three specializations: "Artificial Intelligence," "Systems," and "Theory." Apparently these three specializations were selected because they were common in graduate programs. This selection corresponded well with the four most common specialization names of the present research.

The most common courses of the specializations in Software Systems were presented previously in Table 8 of Section 5.2.3. These results are repeated in Table 50. The courses are ordered first according to the column Undergraduate and then according to the column Graduate. The column Undergraduate refers to the undergraduate programs and Graduate to the graduate programs of the present research. In addition, the table includes the new column Required where the results from the survey by McCauley and Manaris (2002, p. 4) are presented. They asked which upper-level courses were required during the academic year 2001–2002. Their sample was 45 accredited undergraduate computer science programs in the USA.

Table 49. Proportions (%) of most common courses of specializations in Software Systems and how often courses were required in accredited undergraduate programs.

Course	**Undergraduate** (n = 10)	**Graduate** (n = 18)	**Required** (N = 45)[a]
Computer Networks	70	44	18
Compilers	60	61	16
Databases	60	61	31
Operating Systems	60	44	96
Programming Languages	40	61	87
Software Engineering	40	22	76
Computer Architecture	40	17	69
Computer Graphics	40	17	0
Distributed Systems	30	50	—
Advanced Operating Systems	20	50	—

Note. Dash (—) indicates that the proportion was not reported.

[a]Source: McCauley & Manaris (2002, p. 4).

The results of the present research were at odds with the results of McCauley and Manaris. It is possible that the three uppermost courses were required or elective often in the specializations in Software Systems because, according to McCauley and Manaris, they were not so commonly required in the undergraduate programs. However, it was not possible or did not make sense that the course Operating Systems would be required in almost every undergraduate program but still offered in 60% of the specializations in Software Systems. The same problem concerned the courses Programming Languages, Software Engineering, and Computer Architecture.

In addition, the results of the present research are conflicting with the recommendations of Computing Curricula 2001 (Engel & Roberts, 2001, p. 47). The curriculum model for a research university in the USA included the following intermediate courses: Computer Architecture, Operating Systems, Net-Centric Computing, Information and Knowledge Management, and Software Development. For example, it does not make sense that the course Operating Systems would be offered in 60% of the specializations in Software Systems if the programs followed the recommendations of Computing Curricula 2001. This problem is similar to the conflict with the results of the present research and the survey by McCauley and Manaris (2002).

This is a somewhat serious question that would require more detailed comparisons of the present research and McCauley and Manaris' research. That is, the data should be compared, not just the aggregated results. A possible explanation for the differences is that the sampling principles of the present research and McCauley and Manaris' survey were different. For example, only 39% of the undergraduate programs of the present research were accredited whereas all programs in McCauley and Manaris' survey were accredited.

However, such a detailed comparison was not conducted because the results of Table 8 corresponded well enough with the other results of the present thesis. Here, only the results of the column Undergraduate of Table 8 were compared because McCauley and Manaris' survey and Computing Curricula 2001 were targeted at undergraduate programs. The summarized results of the whole thesis are presented later in Section 20. The greatest difference was that according to the content analysis of degree requirements, the course Computer Networks was very common (the proportion was 70%) whereas in the summarized results, the item "Internet protocols" can be classified as being somewhat important but not as being very important. The courses Compilers, Databases, and Operating Systems were commonly required or offered in the specialization in Software Systems as well but this corresponds very well with the summarized results of the present thesis.

Finally, the results of the present research on prerequisites are compared. These results were presented previously in Figure 4 of Section 5.2.4. In Computing Curricula 2001 (Engel & Roberts, 2001, p. 44), advanced courses

were listed. However, it was not possible to analyze prerequisites because the full course descriptions were not provided. They wrote: "Instead, we plan to create web pages for these courses, which will be accessible from the CC2001 web page." Unfortunately, no such web pages were found. Nonetheless, the Computing Curricula 2001 listed the following advanced courses: Advanced Operating Systems, Compiler Construction, and Distributed Systems. This corresponds well with the results of the present research. However, also the course Software Engineering was listed as an advanced course whereas according to the results of the present research, it was rather an intermediate course. In addition, in Computing Curricula 2001 the course Automata and Language Theory was listed as an advanced course whereas in the present thesis the similar course Automata and Formal Languages was classified as an introductory course.

21.1.5 Delphi study targeted at professors and lecturers

Next, the results of the Delphi study targeted at the professors and lecturers are compared with the results of Kim et al. (1999). They used the following scale: 1 = Least important, … , 7 = Most important. Their questionnaire had seven items that were close enough for comparison. The means of these items were (p. 516): Telecommunications and networking 5.5, Software engineering and maintenance 5.2, Managing data resources 5.1, Client/server computing 5.0, Developing and maintaining distributed systems 5.0, and Improving information security and control 4.9, and Internet and electronic commerce 4.2. Their results did not correspond well with those of the present research because, for example, the respondents of the present research evaluated distributed systems as considerably more important than telecommunications and networking. As a general comment, the differences between their means were slight. Simply put, their respondents evaluated all these skills as being quite important whereas in the present research, it was clearer which technical skills were evaluated as being more important than others.

21.1.6 Cross-sectional analysis of job advertisements

The results of the cross-sectional job advertisement analysis of the year 2004 (Section 13) were compared with the results of the Information Technology Association of America [ITAA] (2002), Litecky and Arnett (2001), Prabhakar et al. (2004), and Salary Services (2004a; 2004b). The comparisons are divided into subsections in the following order: programming languages, platforms, databases, distributed technologies, differences between software developer positions, and differences between entry-level and senior-level positions. Networking skills are not compared because the cross-sectional analysis of the present thesis had no results on individual networking skills.

Programming languages

The results of the five most common languages according to the present thesis are presented in Table 50. The rows are ordered so that the results of the analyses targeted at software developer positions are presented first, followed by the results of the analyses targeted at all IT positions. The columns are ordered according to the results of the present thesis so that the greatest proportion is on the left.

The results of Salary Services (2004a) from the last quarter of the year 2003 were sufficiently detailed to make it possible to count the proportions of individual skills for a similar subsample to that in the present thesis. In addition, the results of the CD-ROM that was provided with the report were used. The report and CD-ROM listed results for approximately 50 job titles. The author of the present thesis counted proportions using the following job titles: Analyst Programmer, Graduate developer/analyst programmer, Programmer, Senior programmer, Senior software engineer, Senior systems developer, Software engineer, and Systems developer.

The row "Litecky, personal communication" refers to the slides of Prabhakar, Arnett, and Litecky's conference presentation (C. Litecky, personal communication, December 8, 2004). The sampling period was apparently September 2004.

Table 50. Proportions (%) of five programming languages according to five job advertisement analyses.

Analysis	Java	C++	C	Visual Basic	C#
Targeted at software developers:					
Present thesis	35	31	23	15	9
Salary Services (2004a)	27	32	24	20	9
Targeted at all IT positions:					
ITAA (2002, pp. 45–46)	11	19	—	4	—
Litecky & Arnett (2001)	—	12	7	11	—
Litecky, personal communication	15	—	—	8	—

Note. Dash (—) means that the proportion was not reported.

It can be seen from the table that the results corresponded quite well. As one can expect, the proportions in the analyses targeted at all IT positions were smaller than in the analyses targeted at software developer positions. The most important single difference was that according to the present thesis, Java was the most commonly required programming language when according to Salary Services and the ITAA it was C++. The fact that the ITAA's analysis is from the year 2002 can explain the difference between the ITAA's analysis and the present thesis. However, the author did not find any obvious explanation for the difference between Salary Services and the present thesis. A partial explanation

could be, as will be shown later in Table 51, that Unix appears to be a more common platform and Windows a less common platform in the USA than in the UK. One could assume that Java is used in Unix often and Visual Basic in Windows.

Another possible explanation is that there just are considerable differences between the countries. This is possible because according to Athey and Plotnicki (1992; 1998), there were large differences between ten American cities. For example, the proportions for C varied from 9% in Los Angeles to 31% in San Jose (1992, p. 52). The explanation for these differences would require detailed knowledge of the economics and industrial structure of the countries. For example, Athey and Plotnicki (1992, p. 52) wrote: "Not surprisingly, San Jose, home of many PC hardware and software development companies, was the only city to have a higher percentage of advertisements for a language other than COBOL."

Platforms

The proportions of different platforms are presented in Table 51. The results of Salary Services were counted in a similar manner to that in the present thesis but the results of the other analyses are typically for an individual operating systems skill. For example, the ITAA's proportion for Unix is for "Unix Solaris," which was the most common Unix operating system, not for the category Unix where all different Unix operating systems would be combined. This explains partly why the proportions of the analyses targeted at all IT positions are so much smaller.

According to Salary Services, the proportion of mainframe/midrange was considerably smaller in the UK than in the USA. The author of the present thesis does not have an explanation for this difference but it is possible that there are large differences between the countries because, as was mentioned previously, there were large differences between ten American cities. For example, the proportion of VAX/VMS varied from 6% in San Francisco to 32% in Boston (Athey & Plotnicki, 1992, p. 53). They (ibid., p. 53) explained: "Not surprisingly, with DEC headquartered in neighboring Maynard, VAX has the strongest presence in the Boston area." One could ask if the proportion of mainframe/midrange skills is greater in the USA than in the UK because the proportion of very large companies, government agencies such as the Federal Bureau of Investigation, and other organizations such as American Red Cross might be greater in the USA than on average.

Table 51. Proportions (%) of platforms according to five job advertisement analyses.

Analysis	Windows	Unix	Main-frame/ midrange	Cross-platform	Mac-intosh
Targeted at software developers:					
Present thesis	42	29	23	17	0
Salary Services (2004a)	55	21	5	—	—
Targeted at all IT positions:					
ITAA (2002, pp. 46–47)	11	3	4	—	0
Litecky & Arnett (2001)	12	17	7	—	—
Litecky, personal communication	12	17	—	—	—

Note. Dash (—) means that the proportion was not reported.

Database skills

The most common database skills are presented in Table 52. In the row "Present thesis, Surakka (2005c)" the proportion of SQL was not presented previously in Section 13.2 but is from the previous report (Surakka, 2005c, p. 24).

Generally, the results corresponded well because according to all analyses the most common database vendor was Oracle. The biggest difference was that according to the results of Salary Services, SQL was required in the UK much more often than in the USA. The author of the present thesis does not have an explanation for this difference but, as explained earlier, it is possible that there are large inter-country differences. Athey and Plotnicki (1992; 1998) did not report the proportions of database skills for different cities but there were at least some differences because they (1998, p. 76) wrote: "When demands for database skills were analyzed by city, Oracle was first or tied for first in seven of the ten cities. In Los Angeles and San Jose, general SQL knowledge was the most prevalent requested skill. While in San Jose, most database job opportunities were for Sybase."

Table 52. Proportions (%) of five database skills according to five job advertisement analyses.

Analysis	Oracle	SQL	SQL Server	DB2	Sybase
Targeted at software developers:					
Present thesis, Surakka (2005c, p. 24)	22	19	11	7	5
Salary Services (2004a)	16	31	14	1	5
Targeted at all IT positions:					
ITAA (2002, p. 46)	14	14	4	4	3
Litecky & Arnett (2001)	12	11	—	—	—
Litecky, personal communication	15	15	9	3	—

Note. Dash (—) means that the proportion was not reported.

Distributed technologies

The most common distributed technology skills are presented in Table 53. Unlike the previous three tables, Table 53 has no rows for the Information Technology Association of America [ITAA] (2002) and Litecky and Arnett (2001) because they did include suitable results. The results of the present thesis and those of Salary Services corresponded well. According to Salary Services, Microsoft's distributed technologies were required more often than Sun's. However, Salary Services apparently did not use the category J2EE that was the most often mentioned as Sun's distributed technology skill in the present thesis.

In addition, Prabhakar, Arnett, and Litecky (C. Litecky, personal communication, December 8, 2004) reported the following proportions: Web Programming 25.3% and Client/Server or "C/S" 4.4%. These results corresponded quite well with the results of the present thesis because their results were for all IT positions. However, based on these results it was not possible to conclude if Microsoft's technologies were required more often than Sun's because it was possible that the category "Web Programming" referred to both Microsoft's and Sun's technologies.

Table 53. Proportions (%) of six distributed technology skills according to three job advertisement analyses.

Analysis	.NET	ASP	J2EE	JSP	Web-Logic	Web-Spere
Targeted at software developers:						
Present thesis	19	18	13	8	5	5
Salary Services (2004a)	11	11	—	5	—	—
Targeted at all IT positions:						
Litecky, personal communication	12	—	—	—	—	—

Note. Dash (—) means that the proportion was not reported.

Differences between software developer positions

The proportions of selected low-level programming skills are presented in Table 54 where the proportions of the USA are the results of the present thesis. The author of the present thesis counted the proportions of the UK using data from the Salary Services (2004a) report. The report did not include the job title Software developer and the closest alternative was Systems developer. The rows are ordered according to the names of the skills. It was not calculated if the differences in the Salary Services results were statistically significant but probably the differences between software engineers and two other job positions were statistically significant because the sample sizes were so large.

Table 54. Proportions (%) of some low-level programming skills for selected job titles in the USA and in the UK.

Skill	Programmer		Software/Systems developer		Software engineer	
	USA	UK[a]	USA	UK[a]	USA	UK[a]
Assembler	1	1	4	0	14	6
C	16	15	30	20	40	44
C++	22	24	48	28	50	43
"embedded"	1	4	5	3	15	36

[a]Source: Salary Services Ltd. (2004a).

The proportions of Salary Services corresponded well with the results of the present thesis because according to both analyses, the low-level programming skills were more common in software engineer positions. In particular, the phrase "embedded" was mentioned in the British job advertisements targeted at software engineering positions even more often than in the USA (36% and 15%, respectively.)

Litecky and Arnett's (2001) paper, the ITAA's report (Information Technology Association of America, 2002), and the slides of Prabhakar, Arnett, and Litecky (C. Litecky, personal communication, December 8, 2004) were not suitable for comparison because they did not present results for different job titles.

Entry-level versus senior-level positions

Salary Services (2004a) contained the job title Graduate developer/analyst programmer that (p. 293) "is used to classify all programmers & developers who essentially have less than six months commercial experience on programming." In addition, the report contained the job titles Senior programmer, Senior software engineer, and Senior systems developer. It was not possible to compare the groups using categories such as "at least one common database skill" because the report did not include suitable results. However, from the results presented in the report, the author of the present thesis counted the proportions of individual technical skills and these two groups were compared with each other. For brevity, all results are not presented but only some selected results are presented in Table 55. The rows are ordered according to the difference in the proportions.

Table 55. Proportions (%) of some selected skills and difference of these proportions in entry-level ($n = 375$) and senior-level ($n = 3{,}001$) positions in the UK.

Skill	Entry-level positions	Senior-level positions	Difference
C	15	30	15
C++	33	44	11
Java	28	32	4
Unix	18	22	4
.NET	12	13	1
Oracle	14	13	-1
ASP	13	11	-2
SQL	41	27	-14

Note. Source: Salary Services Ltd. (2004a).

The proportions of senior-level positions were typically greater than for entry-level positions but the differences were small. As can be noticed from the table, there were even commonly required skills such as SQL that were mentioned in entry-level positions more often. In particular, the differences of individual distributed technology skills were so small that statistical tests were not even used to analyze the differences. Therefore, these results did not confirm the result of the present thesis that distributed technology skills were required more often in senior-level positions.

Litecky and Arnett's (2001) paper, the ITAA's report (Information Technology Association of America, 2002), and the slides of Prabhakar, Arnett, and Litecky (C. Litecky, personal communication, December 8, 2004) were not suitable for comparison because they did not present results for entry-level positions.

21.2 Evaluation of the thesis

Generally, the results between the different parts of the present thesis correlated or otherwise corresponded well. This is a good property when the triangulation is evaluated as a whole. It was not likely that many different respondent groups and research methods would give misleading results. In addition, the findings of the present thesis corresponded reasonably well with the findings of the previous publications.

From possible respondent groups, managers and directors were not used in the present thesis, which is a shortcoming. However, this decision was made deliberately because the job advertisement analyses were proposed as a substitute for the survey targeted at managers and directors.

The research methods used complemented each other as is proposed in the methodological triangulation. It was difficult to analyze the necessity of

certain subjects such as data structures and algorithms in the job advertisement analyses but easy when the questionnaires were used. The strength of the job advertisement analysis was the possibility to conduct a trend analysis. However, the trend analysis of the present thesis had severe problems because copies of job advertisements published in WWW recruiting services in the past were not freely available. Fortunately, it was possible to control the main findings of the trend analysis of the present thesis by comparing them with previous publications. In particular, the report of Salary Services Ltd. (2004a) was so detailed and sufficiently recent that is was very useful for controlling the results. This is presented later in Section 21.1.2.

The present thesis included eleven research areas. Out of these eleven, the following three were actually not needs assessments: the concept analysis of "software systems," the content analysis of American degree requirements, and the Delphi study on cognitive skills. Three research areas of Part V can be classified as needs assessments but the scope was in basic studies and not targeted at specialization in Software Systems. Thus, one can question whether these six bodies of research fit under the thesis title "Needs assessment of Software Systems graduates." The benefit from these parts was limited but nonetheless they complemented the thesis. By design, these six research areas were reported briefly and together they account for 15–20% of the pages of the entire thesis.

As mentioned already in the introduction of the present thesis, Young and Lee (1997, p. 5) found that internships were an important selection criterion of graduates. Also the results of the present thesis indicate that the students of the institution had a lot of internship or other working experience before graduation. The present thesis cannot answer the question "What properties of a student are important to get an internship?" because the present thesis was targeted at the period after graduation. This topic was not part of the literature search of the present thesis, either. The same technical skills might be important for an internship as for an entry-level position. However, employers might use other criteria as well.

Another limitation was that the present thesis was targeted only or mainly at technical skills and soft skills were investigated only a little. Project management was the only soft skill that was used in the questionnaires.

As a consequence of the background of the present thesis, the results might be more relevant to computer science programs where students enroll directly into the program, specializations are used in order to organize advanced courses, and the course system is mostly topic-based.

21.3 Conclusions

Some conclusions are drawn in the present subsection that is divided further into subsections according to the overall structure of the present thesis.

21.3.1 Concept analysis of "software systems"

Based on the results of the present research, the concepts "software system" and "software systems" can be interpreted at least in three different ways:

1. In plural form "software systems" as an area of education. This interpretation was the most suitable for the purposes of the present thesis.
2. In singular form "software system" as software that is proposed for a certain task.
3. As a synonym for the term "software." In this case, the concepts "software system" and "software systems" can be considered as redundant.

The concept "software systems" is not redundant because it is actually used as a name of a specialization at some American research universities. One can interpret "software" as *all* programs that are installed in a computer whereas "software system" refers to *specific* programs. At any rate, it is the author's opinion that "software system" or "software systems" are not important enough concepts that they should be defined in standards. One can understand the technical meaning (Option 2 in the previous list) well enough by reading the definitions of the terms "software" and "system" given in the IEEE standard (Institute of Electrical and Electronics Engineers, 1990).

What might cause problems is that the plural form "software systems" could have at least two different interpretations. During the thesis work, it was not noticed that the concept would be used widely even in academic studies and apparently it is used only a little or not at all in industry. Thus, it would probably be adequate if professors in charge of specializations in Software Systems knew about the different interpretations. However, the concept was common enough as a specialization name that it could be explained in study guides, teaching materials, handbooks, or encyclopedias.

21.3.2 Content analysis of degree requirements

Based on the results of the content analysis of American degree requirements:

- "Software Systems" or "Systems" were often used as the name of specialization in American research universities.
- The courses Computer Networks, Compilers, Databases, and Operating Systems were typical courses of the specializations in Software Systems of undergraduate programs.

- The courses Compilers, Databases, and Programming Languages were typical courses of the specializations in Software Systems of graduate programs.
- The course Data Structures and Algorithms was a central prerequisite for a specialization in Software Systems.
- The courses Compilers, Distributed Systems, and Advanced Operating Systems were more advanced than the other typical courses of a specialization in Software Systems.

21.3.3 Summative triangulation

According to the results of the present thesis, the following subjects were evaluated as being important: databases, data structures and algorithms, distributed systems, object-oriented programming, operating systems, procedural programming, and software architectures. These items got from four to six points in summative triangulation. From various software life cycle phases, design, implementation, and test were evaluated as being the most important. These items got three points in summative triangulation when all points were obtained from three questionnaires.

According to the results of the present thesis, mathematics for continuous systems and physics were evaluated as being not at all or only a little important. The means of these two subjects were less than 2.0 for all three respondent groups of the present thesis (scale: 1 = Not at all important, … , 4 = Very important). In addition, the subjects were not important according to the results of other used research methods. All other subjects and skills were evaluated as being somewhat important (mean at least 2.5) by at least one respondent group. From various software life cycle phases, “packing and delivery” and retirement were evaluated as being less important than the other phases.

Based on the results of the trend analysis of job advertisements of the present thesis, the results by Gallivan et al. (2004), Maier et al. (1998), and Todd et al. (1995), the conclusion is that the technical requirements for software development positions have changed during the past 35 years so that the number of individual skills required has increased on average. This change is so great and found by more than one researcher so that problems with sampling or other simple explanations are not plausible.

The duties of software developers have changed to become technically more versatile in the respect that typically it is no longer enough to have skills only in 1–2 programming languages. In particular, the results in Section 12.2.2 indicate that it has become more and more common that software developers are expected to have database and distributed technology skills as well.

One purpose of trend analysis is "to predict what will be likely to occur in the future" (Cohen et al., 2000, p. 175). However, no predictions of the future are presented because this was not an objective of the present thesis.

21.3.4 Cognitive skills

The cognitive skills reported in the present thesis (Section 10) can be divided into two main categories: skills associated with composition and skills associated with comprehension. The composition category obviously includes skills that are related to the mastery of the programming languages and environments used. Other important skills associate with having an inherent model of the goal in one's mind, designing interfaces and abstractions, mastering and developing one's own working process, for example. The comprehension category includes skills such as understanding the program as a whole and ability to notice isomorphisms with other known problems.

On a general level, the results confirm that different comprehension-related tasks are an important part of the cognitive skills of a software developer. Approximately 40% of the items mentioned by the respondents can be classified as comprehension-related tasks. Obviously, this is not a surprising result because according to the definition presented in the beginning of Section 2.8, cognitive skills enable human beings to comprehend information.

21.4 Recommendations

Some recommendations are presented in this section. The recommendations were considered only for university-level education.

21.4.1 Definitions of concepts "software system" and "software systems"

The author's definitions of the concepts "software system" and "software systems" are presented in this subsection. First, the definition of the concept "software system" is derived. The following working definition was presented previously in Section 4.2.3 and was derived by combining the definitions of the terms "software" and "system" of the Institute of Electrical and Electronics Engineers (1990):

> A collection of computer programs, procedures, and possibly associated documentation and data organized to accomplish a specific function or a set of functions.

One respondent of the Delphi study targeted at the professors and lecturers wrote: "What are 'procedures' in this definition (if not parts of a program)?" Indeed, one can wonder if the word "procedures" could be removed. According to the Institute of Electrical and Electronics Engineers (1990, p. 158), the definition of the term "procedure" is the following:

> **procedure.** (1) A course of action to be taken to perform a given task.
> (2) A written description of a course of action as in (1); for example, a documented test procedure.
> (3) A portion of a computer program that is named and that performs a specific action.

Based on definition (3), one can interpret that the word "procedures" is not necessary in the working definition because a procedure is a portion of a computer program. Therefore, the word "procedures" was removed from the author's definition of the concept "software system." The author's definition is the following when the formulation is according to Suonuuti (2001):

> **software system**
> collection of computer programs and possibly associated documentation and data organized to accomplish a specific function or a set of functions

Second, the author's definition of the concept "software systems" is presented. Based on Suonuuti (2001), definitions should be given or are typically given only in singular form. However, this instruction is not followed here because apparently the plural form "software systems" has two different meanings whereas the singular form has only one meaning. The definition is the following:

> **software systems**
> (1) collections of computer programs and possibly associated documentation and data when each collection is organized to accomplish a specific function or a set of functions
> (2) area of advanced education in computer science, typically organized as a specialization that covers from three to five courses such as Operating Systems, Databases, and Distributed Systems
> NOTE—The singular form "software system" can be used for (1) but it should not be used for (2).

21.4.2 Specialization names

In order to separate the area of education from software proposed for a certain task, it is recommended that the specialization name will be written as "Software Systems." In addition, it is recommended that instead of the very general specialization name "Systems," an institution should use the name "Software Systems" or "Computer Systems" according to the contents of a specialization. The name "Computer and Software Systems" can be used if a specialization actually covers both areas. However, in that case the area is so broad that, for example, the name "Computer Science (general)" might be more suitable.

In order to separate specializations in Software Systems from other computer science specializations, the name Software Engineering or Software Development should be used if a specialization emphasizes software engineering topics such as design methods (e.g., UML) more than technical topics such as operating systems.

21.4.3 Specialization and continuous education

For an institution, one possible strategy or tactic to respond to the increased and more versatile technical demands of industry is specialization in the specific skill groups or sectors of the job market.

The need for continuous education should be high because technology has changed continuously. One way to promote continuous education would be to offer part-time Master's programs to complement traditional degree programs. In Europe, an interesting question is whether the on-going harmonization of degrees known as the Bologna process (e.g., The Bologna Process…, 2005) will increase the number of part-time Master's programs. In any case, specialization in Software Systems is not exceptional in this respect because the need for continuous education is apparently high for most if not all areas of computer science.

21.4.4 Degree requirements

In this subsection, the recommendations were limited only to the typical requirements of accredited computer science (CS) programs in the USA. Parnas' (1999) paper concerned the differences between CS and software engineering (SE) programs. He wrote "In the SE program, the priority will be usefulness and applicability; for the CS program it is important to give priority to intellectual interest, to future developments in the field, and to teaching the scientific methods that are used in studying computers and software development." The results of the present thesis are probably more relevant to SE programs than CS programs because needs assessments emphasize

usefulness. However, these recommendations are about CS programs and specializations in Software Systems because the number of SE programs is so small. According to the Accreditation Board for Engineering and Technology (2005b), only six institutions offered accredited SE programs whereas approximately 190 institutions offered accredited CS programs. As Parnas put it, "Computer Science departments have tried to fill the gap by including so-called 'Systems' or 'Applied Computer Science' courses in their offerings."

The recommendations are divided into two subsections: Introductory topics and Advances topics.

Introductory topics

From various introductory topics, this subsection is limited to continuous mathematics, physics, theoretical computer science, and programming paradigms. Lethbridge (2000, pp. 49–50) wrote: "Because of the low importance and high forgetability of continuous mathematics and basic science, universities and colleges should either place less emphasis on these topics or they should teach them in a way that makes them more relevant to software engineering students." The author of the present thesis agrees with this recommendation. The role of mathematics in computer science education is a controversial subject that has been covered in several papers (e.g., Bruce, Drysdale, Kelemen, & Tucker, 2003). There was one working group that was "dedicated to promoting mathematics as an important tool for problem-solving and conceptual understanding in computing (Hendersen et al., 2001, p. 114)." Valmari's (2003) paper concentrated on mathematics related to software development that he called "software mathematics."

No similar papers about physics were found. This might indicate that physics is less necessary than mathematics because nobody has bothered or been able to publish a paper to argue why physics would be necessary for computer science or for software development, in particular. In any case, if physics is removed, an institution should take care that the requirements for scientific methods are not removed as an indirect consequence because according to Computing Curricula 2001 (Engel & Roberts, 2001, p. 41), scientific methods should be required. Still, Computing Curricula 2001 did not recommend physics or any other natural science as compulsory because scientific methods could be taught, for example, using laboratory experiments about the performance of algorithms.

Based on the results of the present thesis, the basics of theoretical computer science should be required. McCauley and Manaris (2002, p. 4) reported that 49% of ABET/CAC accredited Bachelor programs required the course Theory of Computation to be taken during the academic year 2001–2002. However, these results only concerned various upper-level courses. Some institutions require theoretical computer science in lower-level courses; that is,

during the first or second year. Based on the report, McCauley and Manaris' survey did not ask about this area. According to Computing Curricula 2001 (Engel & Roberts, 2001, p. 17), basic logic is included in the core topics but, for example, automata theory is not. Based on the results of the present thesis, some other areas of theoretical computer science might be more important than logic. However, the present thesis cannot answer the question what kind of theoretical computer science should be taught because the questionnaires of the present thesis were not detailed enough to make conclusions about this question. As an example, Valmari's (2001) paper is more detailed about this question. He wrote about Computing Curricula 2001: "In my opinion, the suggested content of discrete structures is small and partly poorly chosen. Instead of combinatorics and graphs there could be, for example, the theory of structure of clauses, BNF, constructing and analysis of definitions, or basic mathematics for reactivity and concurrency."

Five programming paradigms are imperative programming, functional programming, object-oriented programming, logic programming and constraint logic programming, and concurrent/distributed computing. Based on the results of the present thesis, the order of importance for these five paradigms is the following: 1. and 2. (tied place; i.e., joint first) imperative programming and object-oriented programming, 3. concurrent/distributed computing, 4. functional programming, and 5. logic programming and constraint logic programming. McCauley and Manaris (2002, p. 3) reported that in ABET/CAC accredited Bachelor programs during the academic year 2001–2002, 31% taught Procedure-Oriented and 82% Object-Oriented paradigm as the primary paradigm and the most common primary programming languages used were C++ (53%), Java (51%), and C (22%). Thus, the correspondence between the curricula and the results of the present thesis is good because the two most important paradigms were well covered.

Advanced topics

From the different advanced topics, this section is limited to databases, concurrent programming, distributed systems, and the relationship between technical and software engineering subjects and skills. McCauley and Manaris (2002, p. 4) reported how often various upper-level courses were required. The results of the present research imply that the course Databases should be required more often because database skills were required often in job advertisements but according to McCauley and Manaris, the course Database Management Systems was required only in 31% of the accredited programs.

Next, McCauley and Manaris' results on concurrent/distributed computing are presented because based on the results of the present thesis, it was the third most important programming paradigm. Their report had no results related to Distributed Systems and Concurrent Programming courses.

However, 96% of the departments required an Operating Systems course (ibid., p. 4). It is common that concurrency is part of an operating systems course. Thus, it is possible but not certain that the situation is good for the third important paradigm, too. The author of the present thesis does not recommend that concurrent programming and distributed systems are required in all computer science undergraduate programs because this would be overkill. After all, not all undergraduate students are aiming to get software development positions. Based on the results of the present thesis, concurrent programming and distributed systems are suitable required topics for a specialization in Software Systems in a Master's program.

Finally, professors in charge of specializations in Software Systems should ensure that technical topics such as operating systems and distributed systems are studied as well as software engineering topics such as design methods (e.g., UML). Omitting software engineering topics altogether from a specialization in Software Systems might be a mistake because many software engineering topics were evaluated as being important. In most institutions, a project course might be a suitable way to implement this. Besides, a project course is recommended in Computing Curricula 2001 (Engel & Roberts, 2001, pp. 42–43) for all computer science students.

21.4.5 Cognitive skills

It is obvious that many of the cognitive skills listed in Section 10 cannot be taught directly in the courses. They are highly related to a long experience gathered when programming solutions to different problems. The challenge for education is to design project assignments where students will face problems that require the mentioned skills, and find a way to present guidelines for adopting such skills.

On a more general level, the deployment of the results of Section 10 might increase the proportion of time used in concept exploration, requirements analysis, and design phases but decrease the proportion of time used in the implementation phase. In the following, a few course examples of such development are mentioned.

Refactoring Course

> This example would be an advanced course that emphasizes comprehension. During the Refactoring Course, a student should repair and/or partly rewrite a program (maybe 2000–3000 lines) that contains different kinds of mistakes and poor planning choices. During the task, a student has to read and thus interpret the structure and the operation of a program written by others. Moreover, he or she should argue about the findings made, and how the code should be improved.

Software Design Workshop

This course would emphasize the composition viewpoint, including analyzing and decision-making skills related to design. The course would contain an open or semi-open design problem that can be solved using several different strategies and tools. The student group should compare various options, argue their pros and cons, and finally evaluate the design that they selected.

Project Course (customization/tailoring)

In a customization/tailoring course the group faces the problem of designing a program for a variety of customers with slightly different needs. They should analyze the needs in the specification phase and argue what kind of architectural solution would enable generating different versions of the basic program, and argue their decisions on the program design. To make the project more challenging they should implement the first version, and thereafter get the requirements for new customers, and then analyze how their initial design works in the new situation.

Project Course (the combination of student projects)

The main goal here is to force students to read and understand the designs and implementation of other students, and continue their own work based on these. For example, a group should split a task into appropriate subgoals and assign a number of other groups a task to design and/or implement solutions for the subgoals. Thereafter the original group should compare a number of submitted designs/implementations and choose one or two of them as a part of their own project. They would have to use and modify the design/code to correspond to their needs and argue about the process; that is, reflect on their own decisions when assigning the subgoals and when comparing the results.

In general, the students should be faced with problems where they have to understand and modify code written by others, and not necessarily the best quality code with good documentation. This would promote both comprehension and analysis skills as well as composition skills.

21.5 Professional or academic emphasis in the curriculum?

The results of the present research are useful and relevant if an institute decides to change its computer science program or specialization in Software Systems

in order to become more professionally oriented. However, obviously each institution should decide for itself to what degree its computer science program should emphasize its professional or academic side. There are strong arguments in favor and against both.

As a compromise, at least larger university departments of computer science might decide to offer specializations with a greater academic emphasis (e.g., Artificial Intelligence or Programming Languages) as well as those with a stronger professional emphasis (e.g., Software Systems or Software Engineering). The Helsinki University of Technology apparently applies this type of compromise, even though it is not explicitly decided or documented as such. For example, the laboratory where the author works offers two specializations during the academic year 2005–2006: the specialization in Programming Languages can be classified as more academically oriented and the specialization in Software Systems as more professionally oriented. These specializations are presented in detail and discussed later in Part VII.

As was mentioned in Section 1.1.1, out of the nine data sources of the present thesis, the following three can be classified as more academic than professional: the degree requirements of research universities (Section 5), professors and lecturers (Sections 4 and 8), and the course catalog of the institution (Section 18). Academically oriented computer science programs can apply these results when they design curricula. In particular, the importance of artificial intelligence and compilers are evaluated as being somewhat greater according to these results than according to the results of the summative triangulation. Not surprisingly, compilers are a suitable topic for specialization in Programming Languages as well.

Smaller institutions probably have to choose one or other option because they might not have a large enough computer science faculty to implement such a compromise. Such institutions are often not research universities and offer degree programs that are more professionally oriented to start with. Thus, the whole question of academic or professional emphasis might be less relevant.

21.6 Admission procedures

Admission procedures of new undergraduate students differ considerably in various institutions and countries. In particular, it would be interesting to discover whether students are admitted to a particular institution or college as a whole or directly into degree programs. This might have an effect on whether physics and continuous mathematics are required in computer science programs. According to Rhoades (1991, p. 132), in the European context, “University students entered a faculty as opposed to the college or university as a whole, and their studies constituted a period of specialized professional

training for entry into a guild." The Helsinki University of Technology follows this admission procedure: students are selected directly to degree programs.

If students enroll in an institution or a college of engineering as a whole, without choosing a degree program or major: (a) the degree program or major in computer science is likely to be selected later, typically during the first or second year and (b) all (engineering) students have common requirements during the first year and possibly also during the second year. Often these common requirements include physics, mathematics, or both. At such institutions, the question of whether physics and continuous mathematics should be required for computer science students might be less interesting or relevant because removing these subjects could be considered as an unrealistic option.

21.7 Future research

The future research is considered in this subsection. First, suggestions for future research are presented for needs assessments. Second, suggestions for research on cognitive skills are presented.

21.7.1 Needs assessments

U.S. News (2004, p. 73) presented the separate TOP10 ranking lists of graduate programs for three specializations: Artificial Intelligence, Systems, and Theory. It is not surprising that the present thesis and previous needs assessments in the field of IT are relevant to the more industry-oriented specialization Systems. From the research methods used in the present thesis, job advertisement analysis is probably not suitable for researching the specializations Artificial Intelligence and Theory. However, concept analysis, the Delphi method, and the content analysis of degree requirements are suitable methods of investigating these two specializations as well.

Major web recruiting services such as Dice make it possible to conduct more specialized analyses with moderate effort when compared with newspaper analyses. For the planning purposes of undergraduate education (Bachelor's degree), it might be interesting to target an analysis only at internships or entry-level positions. In the present thesis, this limitation was not used because the thesis was targeted more at graduate (Master's degree) than at undergraduate education. It is the author's opinion that job advertisement analyses should be targeted more often than previously at entry-level positions because these results were probably the most relevant to the planning of education.

Graduate exit surveys could be used to research what proportion of students had internships or other working positions and what kind of positions they had. In addition to answering the questionnaire, a respondent should

provide copies of internship testimonials. At least in the institution, applications for training credits apparently cover only a small proportion of the working history of a single student before graduation.

21.7.2 Cognitive skills

Next are mentioned three possible research settings that might be interesting for follow-up research related to cognitive skills. By design, only research settings that use the Delphi method are mentioned because the present research was a Delphi study.

- The researchers of the psychology of programming could be asked as respondents, not experienced software developers. For example, the editors of the book Psychology of Programming (Hoc, Green, Samurçay, & Gilmore, 1990) might be possible candidates. It would be interesting to compare the results of these two respondent groups because it is possible that researchers in this field can mention some skills that software developers cannot—and vice versa. A researcher of psychology of programming might mention, for example, 10–30 cognitive skills when a respondent of the present research mentioned only 3–5 skills.
- The respondents could live in country other than in Finland because there might be some cultural differences related to the cognitive skills of software developers. These differences might be small but nevertheless, it would be interesting to explore if this is the case.
- Also a third questionnaire round could be organized. In the present research, only two questionnaire rounds were conducted because the respondents were promised that participating would take no more than 1–3 hours.

If a similar research project is repeated in the future, it is suggested that (a) the division composition versus comprehension, and (b) the definition of cognitive skills that is given at the beginning of Section 2.8 would also be used in the questionnaires. In addition, it is suggested that the first questionnaire should concentrate completely or mainly on cognitive skills. In the present research, the questions about cognitive skills were only a small part of the first questionnaire.

Automation of low-level skills in programming

No actual research setting is presented in the present subsection but only a possible hierarchy of low-level programming skills that might be automated is presented and discussed. Other researchers might use this hierarchy for the planning of research settings.

In the second questionnaire targeted at the experienced software developers, the respondents were asked about typing skills and the use of an

editor as well. This might seem odd because these are so low-level time-based skills; that is, fast typists who can use an editor well might be faster programmers. It would not be so interesting if software developers were 10% or even 50% faster but one could wonder if learning problems in low-level skills caused difficulties when students were learning higher-level programming skills. This might be the case based on earlier findings about skill automation and learning in general. For example, according to Wiedenbeck (1985, p. 384):

> Studying children, Perfetti and Hogaboam (1975) and Perfetti and Goldman (1976) found that less-skilled readers were significantly slower at low-level skills, such as letter and word encoding. This lack of automation of low-level skills led to inadequate discourse understanding, since memory for sentence wording decayed while the reader was trying to encode words.

One possible hierarchy of some low-level and intermediate-level programming skills or areas of knowledge is proposed in Figure 14. By design, very high-level skills are not presented in the figure because it is assumed that these skills cannot be automated. The related means from the respondents' answers are added into the figure. Next, the means are explained starting from the bottom of the figure: (a) In the boxes "Recognition of characters," "Editing skills," and "Design patterns," a dash (—) indicates that none of the results were really related. (b) The mean of the box "Typing skills" was presented in the body text in the beginning of Section 10.2.1. (c) The mean of the box "Syntax of programming languages" refers to Comment 2b in Table 19 (Section 10.2.1). (d) The mean of the box "Programming style and idioms" is a pooled mean from Comments 2b, 7b, and 13 of Table 19. (e) The mean of the box "Algorithms and data structures" refers to Comment 6 in Table 19. (f) The mean of the box "Programming tools and practices" is a pooled mean from Comments 2a, 2b, and 9 of Table 19, and Comment 8b of Table 20.

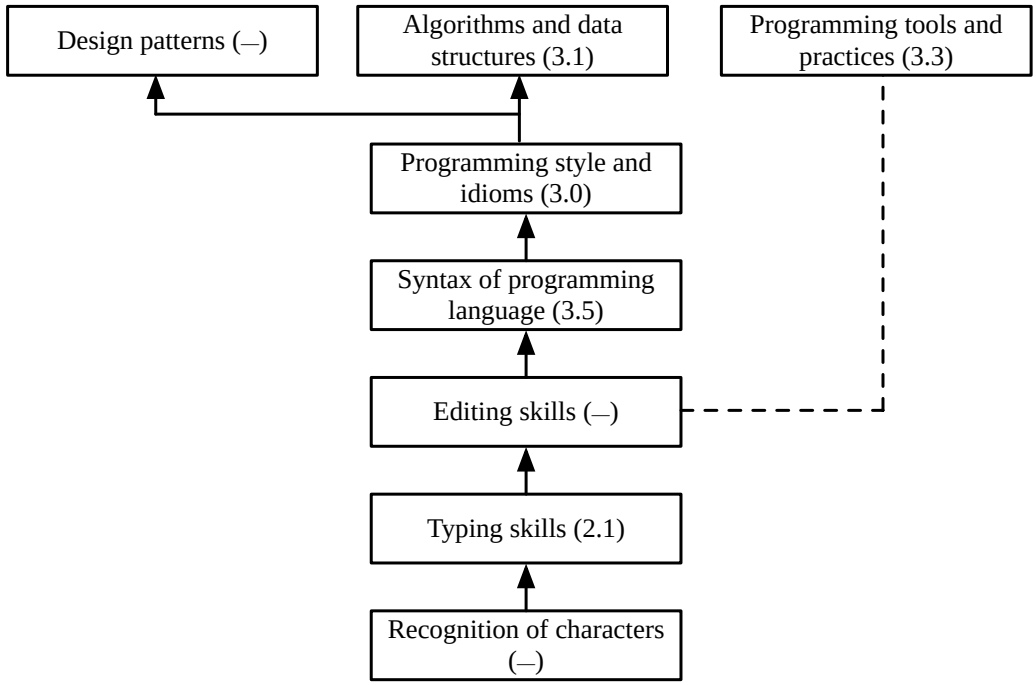

Figure 14. Possible hierarchy of some low-level and intermediate-level programming skills or areas of knowledge. Numbers are means from respondents' answers in related skills and a dash (—) means that no result was related.

The author of the present thesis does not have proper evidence to support the figure; the author just presents his arguments, which are based on his experience. First, it is obvious that all students of an undergraduate programming course know the alphabet, and learning the necessary special characters such as brackets and a tilde is not difficult for them. Second, good typing skills, especially the ability to concentrate on looking constantly at the screen, reduces interruptions in thinking. Perhaps frequent interruptions in order to look at the keyboard distract one from the reasoning process when reading and constructing programming language idioms.

Third, an editor can significantly reduce the workload of programming by automating a number of issues. For example, Emacs has different functions such as recognizing the syntax, support for the automatic indentation, showing the pairs of braces or brackets, and providing a number of ready-made keyboard commands to create various syntactical constructions for various programming languages. An experienced programmer can greatly benefit from these features if he or she is familiar with them and frequently applies them. For example, the automatic indentation not only saves the time to write the appropriate number of spaces but also easily points out possible errors when the indentation does not seem to work properly. Moreover, the keyboard commands are beneficial if a programmer wants to use a mouse as little as possible. During the literature

search, no really relevant publications were found about typing or editing as part of programming but Fry (1997, p. 63) wrote: “Switching between mouse and keyboard is bad. Most hackers I know think in terms of keyboard commands that perform equivalent mouse operations, so they don’t have to switch to and from the mouse.” This is just an anecdotal observation but nevertheless, the author of the present thesis agrees with this observation.

The fourth and fifth levels are concerned with knowledge of programming languages. Experienced programmers know the syntax and semantics of several languages by heart, which reduces their need to consult manuals and the number of syntactical and semantic errors they face when processing programs. What is more important is that they know how the language should be used to implement commonly appearing structures such as building linked lists.

Finally, the skills related to design patterns, and the selection of data structures and algorithms are intermediate-level or high-level skills. These skills are based both on knowledge about these issues and experience of what works well in practice.

Part VII: Case of Helsinki University of Technology

As explained in Section 3.1, the results of needs assessment can be used, for example, for planning or remediation to improve the situation. In this part, the results of the present needs assessment are applied to the planning of the case example that is the specialization in Software Systems at the Helsinki University of Technology. First, the related work is reviewed. Second, the case example is described. Third, the degree requirements of the institution are compared with the results of the present thesis and Computing Curricula 2001 (Engel & Roberts, 2001). Finally, recommendations are presented.

22 Related work

Here, the literature search was limited only to publications where the Degree Program of Computer Science and Engineering or the Laboratory of Information Processing Science at the Helsinki University of Technology were evaluated. The related publications are sorted by the year of publication in descending order because they were equally relevant.

The Laboratory of Information Processing Science is responsible for the specialization in Software Systems. The same laboratory is responsible also for the basic-level of programming education that was selected as a center of excellence in higher education in Finland for the periods 2001–2003 and 2004–2006 (Moitus, 2000; Parpala & Seppälä, 2003). The Finnish Higher Education Evaluation Council (FINHEEC) coordinated the selection process. For the period 2004–2006, the Finnish institutions of higher education suggested 64 candidates when each institution was allowed to make from one to eight suggestions based on the number of students. Twenty out of these 64 candidates were selected as centers of excellence. The experts selected by FINHEEC evaluated the applications of the candidates as part of the selection process. The evaluation of the application of the Laboratory of Information Processing Science is published in Parpala and Seppälä (2003, pp. 311–312). However, this evaluation is less relevant to the present thesis because the application was limited to introductory studies.

One extensive evaluation of Finnish computer science education was conducted when several degree programs from the universities and polytechnics were evaluated (Hara, Hyvönen, Myers, & Kangasniemi, 2000). However, from the viewpoint of one specialization this evaluation was not precise enough. The recommendations were, for example, about yearly intakes, teaching methods, and monitoring how students proceed in their studies (ibid., p. 59).

Computer science education was evaluated as part of the Finnish evaluation of exact sciences, which were defined as mathematics, physics, chemistry, and computer science (Lounasmaa, 1996, p. 1). This evaluation covered both research and education. Several laboratories or other units of eleven institutions were selected for this evaluation where indexes were counted for each unit in undergraduate education, graduate education, and research. The purpose was to count indexes for outputs such as the number of degrees in relation to resources such as the number of faculty. Based on these indexes, the units were given grades from A to E where A was the best. The Laboratory of Information Processing Science got grade C in postgraduate education and D in research (ibid., Table 7). No grade for undergraduate education was given to the laboratory but the grade was common with another laboratory of the computer science department. This grade for undergraduate education was C (ibid., Table 7). Thus, based on these grades, the laboratory was satisfactory in education but almost poor in research. He wrote about the laboratory (ibid., p. 81, translated from Finnish): "In particular, there are possibilities for development in research because the classification was CD." However, this evaluation is less relevant to the present thesis because it is almost ten years old and not detailed enough. The recommendations about education were, for example, about financial aid to students and tutoring (ibid., p. 154).

23 Description of case example

The case example is the Helsinki University of Technology that was described previously in Section 15. The institution changed its degree structure during the thesis project. In the next two subsections, the new structures of Bachelor's and Master's degrees are presented.

23.1 Scope by structure of Bachelor's degree

From the beginning of the academic year 2005–2006, the institution will offer a Bachelor's degree and use European Credit Transfer System (ECTS) credits as a consequence of the harmonization of European degrees known as the Bologna process (The Bologna Process…, 2005). Next, the scope of the present thesis is explained using the structure of the new Bachelor's degree. The structure is presented in Figure 15. A three-year Bachelor's degree will be 180 ECTS credits.

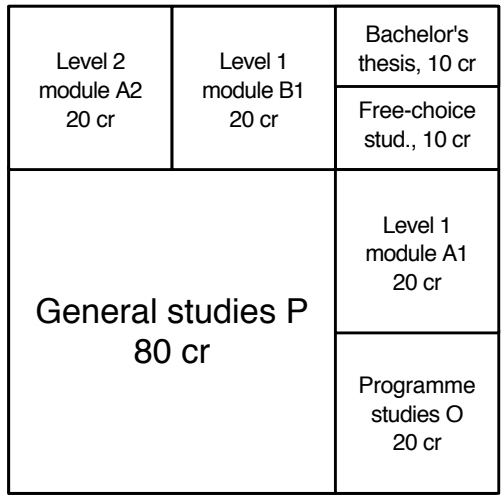

Figure 15. Structure of new Bachelor's degree (Helsinki University of Technology, 2005b). Abbreviation "cr" means ECTS credits.

The modules General studies P and Programme studies O are common for all students of the program. These modules include mainly mathematics, physics, and various introductory courses in computer science. The numbers of credits for these modules are presented in Table 56 where the computer science courses are divided according to the areas of Computing Curricula 2001 (Engel & Roberts, 2001, p. 17). Whether the recommendations for the core topics of Computing Curricula 2001 (ibid., p. 17) are covered is mentioned in the column "CC2001 covered?" The area Computational Science is not relevant because the area is mentioned in Computing Curricula 2001 but no core hours are recommended.

It can be noticed from the table that the following seven areas of Computing Curricula 2001 are not at all or only partly covered: Architecture and Organization, Discrete Structures, Operating Systems, Graphics and Visual Computing, Intelligent Systems, Social and Professional Issues, and Software Engineering.

Table 56. Common studies for all students in computer science program of institution and whether these requirements cover recommendations of CC2001.

Subject	ECTS credits[a]	CC2001 covered?
Other than computer science studies:		
Mathematics	30	Not relevant
Physics	12	Not relevant
Other studies (e.g., foreign languages)	15	Not relevant
Areas of Computing Curricula 2001:		
Programming Fundamentals	5	Yes
Algorithms and Complexity[b]	9	Yes
Net-Centric Computing	5	Yes
Human-Computer Interaction	2	Yes
Information Management	5	Yes
Programming Languages	6	Mostly
Architecture and Organization	3	Partly
Discrete Structures	0	Partly
Operating Systems	0	No
Graphics and Visual Computing	0	No
Intelligent Systems	0	No
Social and Professional Issues	0	No
Software Engineering	0	No
Computational Science	0	Not relevant
Computing-related subjects:		
Neural networks and signal processing	4	Not relevant
Multimedia	4	Not relevant
Sum	**100**	

[a]Source: Helsinki University of Technology, 2003c, pp. 15–16. The author classified the courses into the areas of Computing Curricula 2001 [CC2001] (Engel & Roberts, 2001, p. 17).
[b]Includes also the course Introduction to Theoretical Computer Science.

After these common studies, a student has to choose one Level 1 module A1 known as A1 module from two alternatives: "Computer Science and Engineering" and "Neural Networks and Signal Processing[9]." The A1 module Computer Science and Engineering is relevant to the present thesis. The required courses of this module in the academic year 2005–2006 are as follows: Intermediate Course in Programming T2 (the C language), Introduction to Software Engineering, Computer and Operating System, and Logic in Computer Science: Foundations (Kerola, 2005, p. 16). These courses cover the following two core areas of Computing Curricula 2001: Architecture and Organization and Software Engineering. In addition, the course "Logic in Computer Science: Foundations" partly covers the area Discrete Structures. Thus, the following CC2001 core areas are still not at all or poorly covered after the A1 module is studied: Operating Systems, Graphics and Visual Computing, Intelligent Systems, and Social and Professional Issues.

[9] The official name is "Computer and Information Science" but "Neural Networks and Signal Processing" is used here because it is probably easier to understand for most readers.

Next, a student will choose one Level 2 module A2 known as the A2 module. The Degree Program of Computer Science and Engineering will offer seven A2 modules. From these, the A2 module Software Technology is relevant to the present thesis. During the academic year 2005–2006, the required courses of this module are the following (ibid., p. 21): Design and Analysis of Algorithms, Operating Systems and Concurrent Programming, Introduction to Compiling, and Introduction to Artificial Intelligence. These courses cover the following two core areas of Computing Curricula 2001: Intelligent Systems and Operating Systems. Thus, the following core areas are still not at all or poorly covered after the A2 module is studied: Graphics and Visual Computing, and Social and Professional Issues.

In addition, a student has to choose one Level 1 module B1 known as the B1 module or minor that can be from the computer science department, from the other departments of the institution, or even from another institution. The details of B1 modules are not considered here because they are not relevant to the present thesis.

A student has to also take some elective courses to fulfill the total requirement of 180 ECTS credits ("Free-choice stud." in the figure). Finally, in the module Bachelor's thesis, a student must take a seminar and write a brief report.

23.2 Scope by structure of new Master's degree

The structure of the new Master's degree is presented in Figure 16. Two-year Master's degree will be 120 ECTS credits. A student has to take one Level 3 module A3 known as the A3 module or major, one Level 2 module B2 known as the B2 module or minor, and one Special module C known as the C module. The computer science department offers the A3 modules but the B2 and C modules can be from the other departments and institutions as well. In addition, a student has to take 20 credits of elective courses referred to as "Free-choice studies V2" in the figure, 10 credits of courses on research methodology referred to as "Method." in the figure, and conduct a Master thesis project.

The program will offer eighteen A3 modules. From these, the A3 module Software Systems is relevant to the present thesis. The module has no required courses but two lists of electives. A student has to choose one project course from the following list (Kerola, 2005, p. 22):

- Project in Software Techniques
- Operating Systems Project
- Software Development Project.

In addition, a student has to choose courses so that the extent of the whole A3 module will be at least 20 ECTS credits. Typically, this means that a student has to choose two or three courses from the following list (ibid., p. 22):

- Database Algorithms
- Distributed Systems
- Embedded Systems
- String Algorithms
- Advanced Course on Compilers
- Seminar on Software Techniques.

The details of B2 modules are not considered here because they are not relevant to the present thesis. The C modules are proposed, for example, for more advanced topics than are covered in A3 modules. Thus, the C modules offered by the Laboratory of Information Processing Science would be relevant to the present thesis. However, the degree requirements of C modules are not considered because these are not published yet.

During the academic year 2005–2006, the methodological studies are agreed with the professor of a major (ibid., p. 16). The methodological studies are not compared with the results of the present thesis because the course requirements are not published and the requirements might vary by student.

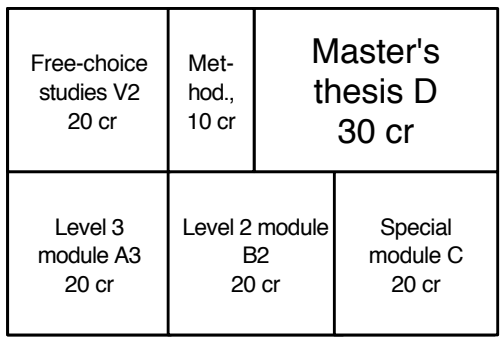

Figure 16. Structure of new Master's degree (Helsinki University of Technology, 2005b). Abbreviation "cr" means ECTS credits.

23.3 Generality of specialization in Software Systems

In the present subsection, the requirements of the case example are compared with the requirements of the American specializations in Software Systems. These American requirements were content analyzed earlier in Section 5. The requirements of the American undergraduate programs are compared with the requirements of the A2 module Software Technology in Table 57. The requirements of the American graduate programs are compared with the requirements of the A3 module Software Systems in Table 58.

The most common courses of the American specializations are presented in the column Course of Table 57. How often the course was required or elective in the American specializations is presented in the column "Proportion in the USA." Whether the course is required in the case example is presented in the column "Required in case example?" where the text "As prerequisite" means that the course was not required in the A2 module Software Technology but already previously in the modules General studies P, Programmer studies O, or A1 Computer Science and Engineering. It can be noticed from the table that the four most common courses in the American programs were required at the institution in the A2 module Software Technology or already previously as a prerequisite.

Table 57. Most common courses of specialization in Software Systems in American undergraduate programs ($n = 10$), their proportions, and whether course is required in case example.

Course	**Proportion in the USA** (%)	**Required in case example?**
Computer Networks	70	As prerequisite
Compilers	60	Yes
Databases	60	As prerequisite
Operating Systems	60	Yes
Computer Architecture	40	As prerequisite
Computer Graphics	40	No
Programming Languages	40	No
Software Engineering	40	As prerequisite
Distributed Systems	30	No
Advanced Operating Systems	20	No

Similarly, the most common courses of the American graduate programs are presented in Table 58 and compared with the requirements of the case example. It can be noticed that out of the three most common courses in the American programs, two are required as prerequisites for the A3 module Software Systems. However, the course Programming Languages is not required or even offered as an elective course in the A3 module Software Systems. A probably explanation is that the Laboratory of Information Processing Science offers the A3 module Programming Languages where the course Principles of Programming Languages is offered as an elective.

Table 58. Most common courses of specialization in Software Systems in American graduate programs (n = 18), their proportions, and whether course is required in case example.

Course	Proportion in the USA (%)	Required in case example?
Compilers	61	As prerequisite
Databases	61	As prerequisite
Programming Languages	61	No
Advanced Operating Systems	50	No
Distributed Systems	50	Elective
Computer Networks	44	As prerequisite
Operating Systems	44	As prerequisite
Software Engineering	22	As prerequisite
Computer Architecture	17	As prerequisite
Computer Graphics	17	No

24 Comparison

First, the offered specializations of the whole degree program of the institution are compared with the most common specializations in American research universities. Second, comparison at course level is made for the specialization in Software Systems.

24.1 Offered specializations

The most common specializations in the American research universities according to the content analysis of degree requirements (Section 5.2.2) are presented in Table 59. How often a specialization was offered is presented in the columns Undergraduate and Graduate. The rows are ordered first according to the column Undergraduate and then according to the column Graduate. Whether a similar specialization will be offered in the computer science program of the institution during the academic year 2005–2006 is presented in the column "Institution offers?" It can be noticed that the institution will not offer the specializations Computer Systems, Artificial Intelligence, Scientific Computing, Computer Graphics, and Databases that were at least somewhat common in the American undergraduate or graduate programs; that is, the proportion was greater than 30%.

Table 59. Proportions (%) of offered specializations in American computer science programs and if they are offered at institution.

Specialization	**Undergraduate** (n = 18)	**Graduate** (n = 29)	**Institution offers?**
Computer Systems	72	55	No
Theoretical Computer Science	67	72	Yes
Software Systems	56	62	Yes
Artificial Intelligence	50	69	No
Scientific Computing	44	28	No
Programming Languages	39	28	Yes
Computer Graphics	28	35	No
Computer Networks	22	24	Yes
Algorithms	22	21	No
Databases	11	35	No
Software Engineering	11	14	Yes
Usability	11	14	Yes

Next, the specializations not offered are discussed in the same order as they are presented in the table. Apparently specialization in Computer Systems is rare in all Finnish universities or not offered at all. A possible explanation is that components of a computer are rarely designed, manufactured, and assembled in Finland. Apparently, there is only one Finnish company that designs and assembles computers (www.pomi.fi). However, this specialization might be useful for systems administration positions as well. In the job advertisement analyses of the present thesis, the proportions of different IT job titles were not revealed but according to the ITAA's 2003 workforce survey (Information Technology Association of America, 2003, p. 5), the proportion of the category "Technical support" was 18.5% in the USA. According to the draft of Computing Curricula 2005 (Shackelford, 2005, p. 31): "…, there is a fourth career path that CS programs do not target but nonetheless draws many computer science graduates: *Career Part 4: Planning and managing organizational technology infrastructure*. This refers to the work which the new *information technology* (IT) programs explicitly aim to educate students."

During the academic year 2004–2005, the specialization in Artificial Intelligence was still offered but it is no longer offered in the academic year 2005–2006. However, it is possible that the Laboratory of Information Processing will later offer it as a C module because "Intelligent Systems" was mentioned as a possible topic for a C module in the series of slides used during the planning meetings in the spring of 2005 (L. Malmi, personal communications, January 13, 2005).

The Department of Electrical and Communications Engineering of the institution offers the specialization in Scientific Computing. Thus, an interested computer science student can study it as a minor.

At the institution, computer graphics is part of the specialization in Digital Media that is offered in the computer science program. Other topics of the specialization are multimedia, hypermedia, and user interfaces.

It is possible that the Laboratory of Information Processing Science will later offer Algorithms as a C module because it was mentioned as a possible topic for a C module in the series of slides used during the planning meetings in the spring of 2005 (L. Malmi, personal communications, January 13, 2005).

At the institution, three database courses are offered. The Laboratory of Software Business and Engineering offers the courses Database Management and Seminar on Database Management and the Laboratory of Information Processing Science offers the course Database Algorithms. According to the job advertisement analyses of the present thesis, database skills were required often in software developer positions and according to the same ITAA's survey (ibid., p. 5), the proportion of the category "Database development/administration" was 9.8%. Perhaps a specialization in databases could attract some motivated students as well.

24.2 Required courses of specialization in Software Systems

First, the degree requirements of the institution are presented for the A2 module Software Technology and the A3 module Software Systems. During the academic year 2005–2006, the required courses for the A2 module Software Technology are the following (Kerola, 2005, p. 21):

- Design and Analysis of Algorithms
- Operating Systems and Concurrent Programming
- Introduction to Compiling
- Introduction to Artificial Intelligence.

Second, it is compared how these requirements correspond with the results of the present thesis:

- The degree requirements and the results of the present thesis correspond well for the course Operating Systems and Concurrent Programming because the subjects operating systems and concurrent programming were evaluated as being important.
- The case of compilers is less clear because it was an important topic according to the software developers and the professors and lecturers but only a little or somewhat important according to the students.
- The biggest difference is that artificial intelligence is required but it was evaluated as being a less important topic by the software developers and the students.

- The evaluation of the course Design and Analysis of Algorithms was difficult because its importance was not questioned in the questionnaires. The questionnaire item "Data structures and algorithms" was evaluated as being important which might indicate that the topics of Design and Analysis of Algorithms are important as well. However, this is a tentative assumption. According to Lethbridge's (1999, pp. 32–33) results, the item Computational Complexity and Algorithm Analysis was evaluated as being less important than the items Data Structures and Design of Algorithms.

Third, the degree requirements of the institution are presented for the A3 module Software Systems. The module has no required courses but two lists of electives. A student has to choose one project course from the following list (ibid., p. 22):

- Project in Software Techniques
- Operating Systems Project
- Software Development Project.

In addition, a student has to choose courses so that the extent of the whole A3 module will be at least 20 ECTS credits. Typically, this means that a student has to choose two or three courses from the following list (ibid., p. 22):

- Database Algorithms
- Distributed Systems
- Embedded Systems
- String Algorithms
- Advanced Course on Compilers
- Seminar on Software Techniques.

Fourth, it is compared how the requirements of the A3 module correspond with the results of the present thesis. The author interpreted the degree requirements so that (a) project courses are more important than the courses from the second list because one project course is required, and (b) all courses from the second list are equally important. The requirements and the results of the present thesis correspond well for typical project course topics and because the corresponding items operating systems, concurrent programming, and distributed systems, and the software development life cycle phases requirements, design, implementation, and test were evaluated as being very important or at least important.

The biggest difference is that based on the degree requirements, embedded systems is equally important with the other topics of the second list but, according to the results of the present thesis, it was a less important topic. In particular, the importance of distributed systems is greater than the importance of embedded systems according to the results of the present thesis.

It is unclear how important the database algorithms, string algorithms, and advanced topics on compilers are because their importance was not queried in the questionnaires. Database management systems were evaluated as being important and compilers as at least somewhat important which might indicate that these topics were important as well. However, this is an uncertain assumption. No results are suitable for evaluating the importance of string algorithms.

Seminar on Software Techniques is not compared with the results of the present thesis because the topic of the seminar typically changes each term.

25 Recommendations

In this section, the recommendations for the institution are presented. First, recommendations for setting up new modules are presented. Second, recommendations for the A2 and A3 modules are presented.

25.1 New modules

It is recommended that a new C module "Databases" will be set up. Apparently, the Laboratory of Information Processing Science does not have enough resources for setting up such a module alone but this might be possible in co-operation with the University of Helsinki and the Laboratory of Software Business and Engineering at the Helsinki University of Technology. This topic would be suitable for an A3 module as well. However, it is recommended as a C module because setting up C modules is probably easier than A3 modules. The module could be later changed to an A3 module if it is popular enough among students. The A2 modules "Software Technology" or "Software Engineering and Business" would be suitable enough prerequisites for the C module Databases.

It is recommended that the need and possibilities for a new A2 module "Systems Administration" will be further investigated. This module would be a kind of Finnish substitute for specialization in Computer Systems that was common in American research universities. A possible course list for this module could be, for example: Operating Systems and Concurrent Programming, Operating Systems Project, Computer Networks, and Information Security Technology. This module might be terminal; that is, proposed for students who do not plan to continue their studies after a Bachelor's degree and no A3 module in systems administration would be offered. However, the Laboratory of Information Processing Science could not alone decide on setting up this type of module because it was proposed as an A2 module. This module is not recommended as an A3 or C module because

the topic is more suitable for an A2 module. The main reasons are that junior systems administrators are entry-level positions and the other A2 modules are not suitable as prerequisites for this module.

25.2 Requirements of A2 and A3 modules

In this subsection, the recommendations for the requirements of A2 and A3 modules are presented. However, the criteria used will be presented before the recommendations because these kinds of recommendations are typically compromises between several criteria. Mainly, the results of the present thesis were used for making the recommendations. In addition, Computing Curricula 2001 (Engel & Roberts, 2001) was considered seriously. As was explained previously in Section 23.1, the requirements before the A2 and the A3 modules adequately covered the core requirements other than the following topics: Graphics and Visual Computing (3 core hours), Intelligent Systems (10 core hours), Operating Systems (18 core hours), and Social and Professional Issues (16 core hours). From these topics, Graphics and Visual Computing was not considered because the number of required core hours was so small and Social and Professional Issues was not considered because this topic was more suitable for being implemented as a co-operation of the whole computer science department.

The following criteria were deliberately not used: (a) Costs. For example, one could select cheaper courses for the A2 module where the number of students was probably quite large (50–100 students per year). (b) The abilities of computer science students who will study an A2 module as a minor. The module for minor students will be called a "B1 module" but the requirements are exactly the same as in the A2 module. For example, one could select less demanding courses for the A2/B1 module in order for minor students to be able to pass the module as well. In other words, only the needs and abilities of major students were considered.

Next, recommendations are presented for the A2 module Software Technology. It is recommended that the course Introduction to Compiling is changed to the course Operating Systems Project. Thus, the following courses are recommended for the A2 module Software Techniques:

- Operating Systems and Concurrent Programming
- Operating Systems Project
- Introduction to Artificial Intelligence
- Design and Analysis of Algorithms.

The main reasons for this recommendation are:

- The courses "Operating Systems and Concurrent Programming" and Operating Systems Project were selected because according to the results of

the present thesis, concurrent programming was the third important programming paradigm. The course Operating Systems and Concurrent Programming alone is not enough because it does not include a programming assignment. Thus, Operating Systems Project is needed to make sure that students learn at least some concurrent programming. In addition, a Capstone Project was recommended in Computing Curricula 2001 (Engel & Roberts, 2001, p. 45). Operating Systems Project is demanding and suitable enough to cover this recommendation.

- The course Introduction to Artificial Intelligence was selected because according to Computing Curricula 2001 (ibid., p. 17), some topics of intelligent systems should be covered.
- The course Design and Analysis of Algorithms was the most difficult to choose. It was selected because based on the results of the questionnaires, data structures and algorithms were evaluated as being important or very important. In the questionnaires, it was not asked if the design and analysis of algorithms was important but the above result might indicate that it was.
- The course Introduction to Compiling was not selected because it was more suitable for the A3 module Programming Languages. Based on the analysis of course prerequisites, the courses Compilers and Programming Languages were somewhat linked.

Next, recommendation for the A3 module Software Systems is presented. Here, Computing Curricula 2001 was no longer useful and the recommendations were based on the author's interpretations of the results of the present thesis. The Laboratory of Information Processing Science will offer two A3 modules during the academic year 2005–2006: Software Systems and Programming Languages. It is emphasized that it was also considered if a course was more suitable for the A3 module Programming Languages. In some cases, a subject was evaluated as being important but the corresponding course was not recommended in the A3 module Software Systems because it was considered as being more suitable for the A3 module Programming Languages. In addition, it was assumed that the A3 module Software Systems would be more industry-oriented or less academic and the A3 module Programming Languages more academic. At least in Finland, the typical courses of a specialization in Programming Languages were not usually offered in polytechnics[10] when the typical courses of a specialization in Software Systems were often offered in polytechnics as well. This was interpreted so that a specialization in Software Systems would be more industry-oriented or less academic than a specialization in Programming Languages.

[10] Finnish polytechnics can be classified into the category Baccalaureate Colleges in the Carnegie Classification of Institutions of Higher Education (Carnegie Foundation, 2005).

The following courses are recommended for the A3 module Software Systems:

- Distributed Systems
- Embedded Systems
- Project in Software Techniques or Software Development Project
- Elective computer science course (if required for credits).

It is possible that a student might not have to take an elective course because the number of credits for the project courses varies.

The main reasons for this recommendation were:

- The course Distributed Systems was selected because according to the results of the present thesis it is an important topic. In addition, its importance was forecasted to increase in the future.
- A project course was selected in order for students to get experience of the software development life cycle phases requirements, design, implementation, and test. According to the results of the questionnaires, these phases were important.
- The course Embedded Systems was selected because the Degree Program of Computer Science and Engineering educates engineers. Based on the content analysis of job advertisements published on the WWW, embedded systems and other low-level programming skills were required more often in software engineer positions than in other software developer positions.
- The course Software Architectures was not included even though it was evaluated as being important in the questionnaires. Software Architectures was not included because it had the course Software Design and Specification Methods as a prerequisite. Software Architectures would probably have been recommended instead of Embedded Systems if Software Architectures did not have such prerequisites.

In addition, a recommendation for the A3 module Programming Languages is presented. This was not an original goal of the present thesis. However, this recommendation was made to concretize how the A3 modules Software Systems and Programming Languages would be different. The following courses are recommended for the A3 module Programming Languages:

- Principles of Programming Languages
- Introduction to Compiling
- Functional Programming
- Logic and Constraint Programming or Advanced Course on Compilers.

The main reasons for this recommendation were:

- The course Principles of Programming Languages is central to this module even by the course name.

- The course Introduction to Compiling was selected because based on the analysis of the course prerequisites, the courses Programming Languages and Compilers were somewhat related. In addition, compilers were evaluated as being important in the questionnaires by the software developers and by the professors and lecturers.
- The course Functional Programming was selected because based on the results of the present thesis, functional programming was more important than logic programming.
- The courses Logic and Constraint Programming and Advanced Course on Compilers were recommended as alternatives because based on the results of the present thesis, it was not possible to conclude if one of them was more important than the other.

Finally, the recommended courses of the A2 and A3 modules are summarized in Figure 17.

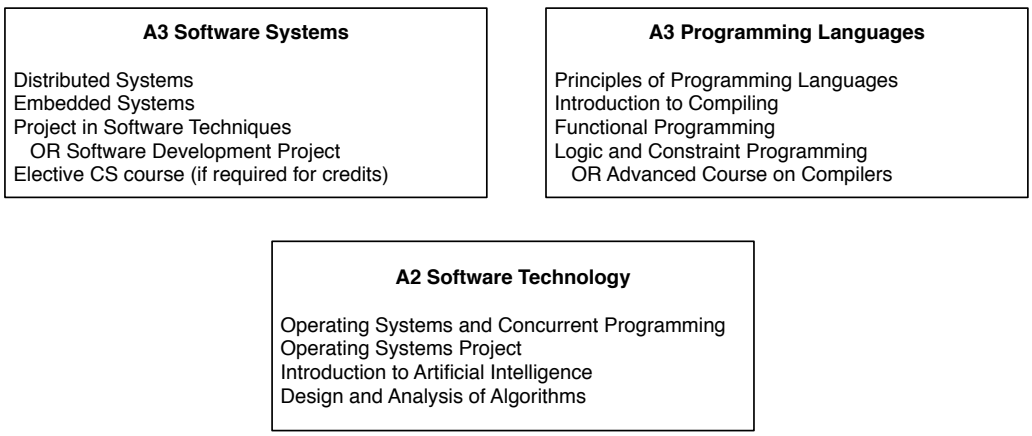

Figure 17. Recommended courses for three modules.

Part VIII: Summary of the thesis

The summary of the present thesis is presented in this part. First, the research problem, research methods, and data sources are presented. Second, the main results and other contributions are described. Finally, the recommendations are repeated.

Research problem, methods, and data sources

The main research problem of the present thesis was: What technical skills do graduates from a specialization in Software Systems need? Technical skills refer to, for example, operating systems and object-oriented programming. Soft skills such as communication skills were investigated not at all or only a little in the present thesis because it was assumed that technical skills are essential to get the first entry-level position as software developer.

From various information technology (IT) positions, such as those of consultants, database administrators, project managers, and systems administrators, the present thesis was targeted at software developer positions. Software developers were used to denoting programmers and software engineers as well.

The thesis project was conducted in 2001–2005. Triangulation; that is, several research methods and data sources were used to solve this problem (see Figure 18).

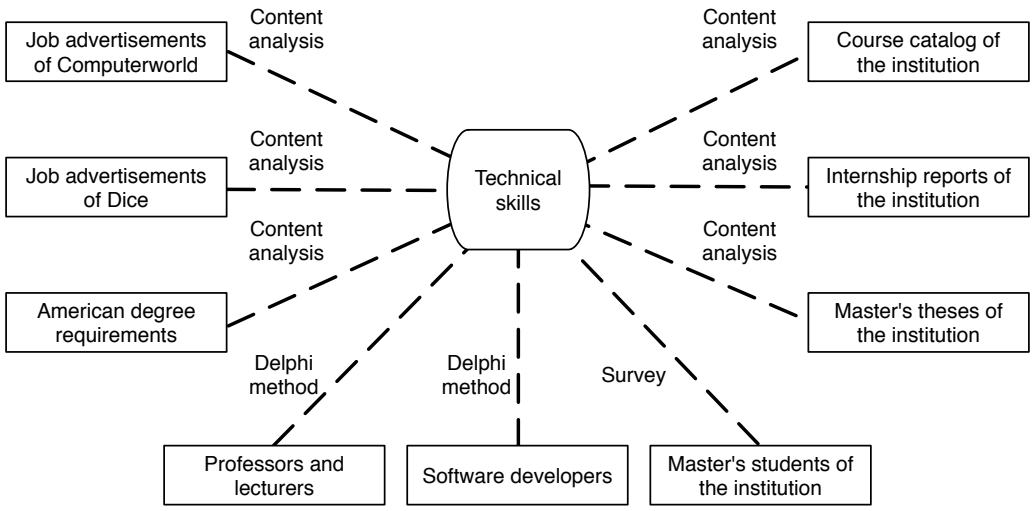

Figure 18. Data sources and research methods of present thesis.

Main results and contributions

The results of the present thesis are summarized in Table 60. The purpose of this summarization was to find out which subjects or skills were evaluated as being important. For each subject or skill, a sum of points was counted so that from a single body of research it was possible to get zero or one point. One point means that a subject or skill was important according to the research in question. The columns from "Concept analysis" to "Job advert." refer to the different research areas of the present thesis. The rows are ordered first according to the column Sum and then according to the names of subject or skills. Two different types of job advertisement analyses were classified as one research area. Thus, the maximum number of points was six because six separate research areas or research types were used.

Table 60. Summarized results of present thesis.

Subject or skill	Concept analysis	Delphi study (developers)	Delphi study (professors)	Survey (students)	Job advertisements	Degree requirements	Sum
Mathematics, physics, and theoretical CS:							
Discrete mathematics			1				1
Other areas of theoretical CS (e.g., automata)		1					1
Logic (in particular, propositional and predicate l.)							0
Mathematics for continuous systems							0
Physics							0
More technical or part of the operational system:							
Operating systems	1	1	1	1	1	1	6
Database managements systems	1		1	1	1	1	5
Distributed systems	1	1	1		1	1	5
Compilers	1	1	1			1	4
Concurrent programming	1	1	1	1			4
Data structures and algorithms	1	1	1	1			4
Object-oriented programming		1	1	1	1		4
Procedural programming		1	1	1	1		4
Software architectures	1	1	1	1			4
Computer architecture		1	1			1	3
Computer/data security		1	1	1			3
Internet protocols		1		1		1	3
Implementing techniques of user interfaces			1	1			2
Script programming		1		1			2
Computer graphics						1	1
Embedded systems	1						1
Extensible Markup Language (XML) techniques				1			1
Functional programming		1					1
Systems programming		1					1
Artificial intelligence and knowledge engineering							0
Implementing techniques of WWW systems							0
Logic programming							0
Telecommunications techniques other than Internet pr.							0
Real-time systems							0
Software eng. (different phases of life cycle):							
Concept exploration		1	1	1			3
Design		1	1	1			3
Implementation		1	1	1			3
Requirements		1	1	1			3
Test		1	1	1			3
Approval							0
Installation and checkout							0
Operation and maintenance							0
Packaging and delivery							0
Retirement							0
Software engineering (possible in several phases):							
Documenting		1	1	1			3
Project management		1	1	1			3
Version and configuration management		1	1	1			3

The most important contributions of the present thesis are:

- The present thesis provided findings that the requirements for software developers increased and have required greater versatility during the past 15 years. This general trend was reported apparently the first time in 1995 for the period 1970–90 (Todd et al., 1995). However, it was interesting to know if this trend had continued after 1990.
- Based on the summarized results, the following technical subjects were evaluated as being important. These items are presented in alphabetical order: compilers, concurrent programming, data structures and algorithms, database management systems, distributed systems, object-oriented programming, operating systems, procedural programming, and software architectures. Most of these subjects or skills were previously reported as being important for software developers, for example, by Lethbridge (2000).
- The present thesis provided supporting findings that physics and continuous mathematics were not important for software developers. Previously, Lethbridge (2000) reported similar results. These supporting results were useful because Lethbridge's methodology was criticized (Kitchenham & Pfleeger, 2002, p. 17). The necessity for these subjects is an important question because the proportion of physics and continuous mathematics is large in computer science education on average.
- In the job advertisement analyses of the present thesis, technical skills were analyzed in a more detailed manner than in the previous analyses on average. In particular, some results concerning distributed technology skills were new and more detailed than previously published.
- In the questionnaires of the present thesis, different programming paradigms were analyzed in a more detailed or different manner than previously. Based on the results, it was possible to conclude the order of importance of these paradigms.

The following were contributions from the viewpoint of research methods:

- The thesis has been so far the most versatile triangulation in the area in question. In particular, the content analysis of American degree requirements and the concept analysis of "software systems" were novel parts.
- Previously, statistical tests were used in surveys often but rarely in job advertisement analyses. This was interesting because job advertisement analysis was the most common research type in this area. In the present thesis, statistical tests were used to analyze the results of job advertisement analyses as well.

Recommendations

First, general recommendations for computer science programs are presented:

- Lethbridge (2000, pp. 49–50) wrote: "Because of the low importance and high forgetability of continuous mathematics and basic science, universities and colleges should either place less emphasis on these topics or they should teach them in a way that makes them more relevant to software engineering students." The author of the present thesis agrees with this recommendation.
- The basics of theoretical computer science should be required. However, no detailed recommendation is given as to what these basics topics should include.
- The course on databases should be required more often.

Finally, the case-specific recommendations for the Helsinki University of Technology are presented. It is recommended that the Laboratory of Information Processing Science will set up a new C module "Databases" in co-operation with the University of Helsinki and the Laboratory of Software Business and Engineering at the Helsinki University of Technology.

The recommended courses of the A2 module Software Technology and the A3 modules Software Systems and Programming Languages are presented in Figure 19.

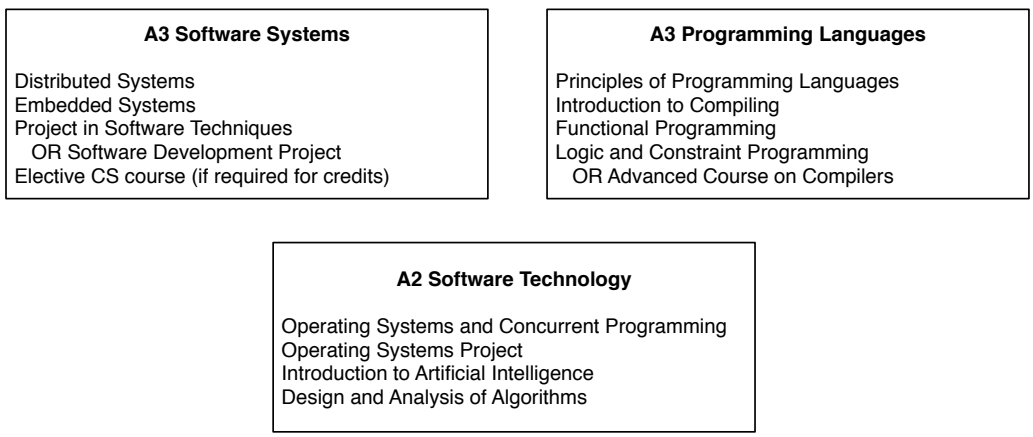

Figure 19. Recommended courses for three modules.

References

Accreditation Board for Engineering and Technology. (2004). *Criteria for accrediting computing programs. Effective for evaluations during the 2005-2006 accreditation cycle*. Retrieved June 17, 2005, from the Accreditation Board for Engineering and Technology web site: http://www.abet.org/.

Accreditation Board for Engineering and Technology. (2005a). *Accredited Computing Programs*. Retrieved January 14, 2005, from the Accreditation Board for Engineering and Technology web site: http://www.abet.org/accredited_programs/computing/schoolall.asp.

Accreditation Board for Engineering and Technology. (2005b). *Accredited Programs*. Retrieved May 14, 2005, from the Accreditation Board for Engineering and Technology web site: http://www.abet.org/accredited_programs.html.

Adelman, C. (2000). A parallel universe. *Change*, *32*, 3, 20–29.

American Psychological Association. (2001). *Publication manual of the American Psychological Association* (5th ed.). Washington: American Psychological Association.

Anderson, J. (1994). Content and text analysis. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 2, pp. 1074–1079. Oxford, UK: Pergamon.

Association for Computing Machinery. (1998). *The ACM Computing Classification System [1998 Version]*. Retrieved August 18, 2003, from Association for Computing Machinery web site: http://www.acm.org/class/1998/.

Athey, S., & Plotnicki, J. (1992). A comparison of Information System job requirements in major metropolitan areas. *Interface: The Computer Education Quarterly, 13*, 4, 47–53.

Athey, S., & Plotnicki, J. (1998). The evaluation of job opportunities for IT professionals. *Journal of Computer Information Systems*, *38*, 3 (Spring), 71–88.

Avant, K. C. (2000). The Wilson method of concept analysis. In B. L. Rodgers & Knafl, K. A. (Eds.), *Concept development in nursing: Foundations, techniques, and applications* (2nd ed.), pp. 55–76. Philadelphia, PA: Saunders.

Bailey, J. L., & Stefanik, G. (2001). Industry perceptions of the knowledge, skills, and abilities needed by computer programmers. In M. Serva (Ed.), *Proceedings of the 2001 ACM SIGCPR Conference on Computer Personnel Research* (pp. 93–99). New York: ACM Press.

Beise, C. M., Padget, T. C., & Canoe, F. J. (1991). Information Systems graduates: What are they really doing? In T. W. Ferratt (Ed.), *Proceedings of the 1991 Conference on SIGCPR* (pp. 14–25). New York: ACM Press.

Ben-Ari, M. (2004). Situated learning in computer science education. *Computer Science Education, 14*, 2, 85–100.

Bowker. (2004). *Ulrich's periodical directory*. Retrieved November 24, 2004, from the web site: http://www.ulrichsweb.com/ulrichsweb.

Bray, M., Brune, K., Fisher, D. A., Foreman, J., Gerken, M., Gross, J., et al. (1997). *C4 Software technology reference guide—A prototype*. Handbook, CMU/SEI-97-HB-001. Software Engineering Institute, Carnegie Mellon University.

Brooks, R. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, *18*, 6, 543–554.

Bruce, K. B., Drysdale, R. L. S., Kelemen, C., & Tucker, A. (2003). Why math? *Communications of the ACM*, *46*, 9, 41–44.

Capretz, L. (2003). Personality types in software engineering. *International Journal of Human-Computer Studies, 58*, 2, 207–214.

Carnegie Foundation. (2005). *The Carnegie Classification of Institutions of Higher Education*. Retrieved on January 7, 2005, from the Carnegie Foundation web site: http://www.carnegiefoundation.org/Classification/index.htm.

Clandinin, D. J., & Connelly, F. M. (1994) Curriculum inquiry, forms of. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 3, pp. 1316–1320. Oxford, UK: Pergamon.

Cohen, L., Manion, L., & Morrison, K. (2000). *Research methods in education* (5th ed.). London: RoutledgeFarmer.

Conover, W. J. (1999). *Practical nonparametric statistics* (3rd ed.). New York: John Wiley & Sons.

Denzin, N. K. (1994). Triangulation in educational research. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 11, pp. 6461–6466. Oxford, UK: Pergamon.

Détienne, F. (2002). *Software design—Cognitive aspects*. London: Springer.

DeZure, D. (2003). Innovations in the undergraduate curriculum. In J. W. Guthrie (Ed.), *Encyclopedia of education* (2nd ed.), Vol 2, pp. 509–514. New York: Macmillan.

Diaz-Herrara, J. L., & Hilburn, T. B. (Eds.). (2003). *Computing curriculum—Software engineering*. Public draft 1, July 17, 2003. IEEE Computer Society and Association for Computing Machinery.

Educational Resources Information Center. (n.d.). *ERIC thesaurus*. Retrieved October 10, 2002, from Educational Resources Information Center web site: http://www.ericfacility.net/extra/pub/thessearch.cfm.

Encyclopædia Britannica. (n.d.). *Merriam-Webster's Collegiate Dictionary*. Retrieved on June 21, 2005, from Encyclopædia Britannica web site: http://search.eb.com/dictionary.

Engel, G., & Roberts, E. (Eds.). (2001). *Computing curricula 2001. Computer science*. IEEE Computer Society and Association for Computing Machinery. Retrieved on October 18, 2002, from IEEE Computing Society web site: http://www.computer.org/education/cc2001/final/cc2001.pdf.

Fekete, A., & Kummerfeld, B. (2002). Design of a major in software development. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education* (pp. 73–77). New York: ACM Press.

Fincher, S., & Petre, M. (Eds.). (2004). *Computer science education research*. Lisse, Netherlands: Taylor & Francis.

Foshay, A. W. (1991). Spiral curriculum. In A. Lewy (Ed.), *The international encyclopedia of curriculum* (pp. 171–172). Oxford, UK: Pergamon.

Fry, C. (1997). Programming on an already full brain. *Communications of ACM, 40*, 4, 55–64.

Gallivan, M. J., Truex, D. P., III, & Kvasny, L. (2004). Changing patterns in IT skill sets 1988–2003: A content analysis of classified advertising. *Data Base for Advances in Information Systems, 35*, 3, 64–87.

Gade, M. L. (1991). The United States. In P. G. Altbach (Ed.), *International higher education: An encyclopedia*, Vol. 2, pp. 1081–1096. Chicago, USA: St James Press.

Geist, R., Chetuparambil, M., Hedetniemi, S., & Turner, A. J. (1996). Computing research programs in the U.S. *Communications of the ACM*, *39*, 12, 96–99.

Glatthorn, A. A., & Foshay, A. W. (1991). Integrated Curriculum. In A. Lewy (Ed.), *The international encyclopedia of curriculum* (pp. 160–162). Oxford, UK: Pergamon.

Green, G. I. (1989). Perceived importance of systems analysts' job skills, roles, and non-salary incentives. *MIS Quarterly*, *13*, 2, 115-133.

Greeno, J., & Simon, H. (1988). Problem solving and reasoning. In: R. C. Atkinson, R. J. Herrstein, G. Lindzey, & R. D. Luce (Eds.), *Stevens Handbook of Experimental Psychology*, vol. 2.

Hara, V., Hyvönen, R., Myers, D., & Kangasniemi, J. (Eds.). (2000). *Evaluation of education for the information industry*. Publications of Finnish Higher Education Evaluation Council, 8:2000. Helsinki, Finland: Edita.

Haywood, E., & Madden, J. (2000). Computer technology students—What skills do they really need? In *Proceedings of the Australasian Computing Education Conference* (pp. 139–144). New York: ACM Press.

Helsinki University of Technology. (2001a). *Degree regulations*. Retrieved October 10, 2002, from Helsinki University of Technology web site: http://www.hut.fi/Study/degree.html.

Helsinki University of Technology. (2002a). *Opetusohjelma 2002–2003* [Study program 2002–2003]. The publications of the Department of Administration 2002/7. Helsinki, Finland: Edita Prima Ltd.

Helsinki University of Technology. (2002b). *Study programme. ECTS guide 2002–2003*. The publications of Department of Administration 2002/9. Helsinki, Finland: Edita Ltd.

Helsinki University of Technology. (2003). *Report 2002*. Retrieved on May 7, 2005, from Helsinki University of Technology web site: http://www.tkk.fi/General/TKK_VSK_english.pdf.

Helsinki University of Technology. (2004). *Study Guide 2004–2005*. Helsinki University of Technology, Department of Computer Science and Engineering, Degree Program of Computer Science and Engineering. Retrieved on June 29, 2005, from Helsinki University of Technology web site: http://www.tkk.fi/Units/CSE/Studies/Study_Guide_04/majors/software_systems.htm.

Helsinki University of Technology. (2005a). *Degree structure reform*. Retrieved on April 20, 2005, from Helsinki University of Technology web site: http://kva.tkk.fi/en/Studies/DegreeStructureReform.html.

Helsinki University of Technology. (2005b). *New Degree Structure of the CSE Degree Programme*. Retrieved on April 26, 2005, from Helsinki University of Technology web site: http://www.tkk.fi/Units/CSE/Studies/new_degree_structure.htm.

Henderson, P. B., Baldwin, D., Dasigi, V., Dupras, M., Fritz, S. J., Ginat, D., et al. (2001). Striving for mathematical thinking. *SIGCSE Bulletin*, *33*, 4, 114–124.

Hingorani, K. K., & Sankar, C. S. (1995). Entry level MIS jobs: Industry expectations versus academic preparation. *Journal of Computer Information Systems*, *35*, 4, 18–27.

Hirmanpour, I., Hilburn, T. B., & Kornecki, A. (1995). A domain centered curriculum: An alternative approach to computing education. In *Proceedings of the twenty-sixth SIGCSE technical symposium on Computer science education* (pp. 126–130). New York, NY: ACM Press.

Hoc, J. M., Green, T. R. G., Samurçay, R., & Gilmore, D. J. (Eds.). (1990). *Psychology of programming*. London: Academic Press.

Hordeski, M. (1978). *Illustrated dictionary of microcomputer terminology*. USA: TAB BOOKS.

Husén, T. (1994). Research paradigms in education. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 9, pp. 5051–5056. Oxford, UK: Pergamon.

Information Technology Association of America. (2002). *Bouncing back: Jobs, skills and the continuing demand for IT workers*.

Information Technology Association of America. (2003). *2003 Workforce Survey*.

Institute of Electrical and Electronics Engineers. (1990). *IEEE Standard Computer Dictionary. A Compilation of IEEE Standard Computer Glossaries*. 610. New York.

Jenkins, D. (1991). Curriculum research. In A. Lewy (Ed.), *The international encyclopedia of curriculum* (pp. 46–51). Oxford, UK: Pergamon.

Kerola, T. (Ed.). (2005). *Opinto-opas 2005–2006* [Study guide *2005*–2006]. Helsinki University of Technology, Department of Computer Science and Engineering, Degree Program of Computer Science and Engineering. Helsinki, Finland: Picaset Ltd.

Kerola, T., Knuuttila, M., & Kujanpää, E. (Eds.). (2004). *Opinto-opas 2004–2005* [Study guide 2004–2005]. Helsinki University of Technology, Department of Computer Science and Engineering, Degree Program of Computer Science and Engineering. Helsinki, Finland: Picaset Ltd.

Kim, Y., Shim, S. J., & Yoon, K. P. (1999). Bridging the gap between practitioner-educator perceptions of key IS issues for effective implementation of IS curriculum. In *Proceedings of the 10th Information Resources Management Association International Conference* (pp. 513–518). Hershey, PA: Idea Group Publishing.

Kitchenham, B., & Pfleeger, S. L. (2002). Principles of survey research. Part 5: Population and samples. *Software Engineering Notes*, *27*, 5, 17–20.

Knapp, J. A. (1993). Information Systems: Educational offerings vs. industry needs—How well do they match? In M. R. Tanniru (Ed.), *Proceedings of the 1993 Conference on Computer Personnel Research* (pp. 18–26). New York: ACM Press.

Kuikka, M. T. (1992). Finland. In B. R. Clark & G. R. Neave (Eds.), *The encyclopedia of higher education*, Vol 1, pp. 209–217. Oxford, UK: Pergamon Press.

Lee, D. M. S., Trauth, E. M., & Farwell, D. (1995). Critical skills and knowledge requirements of IS professionals: A joint academic/industry investigation. *MIS Quarterly*, *19*, 3, 313–340.

Lethbridge, T. C. (1999). *The relevance of education to software practitioners: Data from the 1998 survey*. Technical report TR-99-06 Rev. 2. University of Ottawa, Computer Science. Retrieved on November 8, 2002, from University of Ottawa web site: http://www.site.uottawa.ca/~tcl/edrel/EdrelTechReport.doc.

Lethbridge, T. C. (2000). What knowledge is important to a software professional? *Computer*, *33*, 5, 44–50.

Lethbridge, T. C. (n.d.). *1998 education relevance survey results*. [Data file]. Retrieved on June 20, 2004, from University of Ottawa web site: http://www.site.uottawa.ca/~tcl/edrel/EdrelData1998.xls.

Lewy, A. (Ed.). (1991). *The international encyclopedia of curriculum*. Oxford, UK: Pergamon Press.

Lietz, P., & Keeves, J. P. (1994). Cross-sectional research methods. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 2, pp. 1213–1220. Oxford, UK: Pergamon.

Litecky, C., & Arnett, K. (2001). An update on measurement of IT job skills for managers and professionals. In *Proceedings of the Seventh Americas Conference on Information Systems* (pp. 1922–1922).

Litecky, C. R., Arnett, K. P., & Prabhakar, B. (2004). The paradox of soft skills versus technical skills in IS hiring. *Journal of Computer Information Systems, 45*, 1, 69–76.

Litecky, C., Prabhakar, B., & Arnett, K. (1996). MIS job market: Shaken but not stirred. *Journal of Systems Management*, *47*, 4, 51–54.

Lounasmaa, O. V. (1996). *Huippuyksikköä ei perusteta vaan se syntyy* [Center of excellence is not set up but it will born]. Opetusministeriön työryhmien muistioita, 3:1996. Helsinki, Finland: Yliopistopaino.

Maier, J. L., Clark, W. J., & Remington, W. S., Jr. (1998). A longitudinal study of the management information systems (MIS) job market. *Journal of Computer Information Systems*, *39*, 1 (Fall), 37–42.

Marciniak, J. J. (Ed.). (2002). *Encyclopedia of software engineering* (2nd ed.). Vols 1 and 2. New York: John Wiley & Sons.

Marder, J. V. (Ed.). (1991). British education thesaurus (2nd ed.) Leeds, UK: Leeds University Press.

Mawhinney, C. H., Morrell, J. S., & Morris, G. H. (1994). The IS undergraduate curriculum: Closing the gap. In *Proceedings of the Eleventh Information Systems Education Conference* (pp. 249–256).

Mawhinney, C. H., Morrell, J. S., Morris, G. J., & Helms, S. (1995). Updating the IS curriculum: Student perceptions of industry needs. In L. Olfman (Ed.), *Proceedings of the 1995 ACM SIGCPR Conference on Supporting Teams, Groups, and Learning Inside and Outside the IS Function Reinventing IS* (pp. 233–234). New York: ACM Press.

Mawhinney, C. H., Morrell, J. S., Morris, G. J., & Monroe, S. R. (1999). Updating the IS curriculum: Faculty perceptions of industry needs. In J. Prasad (Ed.), *Proceedings of the 1999 ACM SIGCPR Conference on Computer Personnel Research* (pp. 219–221). New York: ACM Press.

McCauley, R. & Manaris, B. (2002). *Comprehensive report on the 2001 survey of departments offering CAC -accredited degree programs.* Retrieved on February 11, 2004, from College of Charleston web site: http://stono.cs.cofc.edu/~mccauley/survey/report2001/CompRep2001.pdf.

McDermid, J. A. (Ed.). (1991). *Software engineer's reference book.* Oxford, UK: Butterworth-Heinemann.

McGuffee, J. W. (2000). Defining computer science. *SIGCSE Bulletin, 32*, 2, 74–76.

Milton, J., & Arnold, J. (2003). *Introduction to probability and statistics* (4th ed.). New York: McGrawHill.

Ministry of Education. (2005). *Education: University education.* Retrieved on May 7, 2005, from Ministry of Education web site: http://www.minedu.fi/minedu/education/university_edu.html.

Mitter, W. (1990). Selection mechanisms for entry to higher education. In H. J. Walberg & Haertel, G. D. (Eds.), *The international encyclopedia of educational evaluation*, pp. 408–413. Oxford, UK: Pergamon Press.

Moitus, S. (Ed.). (2000). *Yliopistokoulutuksen laatuyksiköt 2001–2003* [High quality units of higher education 2001–2003]. Publications of Finnish Higher Education Evaluation Council, 6:2000. Helsinki, Finland: Edita.

Monin, D. J., & Dewe, P. J. (1994). Skills in an environment of turbulence: A survey of Information Systems professionals in New Zealand. In J. W. Ross (Ed.), *Proceedings of the 1994 Computer Personnel Research Conference on Reinventing IS: Managing Information Technology in Changing Organizations* (pp. 208–218). New York: ACM Press.

Nakayama, M., & Sutcliffe, N. (2000). Introduction to research on IT skill issues. In *Proceedings of Americas Conference on Information Systems 2000* (pp. 1930–1934).

Nakayama, M., & Sutcliffe, N. (2001). IT skills portfolio research in SIGCPR proceedings: Analysis, synthesis and proposals. In M. Serva (Ed.), *Proceedings of the 2001 ACM SIGCPR Conference on Computer Personnel Research* (pp. 100–113). New York: ACM Press.

Nelson, R. R. (1991). Educational needs as perceived by IS and end-user personnel: A survey of knowledge and skill requirements. *MIS Quarterly*, *15*, 4, 502–525.

OCLC Online Computer Library Center. (n.d.) *Size and growth statistics.* Retrieved on June 4, 2004, from OCLC Online Computer Library Center web site: http://wcp.oclc.org/stats/size.html.

Orr, J., & von Hellens, L. (2000). Skill requirements of IT&T professionals and graduates: An Australian study. Research-in-progress. In *Proceedings of the 2000 ACM SIGCPR Conference on Computer Personnel Research* (pp. 167–170). New York: ACM Press.

Parnas, D. (1999). Software engineering programs are not computer science programs. *IEEE Software*, *16*, 6, 19–30.

Parpala, A., & Seppälä, H. (2003). *Yliopistokoulutuksen laatuyksiköt 2004–2006* [High quality units of higher education 2004–2006]. Publications of Finnish Higher Education Evaluation Council, 5:2003. Helsinki, Finland: Edita.

Prabhakar, B., Litecky, C., & Arnett, K. (1995). Boom times ahead! *Journal of Systems Management*, *46*, 1, 24–28.

Postlethwait, S. N. (1991). Module approach. In A. Lewy (Ed.), *The international encyclopedia of curriculum* (pp. 168–170). Oxford, UK: Pergamon.

Ralston, A., Reilly, E. D., & Hemmendinger, D. (Eds.). (2000). *Encyclopedia of computer science* (4th ed.). London: Nature Publishing Group.

Reichgelt, H., & Jovanovic, V. (2003). Software Management as an Information Technology knowledge area. In *Proceeding of the 4th Conference on Information Technology Curriculum* (pp. 31–36). New York: ACM Press.

Rhoades, G. (1991). Graduate education. In P. G. Altbach (Ed.), *International higher education: An encyclopedia*, Vol. 2, pp. 127–146. Chicago, USA: St James Press.

Roberts, E. (2000). Computing education and the Information technology workforce. *SIGCSE Bulletin, 32*, 2, 83–90.

Roberts, E., Cover, C. F., Davies, G., Schneider, M., & Sloan, R. (2002). Computing Curricula 2001: Implementing the recommendations. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education* (pp. 167–168). New York: ACM Press.

Rosier, M. J. (1994). Survey research methods. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 10, pp. 5854–5862. Oxford, UK: Pergamon.

Salary Services Ltd. (2004a). *ComputerWeekly. Survey of appointments data & trends.* Quarterly survey. January 2004.

Salary Services Ltd. (2004b). *TOP IT Skills–All*. Retrieved December 27, 2004, from the Salary Services Ltd. web site: http://www.salaryservices.co.uk.

Sanders, K. E., & McCartney, R. (2003). Program assessment tools in computer science: A report from the trenches. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (pp. 31–35). New York: ACM Press.

Sawyer, S., Eschenfelder, K. R., Diekema, A., & McClure, C. R. (1998). IT skills in the context of BigCo. In R. Agarwal (Ed.), *Proceedings of the 1998 ACM SIGCPR Conference on Computer Personnel Research* (pp. 9–18). New York: ACM Press.

Schmidt, W., Houang, R., & Cogan, L. (2002). A coherent curriculum. The case of mathematics. *American Educator*, Summer 2002, 1–17.

Shackelford, R., Cross, J. H., II, Davies, G., Impagliazzo, J., Kamali, R., LeBlanc, R., et al. (2005). *Computing Curricula 2005. The overview report*. Draft, April 11, 2005. Retrieved on June 21, 2005, from Association for Computing Machinery web site: http://www.acm.org/education/Draft_5-23-051.pdf.

Stanislaw, H., Hesketh, B., Kanavaros, S., Hesketh, T., & Robinson, K. (1994). A note on the quantification of computer programming skill. *International Journal of Human-Computer Studies, 41*, 3, 351–362.

Sturman, A. (1994). Case study methods. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 2, pp. 640–646. Oxford, UK: Pergamon.

Suarez, T. M. (1994). Needs assessment. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 7, pp. 4056–4060. Oxford, UK: Pergamon.

Suonuuti, H. (2001). *Guide to terminology* (2nd ed.). Nordterm: 8. Helsinki, Finland: The Finnish Centre for Technical Terminology.

Surakka, S. (2005a). Analysis of technical skills in job advertisements targeted at software developers. *Informatics in Education*, *4*, 1, 101–122.

Surakka, S. (2005b). *Sami Surakka's Doctoral thesis: Supplementary material*. Available at Helsinki University of Technology web site: http://www.cs.hut.fi/u/ssurakka/DoctoralThesis/supplementaryMaterial/index.html.

Surakka, S. (2005c). *Trend analysis of job advertisements: What technical skills do software developers need?* Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science. TKK-TKO-B156. Available at Helsinki University of Technology web site: http://www.cs.hut.fi/Publications/Reports/B156.pdf.

Surakka, S. (in press-a). Specialization in Software Systems: Content analysis of degree requirements. In *Kolin Kolistelut—Koli Calling 2005. Proceedings of the Fifth Finnish/Baltic Sea Conference on Computer Science Education*. [Accepted for publication on October 17, 2005.]

Surakka, S. (in press-b). What technical skills do software developers need? *Communications of the ACM*. [Accepted for publication on September 29, 2005.]

Surakka, S., & Malmi, L. (2005a). Delphi study of the cognitive skills of experienced software developers. *Informatics in Education*, *4*, 1, 123–142.

Surakka, S., & Malmi, L. (2005b). Need Assessment of Computer Science and Engineering Graduates. *Computer Science Education, 15*, 2, 103–121.

Tennyson, R. D. (1994). Concept learning, teaching and testing for. In T. Husén & T. N. Postlethwaite (Eds.), *The international encyclopedia of education* (2nd ed.), Vol. 2, pp. 1020–1026. Oxford, UK: Pergamon.

The Bologna Process—Towards the European Higher Education Area. (2005). Retrieved March 22, 2005, from Bologna Process web site: http://www.bologna-bergen2005.no/EN/BASIC/Pros-descr.HTM.

Todd, P. A., McKeen, J. D., & Gallupe, R. B. (1995). The evolution of IS job skills: A content analysis of IS job advertisements from 1970 to 1990. *MIS Quarterly*, *19*, 1, 1–27.

Trower, J. K. (1995). The impact of job skill requirements on I.S. curricula. In *Proceedings of the First Americas Conference on Information Systems* (pp. 597–599).

Tucker, A. B. (Ed.). (1997). *The computer science and engineering handbook*. Boca Raton, FL: CRC Press.

Tucker, A. B., Kelemen, C. F., & Bruce, K. B. (2001). Our curriculum has become math-phobic! In *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education* (pp. 243–247). New York: ACM Press.

U.S. News & World Report. (2003). *America's best graduate schools*. 2004 edition.

Valmari, A. (2001). Matematiikan tarve ohjelmistotyössä [The need for mathematics in software work]. *Arkhimedes*, 2001:2, 18–22.

Valmari, A. (2003). Software mathematics as a course topic. In *Kolin Kolistelut—Koli Calling 2003. Proceedings of the Third Finnish/Baltic Sea Conference on Computer Science Education* (pp. 101–109).

Visser, W., & Hoc, J.-M. (1990). Expert software design strategies. In J.-M. Hoc, T. R. G. Green, R. Samurçay, & D. J. Gilmore (Eds.), *Psychology of programming* (pp. 235–249). London: Academic Press.

Watson, H. J., Young, D., Miranda, S., Robichaux, B., & Seerley, R. (1990). Requisite skills for new MIS hires. *Data base*, *21*, 1, 20–29.

Wiedenbeck, S. (1985). Novice/expert differences in programming skills. *International Journal of Man-Machine Studies, 23*, 4, 383–390.

Wilhelm, W. (2001). Alchemy of the Oracle: The Delphi technique. *The Delta Pi Epsilon Journal*, *43*, 1, 6–26.

Winer, C. R. (1989). On RPG programmers. *Systems/3X & AS World*, 11/89, pp. 31, 32, 34, 38, & 42.

Wulf, C. (1991). Federal Republic of Germany. In A. Lewy (Ed.), *The international encyclopedia of curriculum* (pp. 230–233). Oxford, UK: Pergamon Press.

Yliheljo, S., Mulari, P., & Hettula, I. (Eds.). (2002). *Opinto-opas 2002–2003* [Study guide 2002–2003]. Helsinki University of Technology, Department of Computer Science and Engineering, Degree Program in Computer Science and Engineering. Helsinki, Finland: Picaset Ltd.

Young, D., & Lee, S. (1997). Corporate hiring criteria for IS graduates. *Information Systems Management*, *14*, 1, 47–53.

Zweben, S., Reichgelt, H., & Yaverbaum, G. (2005). Computing accreditation: A new criteria structure and new flexibility. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (pp. 560–561). New York: ACM Press.

Appendices

Appendix A: Author's publications related to the present thesis

A.1: Articles in journals and professional magazines

Surakka, S. (2005a). Analysis of technical skills in job advertisements targeted at software developers. *Informatics in Education*, *4*, 1, 101–122.

Surakka, S. (in press-b). What technical skills do software developers need? *Communications of the ACM*. [Accepted for publication on September 29, 2005.]

Surakka, S., & Malmi, L. (2005a). Delphi study of the cognitive skills of experienced software developers. *Informatics in Education*, *4*, 1, 123–142.

Surakka, S., & Malmi, L. (2005b). Need Assessment of Computer Science and Engineering Graduates. *Computer Science Education, 15*, 2, 103–121.

A.2: Conference proceedings

Surakka, S. (2004). Analysis of job advertisements: What technical skills do software developers need? In A. Korhonen & L. Malmi (Eds.), *Kolin Kolistelut—Koli Calling 2004. Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education* (pp. 47–56). Espoo, Finland: Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science.

Surakka, S. (in press-a). Specialization in Software Systems: Content analysis of degree requirements. In *Kolin Kolistelut—Koli Calling 2005. Proceedings of the Fifth Finnish/Baltic Sea Conference on Computer Science Education*. [Accepted for publication on October 17, 2005.]

Surakka, S., & Malmi, L. (2004). Cognitive skills of experienced software developer: Delphi study. In A. Korhonen & L. Malmi (Eds.), *Kolin Kolistelut—Koli Calling 2004. Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education* (pp. 37–46). Espoo, Finland: Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science.

A.3: Other publications

Surakka, S. (2001). *Tutkintovaatimusten määrittely. Tapausesimerkkinä Teknillisen korkeakoulun tietotekniikan koulutusohjelman 1. ja 2. vuoden opetus* [Defining degree requirements: A case study about 1st and 2nd year courses of degree programme in computer science and engineering at Helsinki University of Technology]. Licentiate's thesis, Helsinki University of Technology, Department of Industrial Engineering and Management. Available at Helsinki University of Technology web site: http://www.cs.hut.fi/u/ssurakka/lisensiaattityo/paasivu.html.

Surakka, S. (2005c). *Trend analysis of job advertisements: What technical skills do software developers need?* Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science. TKK-TKO-B156. Available at Helsinki University of Technology web site: http://www.cs.hut.fi/Publications/Reports/B156.pdf.

Appendix B: Software development strategies

The following texts are quotations from Détienne (2002, pp. 26–28):

> **Top-down vs Bottom-up**
> A solution may be developed either top-down or bottom-up, that is from the more abstract to the less abstract or vice versa. In the first case the programmer develops the solution at an abstract level and then refines it, progressively adding more and more detail. In the second case, the solution is developed at a very detailed level before its more abstract structure is identified.
>
> **Forward vs Backward Development**
> A design strategy is described as forward development when the solution is developed in direction of execution of the procedure. It is described as backward if it is developed in the direction opposite to that of the execution of the procedure.
>
> **Breadth-first vs Depth-first**
> A breadth-first strategy means developing all the elements of the solution at one level of abstraction before proceeding to the next, more detailed, level of abstraction. A depth-first strategy means that one element of the system is developed to all levels of abstraction before any other element is developed.
>
> **Procedural vs Declarative**
> The development of a solution is said to be procedural when it is the structure of the procedure that controls the solution; the solution is then based on aims or procedures. The development is said to be declarative when static properties, such as objects and roles, control the solution.
>
> **Mental simulation**
> Simulation can be used to evaluate a solution. In fact, designers often use mental simulation on a partial or complete solution at a higher or lower level of abstraction or on passages of code that they are seeking to understand. Simulation provides a way of verifying that a solution meets the desired objectives and a way of integrating partial solutions by controlling their interactions.

Appendix C: Selected institutions and degree programs

The selected institutions and degree programs are presented in Table C.1. CSE is the abbreviation for computer science and engineering, CIS for computer and information science, EE for electrical engineering, and Eng for engineering. Whether an undergraduate program was accredited is presented in the column Accredited. The rows are ordered according to the name of the institution.

Table C.1. Selected institutions and degree programs.

Institution	Funding[a]	Under-graduate	Accre-dited[b]	Graduate
Brown University	Private	BSc in CS	No	PhD
California Institute of Technology	Private	—	No	PhD in CS
Carnegie Mellon University	Private	BSc in CS	No	MSc in CS
Columbia University	Private	—	No	MSc in CS
Cornell University	Private	BSc	No	PhD in CS
Duke University	Private	BSc	No	MSc
Georgia Institute of Technology	Public	BSc in CS	Yes	MSc in CS
Harvard University	Private	BA in CS	No	MSc in CS
Massachusetts Institute of Technology	Private	BSc in CSE	Yes	M.Eng. in EE&CS
Ohio State University	Public	BSc in CSE	Yes	MSc in CSE
Pennsylvania State University, Univ. Park	Public	BSc in CS	No	MSc in CSE
Princeton University	Private	BSE in CS	No	PhD in CS
Purdue University	Public	BSc in CS	No	MSc in CS
Rice University	Private	BSc in CS	No	MSc in CS
Stanford University	Private	BSc in CS	No	MSc in CS
University of California, Berkeley	Public	BSc in CSE	Yes	PhD in CS
University of California, Irvine	Public	BSc in CS	No	MSc in CS
University of California, Los Angeles	Public	BSc in CS	Yes	MSc in CS
University of California, San Diego	Public	BSc in CS	No	MSc in CS
University of California, Santa Barbara	Public	BSc in CS	Yes	MSc in CS
University of Illinois at Urbana-Champaign	Public	BSc in CS	Yes	MSc in CS
University of Maryland	Public	—	Yes	MSc in CS
University of Massachusetts Amherst	Public	BSc in CS	Yes	MSc in CS
University of Michigan	Public	BSc in CS	Yes	MSc in CSE
University of Minnesota	Public	BSc in CS	Yes	MSc in CIS
University of North Carolina, Chapel Hill	Public	BSc in CS	No	MSc in CS
University of Pennsylvania	Private	BSc in CSE	No	MSc in Eng in CIS
University of Southern California	Private	BSc in CS	Yes	PhD in CS
University of Texas at Austin	Public	BSc in CS	No	MSc in CS
University of Washington	Public	BSc in CS	No	MSc in CS
University of Wisconsin-Madison	Public	BSc in CS	No	PhD in CS

Note. Dash (—) indicates that the degree requirements were found but the name of the degree was not.

[a]Source: Carnegie Foundation (2005).

[b]Source: Accreditation Board for Engineering and Technology (2005a).

Appendix D: Planning of Question 15

Next, the planning of Question 15 of the first questionnaire targeted at the software developers is explained in detail because the question was essential for the present thesis. The questionnaire is available on the web page of the institution (Surakka, 2005b).

It would have been easier to compare the results if Lethbridge's (2000) questionnaire had been used. However, it was not used because it missed some subjects or skills that were considered important to the present thesis. From various types of validity such as internal validity (see Cohen et al., 2000, pp. 105–112), content validity was the only one that was considered during the planning of the question.

The question was planned as group work. Three members of the group had Doctoral degrees in computer science. In addition, the author of the present thesis took part. The group met twice. During the first meeting, a brainstorming method was used: first, some words were written on self-adhesive labels, and second, these labels were organized into some categories. As a result, 42 items were classified into five categories Techniques (computer), Methods (human), Techniques/Methods, Criteria, and Metacognitive skills, where the category "Techniques (computer)" included more technical topics; "Methods (human)" included software engineering topics; and Criteria included different quality properties. The category "Metacognitive skills" included skills such as "problem solving" and the "ability to learn new technologies."

Between the first and the second meeting, the author browsed several publications in order to find different ways to categorize the question items. The publications were: five encyclopedias or handbooks (Bray et al., 1997; Marciniak, 2002; McDermid, 1991; Ralston, Reilly, & Hemmendinger, 2000; Tucker, 1997), one classification (Association for Computing Machinery, 1998), one standard (Institute of Electrical and Electronics Engineers, 1990), and two curriculum reports (Diaz-Herrara & Hilburn, 2003; Engel & Roberts, 2001). From these publications, the following influenced the planning of the question: (a) the core requirements of Computing Curricula 2001 (Engel & Roberts, 2001, p. 17) were important for adding the category "Mathematics, physics and theoretical computer science" and the items "Computer architecture" and "Computer graphics," (b) the division "Used to Support Operational Systems" vs. "Used in Operational Systems" that was used in Bray et al. (1997, p. 10) was important for choosing the category name "More technical or part of the operational system" that was used in the final question, and (c) for the different phases of software development life cycle, the IEEE standard (Institute of Electrical and Electronics Engineers, 1990, p. 186) was used.

The author prepared a memo from the first meeting and sent it to the other participants before the second meeting. Next are explained what decisions

were made during the second meeting. The category “Mathematics, physics and theoretical computer science” was added for the content validity because these subjects were often required as part of a computer science degree. The categories Criteria and “Metacognitive skills” were omitted because it was assumed that the importance of different quality dimensions might vary too strongly per project or application domain and the area of cognitive skills was unfamiliar to the group. The category name “Techniques (computer)” was changed to “More technical or part of the operational system” and the name “Methods (human)” was replaced with the categories “Software engineering (different phases of life cycle)” and “Software engineering (possible in several phases).” The items of the category Techniques/Methods were reclassified to the other categories and after this the superfluous category Techniques/Methods was removed. After these changes, the categories were the same as in the question that was used. In addition, some new items were created and classified into these four categories.

After the second meeting, the author made some smaller changes. For example, the five-point scale was changed to a the four-point scale because the four-point scale (Poor, … , Excellent) was used in some other questions.

Appendix E: Conversion of Lethbridge's results

How the data and some results of Lethbridge's (1999; 2000) research were converted from the answering scale 0–5 to the scale 1–4 is explained in this appendix. First, it is explained how the single answers were converted. Second, the conversion of the means is explained.

E.1: Single answers

Conversion of Lethbridge's single answers was necessary in order to use the Mann-Whitney test. Two alternatives were considered for conversion. In the first alternative, the equation $0.6 \cdot L + 1$ was used where L referred to Lethbridge's original value. In the second alternative, the same equation was used and the converted value was rounded to the nearest digit. Lethbridge's original values and the values of two alternative conversions are presented in Table E.1.

Table E.1. Conversion of Lethbridge's single answers.

Lethbridge's original value	Converted value, alternative 1	Converted value, alternative 2
0	1.0	1
1	1.6	2
2	2.2	2
3	2.8	3
4	3.4	3
5	4.0	4

Alternative 1 was used because it was truly monotonic. In addition, in order to test if the conversion used had an effect on the results, the results of the Mann-Whitney test were calculated using Alternative 2 as well. The values of T_1 (Conover 1999, p. 273) were a little different but the outcomes of the whole test were the same; that is, whether a difference between the two bodies of research was statistically significant.

E.2: Means

For 42 items of the questionnaires of the present research, 28 had items corresponding with items in Lethbridge's questionnaire. In these cases, the means from Lethbridge's questionnaire (scale 0–5) were converted to the scale of 1–4. Equation $0.6 \cdot L + 1$ was used for the conversion where L referred to a value in Lethbridge's scale. For 24 items, there was only one corresponding item on Lethbridge's questionnaire. For four items of the present research there were two or three corresponding items in Lethbridge's questionnaire. In these cases, Lethbridge's means were pooled. The question items of the present

research, the corresponding Lethbridge's question items and the original means, the pooled means, and the converted means are presented in Table E.2. The empty cells in the column "Pooled mean" indicate that pooling was not necessary. The rows are ordered according to the names of the subject or skill.

Table E.2. Conversion of Lethbridge's means.

Question item of the present research	**Corresponding Lethbridge's question item(s) and original mean(s)**	**Pooled mean (scale 0–5)**	**Converted mean (scale 1–4)**
Only one corresponding item:			
Artificial intelligence and knowledge engineering	Artificial Intelligence 1.28		1.77
Compilers	Parsing and Compiler Design 2.28		2.37
Computer architecture	Computer System Architecture 2.71		2.63
Computer graphics	Computer Graphics 1.92		2.15
Computer/data security	Security and Cryptography 2.24		2.34
Database management systems	Databases 3.28		2.97
Design	Software Design and Patterns 3.56		3.14
Discrete mathematics	Combinatorics 1.53		1.92
Distributed systems	Parallel and Distributed Processing 2.25		2.35
Documenting	Technical Writing 3.42		3.05
Implementing techniques of user interfaces	HCI / User Intarfaces 3.30		2.98
Internet protocols	Data Transmission and Networks 3.14		2.88
Logic (in particular, propositional and predicate logic)	Predicate Logic 2.23		2.34
Object-oriented programming	Object Oriented Concepts and Technology 3.32		2.99
Operation and maintenance	Maintenance. Reengineering and Reverse Engineering 2.82		2.69
Operating systems	Operating Systems 3.31		2.99
Physics	Physics 1.64		1.98
Project management	Project Management 3.35		3.01
Real-time systems	Real-Time System Design 2.64		2.58
Requirements	Requirements Gathering and Analysis 3.48		3.09
Software architectures	Software Architecture 3.53		3.12
Systems programming	Systems Programming 2.94		2.76
Test	Testing, Verification and Quality Assurance 3.28		2.97
Version and configuration management	Configuration and Release Management 3.26		2.96
Two or three corresponding items:			
Data structures and algorithms	Data Structures 3.74; Design of Algorithms 3.25	3.50	3.10
Mathematics for continuous systems	Differential and Integral Calculus 1.32; Differential Equations 1.10; Laplace and Fourier Transforms 1.26	1.20	1.72
Other areas of theoretical computer science (e.g., automata)	Automata Theory 2.04; Formal Languages 2.43; Graph Theory 1.98	2.15	2.29
Other telecommunications techniques than Internet protocols	Network Architecture and Data Transmission 2.81; Telephony and Telecommunications 2.34	2.58	2.55

The following conversions were problematic:

- The item "Discrete mathematics" of the present research versus Lethbridge's item Combinatorics was problematic because discrete mathematics is a broader concept than combinatorics. However, the comparison was kept because the role of mathematics is a controversial issue in computer science education and the results of the present research implied that discrete mathematics and theoretical computer science were more important than according to Lethbridge's results.
- The item "Internet protocols" of the present research versus Lethbridge's item Data Transmission and Networks was problematic because Data Transmission and Networks is a broader concept than Internet protocols. However, the comparison was kept because in Lethbridge's questionnaire

the item was under the category “Computer Engineering Software Topics.” The more hardware-related items “Network Architecture and Data Transmission” and “Telephony and Telecommunications” were under the category “Computer Engineering Hardware Topics.”

- The item Documenting of the present research versus Lethbridge’s item Technical Writing was problematic because technical writing was a broader concept than documenting. However, this conversion was used because it was assumed that for Lethbridge’s respondents, technical writing typically referred to documenting.