University Programs in Software Engineering: a Survey

Li An

A Major Report

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements
For the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

September 2001

0-612-64074-4

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the major report prepared

By: Li An

Entitled: University Programs in Software Engineering: a Survey

and submitted in partial fulfilment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

[illegible] Examiner

[illegible] Supervisor

Approved by [illegible]

Chair of Department or Graduate Program Director

SEP 19 2001 2001 [illegible]

Dr. Nabil Esmail, Dean,
Faculty of Engineering and Computer Science

ABSTRACT

University Programs in Software Engineering: a Survey

Li An

Software engineering as a field of study and practice is a relatively new discipline. Some would argue that it is not even an engineering discipline; however, a more prevalent view is that it is simply evolving and not yet mature. Although there exist curriculum guides produced for various computing curricula and some early work done to support software engineering curriculum development, there is no existing document that provides broad and comprehensive information and direction for the development of programs in software engineering.

Seeing the gap between practice and academic study, some courses in Software Engineering program from universities will be concentrated on and some summaries will be concluded for easy understanding. The main purpose of this survey is to provide detailed assistance to faculty in the design and development of programs in software engineering and other related curricula. The most important issue is after seeing this survey; it will be quite clear how to compare the software engineering program in different universities with respect to the program.

ACKNOWLEDGEMENTS

I would like to thank sincerely my major professor, Dr. Grogono, for his guidance, support, encouragement and friendship throughout my post-graduate studies.

I would also like to thank my other graduate committee member, Dr. Fancott for his assistance in preparation of major report.

Most gracious thanks are given to my parents, my parents in law, my brother Qi An, my sisters in law, Yanjun Cang, Wei Weng and Jian Weng, my brothers in law, Liyan Wang and Rongqing Hui, and all my friends, for their support and encouragement during the past several years.

Last but the most, I thank my dearest husband Jie Weng, without whom I may not have been able to reach this goal. I love you, darling!

TABLE OF CONTENTS

1. INTRODUCTION

Software engineering as a field of study and practice is a relatively new discipline. Some would argue that it is not even an engineering discipline; however, a more prevalent view is that it is simply evolving and not yet mature. In November 1997, the Working Group on Software Engineering Education and Training (WGSEET) met in Pittsburgh and discussed the need for a set of guidelines to support the design and implementation of software engineering courses and curricula. Although there exist curriculum guides produced for various computing curricula and some early work done to support software engineering curriculum development, there is no existing document that provides broad and comprehensive information and direction for the development of programs in software engineering [1].

In recent years, software plays an increasingly important and central role in all aspects of daily life. The number, size, complexity, and application domains of programs being developed will grow dramatically. Unfortunately, there are serious problems in the cost, timeliness, and quality of development of many software products. The code in consumer products is doubling every two years; it is almost the norm for software projects to overrun their planned cost and schedule. Many large-scale development projects are never completed, and many of those completed do not meet the user requirements and are of poor quality. These have placed an increasing demand for software developers who are equipped not only to deal with the scientific and technical aspects of computing, but for those who have professional education and preparation for the practice of software engineering. This includes practice related to use of software processes, measurement and analysis, front-end development methods, quality engineering, software maintenance, testing, and working as part of a team [1].

The core area of software engineering includes the following components that define the essence of software engineering [1]:

1. Software requirements

This one includes knowledge concerned with establishing a common understanding of the requirements to be addressed by a software project. It includes methods, techniques, and tools associated with the collection, analysis, specification, and review of software requirements.

2. Software design

This one includes knowledge about principles, methods and techniques for describing how a software product will be implemented such that its requirements are satisfied. It includes methods, techniques, and tools associated with the various types of design activities; architectural design, component design, interface design, data design, and algorithm design.

3. Software construction

This one includes issues related to coding style and standards, internal documentation.

code prototyping, code reuse, analysis, knowledge of language syntax, choice of implementation tools, and implementation strategies in the use of various language paradigms.

4. Software project management

This one includes issues of the creation, development, and maintenance of software projects. It also includes project management, risk analysis, project planning, project administration, and configuration management.

5. Software evolution

This one includes the knowledge and techniques necessary to understand, enhance, and modify software over time. It also includes the issues of software maintenance, extensibility, software adaptability to different environments, and software reengineering.

All of the above will be discussed and illustrated in this survey. In view of the gap between practice and academic study, some courses in Software Engineering program from universities will be emphasized and some summaries will be concluded for easy understanding. The main purpose of this survey is to provide detailed assistance to faculty in the design and development of programs in Software Engineering and other related curricula. The most important issue is that this survey, will give a clear comparison of the Software Engineering programs in different universities.

2. OBJECTIVES

The specific objectives of this survey are as the followings:

- List the related courses for software engineering program from different universities.
- Statistics of the related courses offered of software engineering program in different universities:
  - Frequency of the course most included in different universities;
  - Frequency of the course least included in different universities;
  - Commonality of the project courses;
  - Importance of mathematical courses;
  - Importance of general engineering courses;
  - Percentage of pure computer science courses or;
  - Total number of courses for quality control;
  - Importance of management courses;
  - Importance of team work for software engineering;
- Comparison between computer science program and software engineering program.
- Discussion of the breadth or depth for software engineering program from different universities.
- List the omission of software engineering programs in some significant universities.

- Summarize the current state of software engineering program education and give assistance to the development of software engineering programs.

3. BACKGROUND

3.1 Outline of Software Engineering

Since 1967, when a group of people from different disciplines met to discuss "Software Engineering" at NATO conference in Germany, computer scientists discussed software engineering as if it were a sub field of computer science. Within a computer science department, people specialize in automata theory, language design, operating systems, theorem proving, software engineering, and many other areas [2].

3.1.1 Software Engineering vs. Traditional Engineering

An Engineering approach can be adopted in the regulation of the Computer and Software disciplines. In planning for a comprehensive and credible professional regime, we must examine the similarities and differences between Software and other traditional engineering disciplines, e.g. Mechanical, Civil, Electrical, etc.

Like those in the traditional branches of engineering, Computer and Software Engineers apply theory as well as procedural knowledge in a disciplined manner to produce reliable and useful systems to end customers.

Unlike the traditional branches of engineering, there is currently little in the way of formally defined, universally accepted standards for professional practice in Computer and Software Engineering, and little in the area of commercially accepted product quality standards. Due to the advent of increasingly sophisticated technology and readily available tools, the state of the art is advancing quickly. Tools permit the production of software by even relatively inexperienced and untrained individuals. Public and commercial expectations rise rapidly when technological advances are introduced, and incorporated into various systems, but economic and technical risks rise as well. The application of sound engineering principles and discipline in the development of these systems therefore becomes increasingly important. Software and Computer Engineering disciplines also differ from the traditional engineering disciplines in the extent to which their end products are regulated. Many other Engineering disciplines have greater end-product regulatory requirements [3].

3.1.2 Software Engineering vs. Computer Science

Software Engineering requires certain theoretical underpinnings. As with traditional Engineering disciplines, Software Engineering and Computer Engineering deal with the application of science and theory in a rigorous disciplined manner, to ensure that the goals of the producing reliable and useful systems are achieved.

Computer Science is valued for development of theory and new fundamental knowledge. A significant amount of that knowledge must form part of the knowledge base of the Software / Computer Engineering. In addition to Computer Science theory, however, the Software and Computer Engineering knowledge base must incorporate mathematics, statistics, and theory from other domains such as Electrical Engineering. As with all branches of Engineering, Software / Computer Engineering must incorporate the application of science in solving design problems, and in building real systems [3].

3.1.3 The Practice of Software Engineering

An Engineer is not only one who produces the products associated with the various disciplines (Computer Systems, Electrical Systems, Automobiles, etc). Engineers are distinguished by the professional practice, the mastery of elements of that professional practice, and the activities in which they are engaged, namely those activities requiring the application of sound Engineering principles, such as:

- Analysis of Requirements;
- Allocation of Requirements and Design at various levels;
- Modeling and Analysis of Design;
- Implementation and System Construction;
- Testing, and System Analysis;
- Performance and Behavior Analysis & Modeling;
- Quality Assurance;

Figure 1 illustrates the relationship between the foundation, the complete body of knowledge and the actual practice of Software Engineering. These relationships are important, because they must all be monitored by the Computer and Software Engineering Division, in order to fulfill the mandate of enhancing the qualifications of

Engineers in the Computer and Software fields. A professional practice cannot exist without a body of knowledge, which in turn cannot exist without foundations [3].

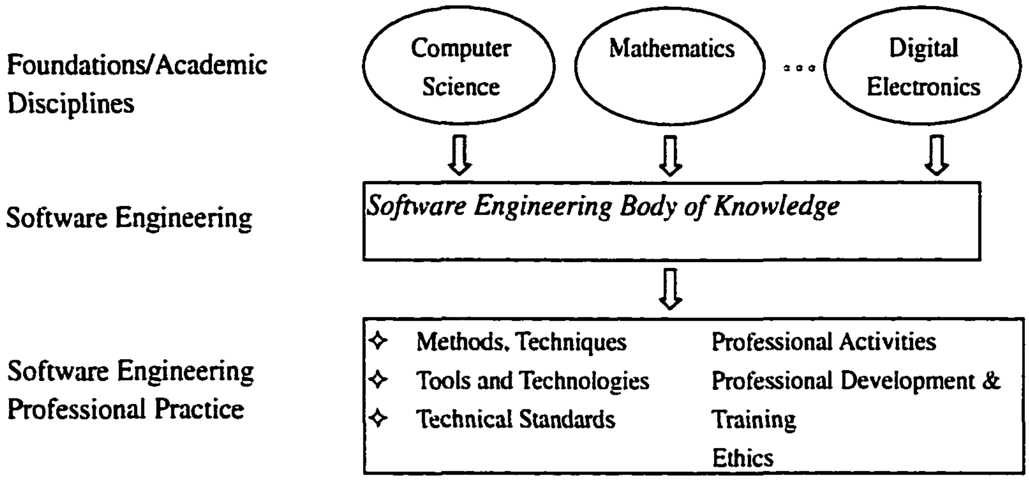

Figure 1: Relation between disciplines

3.2 General Feature of an Undergraduate Program

In undergraduate programs, engineers in all disciplines follow a largely common set of courses, in order to become well versed in fundamentals – fundamental underpinnings are required for computer and software engineering as well.

The Software Engineering programs were accredited in June 2001 which includes the following universities: Ottawa University, University of Western Ontario, and MacMaster University. The Canadian Engineering Qualification Board (CEQB), which, with Canadian Engineering Accreditation Board (CEAB) sets the qualifications for entry into the profession of engineering in Canada, has defined a Syllabus for applicants proceeding by direct examination. The Syllabus has three parts: Software Engineering, Basic Studies and Complementary Studies. The last two are common to all areas of engineering.

Most university Computer Science departments now offer some courses in Software Engineering and in a few cases these are organized as an "Option", "Specialization" or the similar. Few Electrical Engineering/Computer Engineering departments offer more than a few isolated courses on Software Engineering [4].

3.3 Related Work

3.3.1 Peter J. Denning's View

In "Computer Science and Software Engineering: Filing for Divorce" [7], Denning's principal point in the paper is that software engineers and computer scientists should not "separate" or "divorce". They must communicate and work together to make progress and bridge a communication gap in which computer scientists validate new design principal and software engineers validate new programming theories. But the fields have not yet matured enough to permit them to follow separate paths successfully.

The difficulties have always existed for the uneasy tension between software engineers and computer scientists. For engineers, they believe that certification is necessary and valuable, while for scientists most of them do not think that the certification is inevitable. Many computer scientists think that the certification will lock in minimal standards in a changing field of rising standards. Seeing the marriage such as chemical engineering and chemistry [7], a number of software engineers wish to separate from computer science and form their own department.

It is more important today than ever to keep the lines open of communication between software engineers and computer scientists. But these two fields are not matured enough to permit the two sides to follow separate paths successfully. Even in the traditional technologies such as CPU, memory, and etc seems to double nearly 18 months while cost decline. Each doubling opens new fields that form at interdisciplinary boundaries and some examples mentioned by Denning are as the following [7]:

- New computing paradigms with biology and physics including DNA, analog silicon, nano-devices, organic devices, and quantum devices
- Internet computations mobilizing hundreds of thousands of computers
- Neuroscience, cognitive science, psychology, and brain models
- Large scale computational models for cosmic structure, ocean movements, global climate, long-range weather, materials properties, flying aircraft, structural analysis, and economics
- New theories of physical phenomena by "mining" patterns from very large (multiple) datasets

Denning also notes that software engineers and computer scientists need to communicate even when they have different views and they could work together from a common interest in innovation, progress, and solution of major problems in which the practices of experimentation are crucial.

Finally, even we know that in some disciplines, separation between the theory and engineering succeeded, we should also note that it has succeeded because they have matured to some "point" in which they communicate well enough among their engineering, science, and applications branches. For computer science, it is not quite possible yet since it would have caused the communication between software engineers and computer scientists to stop. As Professor Dinning mentions, "communication, not divorce, is the answer".

3.3.2 David L. Parnas's View

In "Software Engineering, an Unconsummated Marriage" [6], Parnas's principal point is that Software Engineering education differs significantly from that of Computer Science and that, even as currently offered within Engineering faculties, the curriculum is often lacking and misguided. He comments upon the need for a strong professional grounding to lead to a (sorely needed) strong professional practice. He enumerated some of the requirements for Software Engineering education, and how these differ from the Computer Science educational approach.

At the risk of oversimplifying Parnas's point, he supports the development of rigor within the Software Engineering profession, and proposed appropriate elements of Software Engineering education to handle various elements of professional practice currently lacking in many "professionals" [5].

In Parnas's paper [6], he gives some examples to illustrate that many programmers are not educated quite well for the job they do. For example, the year-2000 problem shows that most programmers have never learned the basic principles of software design and verification. Lots of programmers ignore well accepted design principles.

Parnas mentions that communication between engineers and programmers who concentrated on software has not been effective. Engineers see programming as a trivial task and many of them refer to programming as just a skill denying that engineering principles must be applied. Many engineers do not appreciate the complexity of programming and the need for a mathematics approach, whereas many computer scientists do not appreciate the need for engineering discipline in software development [6].

He discussed the successes of chemical engineering which has been merged from chemistry with traditional engineering. Software engineering is usually treated as a branch of computer science and should wed a subset of computer science which is similar to regard chemical engineering as a branch of chemistry. Software engineers are engineers while computer scientist are scientists.

Professor concludes that software engineering is quite different from computer science. Knowing how to program is not enough to be a good software engineer. Software engineers should be professional for producing products that are fit to use which requires better understanding of the environment. Software engineers should be responsible for the usability, safety, and reliability of the products and they will be able to apply basic science and mathematics to assure the system will perform properly to the customer [5].

3.3.3 Timothy C. Lethbridge's Survey

In "The Relevance of Education to Software Practitioners: Data from the 1998 Survey" [8], Professor Lethbridge presents the complete results of a survey of software practitioners conducted during the summer and autumn of 1998. He wrote the survey to discover what knowledge is important to the participants, and to better understand their educational and training needs.

Lethbridge's survey shows that during the interaction with software practitioners and managers, it is clear that the knowledge they learned in the formal education did not match the knowledge they needed in the daily work.

In his survey, Lethbridge gets a lot of information from the software practitioners and mangers and shows us that in the daily work, they spend a lot of time performing tasks related to software requirements analysis, designing, programming, and testing. Much more time has been used for them on-the-job training about software process issues in addition to the specific software architectures which they did not learn from the previous formal education. But the practitioners and managers did not make great use of some materials they learned from universities like mathematics. This kind of imbalance means that the curricula of the university should be improved to serve the needs of students better.

As Lethbridge concludes, in North America, it is now a time to study software practice and curricula because a significant transition is occurring as traditional programs in computer science and computer engineering are being joined by programs specializing in software engineering. It is important for software engineers, designers and the people who accredit the programs about what the profession needs. Lethbridge summarized his results as follows [8]:

- Practicing programmers recognize that software development is not just a matter of programming. When asked which topics were most important and which topics they knew most about, they gave programming languages and data structures as the most important topics.
- Behind programming in the "amount-known" and "importance" questions come a cluster of topics that have to do with software design and a variety of topics that have to do with other activities like requirements, user interface design and testing as well as database design and operating systems.
- Mathematics, especially calculus, is extensively taught in computing programs but relatively little mathematics turns out to be important for software engineers in practice and it tends to be forgotten. It is needed be justified by communicating with engineers and other scientists who indeed require it.
- Giving presentations, technical writing, as well as ethics and professionalism were given prominent ranking and therefore educators should increase their emphasis to these topics.
- Engineering educators should boost the computing content in engineering curricula while corporate trainers might target their engineering new-hires for additional computing training because there is considerable difference among engineering students and computing students in terms of the amount of computing knowledge.

- Both universities and corporate training departments could improve their offerings by adding more material for software design, architecture, user interface and project management in addition to giving presentations to an audience and technical writing.

4. SOFTWARE PROGRAM

From Lethbridge's Survey, it is clear that the courses in universities are related to the working experience but only to a certain extent [8]. A question could be asked: what kind of Software Engineering program do we have right now? It is true that there are some standards to follow for the program provider. But do they really follow it or not?

The following information provides some general ideas about the Software Engineering program in different universities and in different option.

The information was obtained from the Internet and was taken carefully in order to ensure validity. Some modification has been done according to the relation of each course. For example, some courses have the same contents even though they have such different course names.

4.1 Universities Included

In this survey, about thirty universities have been taken into account. These universities are mainly in United State or Canada. Universities in other countries were not examined but they could be concluded in a future survey.

The following table gives the list of the universities involved in this survey.

Table 1 University Names for Software Engineering Program

	University Names for Software Engineering Survey	
1	Carnegie Mellon University	
2	University of British Columbia	
3	Concordia University	
4	University of Waterloo	
5	Seattle University	
6	University of Texas at Austin	
7	University of Newcastle	
8	University of Colorado at Colorado Springs	
9	George Mason University	
10	University of Western Ontario	
11	University of Houston-Clear Lake	
12	ConGESE	Carleton University
		Ottawa University
		Queen's University
		Waterloo University
		Western University
		York University
		Toronto University
13	University of Carlgray	
14	Carleton University (certificate of Software)	
15	Carleton University (bachelor of engineering)	
16	Queen's University	
17	McGill University	
18	University of Victoria	

4.2 Universities Excluded

Some popular universities have been searched when the survey was done but they do not provide a Software Engineering program according to the data found from the Internet.

It should be noted that even some very famous universities do not have a Software Engineering program. Instead, they provide some program like Computer

Engineering program or Electronic Engineering program.

In addition, some French universities have been included but, because of the language, it is hard to find and relate the courses they offer to the English university courses.

On the other hand, some very smaller, and less known universities have Software Engineering program in which there are a lot of popular and useful software engineering courses.

The following table lists universities that do not provide Software Engineering program but are nevertheless considered in this survey.

Table 2 University Names not Providing Software Engineering Program

	University Names not Providing Software Engineering Program
1	Dalhousie University
2	University of New Brunswick
3	Laval University
4	University de Montréal
5	Guelph University
6	McMaster University
7	S.P. University
8	Alberta University
9	Massachusetts Institute of Technology
10	Harvard University
11	Yale University
12	Cornell University
13	Princeton University
14	Brown University
15	University of California in Los Angeles
16	University of California in Berkeley
17	Stanford University
18	Illinois University
19	Chicago University
20	Washington University

4.3 Details for Each University offering Software Engineering Program

The following tables for each universities list the core courses for Software Engineering program. There are some other courses optional for the program that may not be included here.

4.3.1 Carnegie Mellon University

The Master of Software Engineering curriculum is designed to prepare the students to excel as a software engineer and as a project leader. At the heart of the program are five core courses, which provide the foundation for the hands-on Studio experience.

In this program, 40% of it is carried out in the Software Development Studio. The Studio provides students with a laboratory for direct application of concepts learned in coursework. It has produced a variety of software products. There are 30% Core Courses to be finished as listed in the following table. There are also quite a lot of elective courses that will not be listed here.

Carnegie Mellon University : Core Courses

Models of Software Systems
Methods of Software Development
Managing Software Development
Analysis of Software Artifacts
Architectures for Software Systems
Software Process Definition

4.3.2 University of British Columbia

The UBC Certificate in Software Engineering is delivered in a series of intensive, face-to-face courses on specific topics in software engineering. Courses employ a range of learning modes including lectures, seminars, demonstrations, participant exercises, assignments and group projects.

The curriculum consists of 102 hours of required courses, giving a solid foundation in the methods of software engineering, and 48 hours of electives, which allows students to specialize in specific facets of software engineering.

University British Columbia: Core Courses

The Software Engineering Process
Requirements Analysis and Specification
System/Software Testing
Software Architecture and Iterative Development Process
Software Project Management

4.3.3 Concordia University

Software Engineering is concerned with theories, methods, and tools required creating reliable, efficient and economical software for the needs of the information technology society. Software engineers design software using the principles and practices of engineering. They apply the theory and mathematics arising from computer science and the emerging areas of software development to the specification, design, implementation, and maintenance of large software projects.

Concordia University: Core Courses

System Hardware
System Software
Discrete Mathematics
Introduction to Programming
Programming Methodology
Document Processing
Information Systems Security
Introduction to Theoretical Computer Science
Metrics and Measurement in Software Development
Software Process
Requirements and Specifications
Software Design
Software Architecture
Quality Control
Operating Systems
Data Structures and Algorithms
Files and Databases
User Interface Design
Software Project Management
Engineering Economics and Management Principles
Design and Analysis of Algorithms
Software Engineering Design Project

4.3.4 University of Waterloo

The Software Engineering program curriculum encompasses the technical and professional background needed to engineer large complex software systems. The curriculum is project-intensive, where students learn by example and by practice. The average Software Engineering student is expected to spend 50-60 hours per week on readings, projects, group meetings, and attending lectures, practicum, and labs.

University of Waterloo: Core Courses

Linear Algebra for Engineering
Calculus 1 For Honours Mathematics
Calculus 2 For Honours Mathematics
Mechanics
Introduction to Methods of Software Engineering
Discrete Mathematics (for Engineers)
Developing Programming Principles
Electricity and Magnetism
Logic and Computation
Principles of Computer Science
Digital Circuits and Systems
Statistics (for Software Engineers)
Foundations of Sequential Programs
Digital Computers
Managerial and Engineering Economics
Algorithms and Data Structures
Software Abstraction and Specification
Dynamic Systems Analysis
Software Project Management
Control Structures
Human-computer Interaction
Software Requirements Specification and Analysis
System Performance Evaluation
Real-Time Operating Systems / Operating Systems
Software Design and Architectures
Computer Networks and Security
Database Systems
Software Testing and Quality Assurance

4.3.5 Seattle University

The Master of Software Engineering program at Seattle University is designed for working professionals. The program builds on the computing experience of its students by providing course work in a range of software engineering and computer science topics, with an emphasis on teamwork and a disciplined approach to problem solving. It offers a balanced core curriculum of technical and managerial courses, and a variety of elective streams to address areas of personal interest. The principles and techniques learned throughout the course-work are integrated into a year-long software project as the capstone experience.

Seattle University: Core Courses

Data Structures and Algorithms
Mathematical Foundations
System Software and Architecture
Technical Communication
Software Project Management
Software Quality Assurance
Requirements Analysis
Software Design
Programming Methods

4.3.6 University of Texas at Austin

The program curriculum offers the required information and skills needed to remain on the competitive edge of software engineering. Essential topics such as domain specific software, data structures, embedded systems, computer networks, data mining and project management are among the many courses offered in the program.

Students enrolled in the Executive Software Engineering (ESE) Program take a total of eight courses during the two-year program. Classes are held once a month as a Friday and Saturday combination. Students are required to take a total of 33 credit hours and complete a Master's Report during the last semester enrolled. The following are the core courses as well as the optional courses, which are offered in the ESE Program.

University of Texas at Austin: Core Courses

Introduction to Software Engineering
Experimental Software Engineering
Domain Specific Software Architectures
Empirical Studies in Software Engineering
System Engineering Program Management and Evaluation

4.3.7 University of Newcastle

Software Engineering graduates from Newcastle are trained with all aspects of building large, complex software systems for applications in most areas of society including industry, commerce, engineering, government and research.

Emphasis in the course is for a professional software engineer to be able to understand application areas and software requirements; understand and use state-of-the-art software technology; architect, design and produce high-quality software systems; and communicate ideas to customers and other software engineers.

Management and ethics also form an important part of the degree since project management, team building, project tracking and cost estimation are important skills required for a successful software engineering career.

University of Newcastle: Core Courses

Introduction to Software Engineering 1
Mathematics 111
Discrete Mathematics
Introduction to Engineering Practice
Computer Engineering I
Mathematics 112
Introduction to Software Engineering 2
The Online Society
Introduction to Algorithmics
Electrical Engineering I
Software Analysis and Verification
Theory of Computation
Operating Systems
Software Process
Computer Engineering II
Database Systems
Advanced Software Engineering
Computer Networks
Electronics 1
Object-Oriented Software Engineering
Microprocessor Systems
Introduction to Telecommunications

4.3.8 University of Colorado

This certificate program provides specialized knowledge and experience in selected areas of software development and maintenance. Emphasizing both technical and managerial aspects of building large, complex software-intensive systems, the program has two purposes: (1) it provides employees of local companies with an opportunity to enhance their software engineering skills and their chances for career advancement, and (2) it provides students currently enrolled in the Masters of Science in Computer Science (MSCS) with more in-depth knowledge in software engineering to enhance employability and career advancement. Students enrolled in the MSCS program must satisfy all the requirements for the MSCS degree and the above Certificate in Software Engineering. They receive their Masters degree and Software Engineering Certificate simultaneously.

University of Colorado: Core Courses

Telecommunications Networks
Engineering and Project Management
Software Architecture
Software Engineering Project
Technology and Human Values

4.3.9 George Mason University

The Graduate Certificate Program in Software Engineering provides knowledge, tools, and techniques to those who are working in, or plan to work in, the field of software engineering, but do not want to complete all of the requirements for a Master's degree in Software Engineering. The Certificate in Software Engineering may be pursued concurrently with any of the graduate programs in the School of Information Technology and Engineering.

George Mason University: Core Courses

Software Construction
Software Requirements and Prototyping
Software Design

4.3.10 University of Western Ontario

This program is offered the Department of Electrical and Computer Engineering. The Faculty of Engineering Science has received accreditation of the Software Engineering program in June 2001. Software Engineering is the newest branch of Engineering. The Software Engineering curriculum deals with the components of the software process and the technical skills necessary to apply that process in a systematic, disciplined and quantifiable manner. Students also acquire the management skills needed to lead a team that can engineer software and meet appropriate quality standards within specified cost and time schedules. In addition to a number of specialized topics in software engineering, the program also includes courses on fundamental topics in Electrical and Computer Engineering, and Computer Science.

University of Western Ontario: Core Courses

Applied Mathematical and Numerical Methods for Electrical Engineering
Introduction to Electrical Engineering
Electrical and Electronic Circuits
Engineering Communications
Object Oriented Design for Software Engineers
Software Engineering Laboratory
Discrete Structures for Software Engineers
Computer Science Fundamentals
Data Structures and Algorithms
Applied Statistics for Engineers and Scientists
Signal Processing
Digital Logic Systems
Microprocessors and Microcomputers
Software Engineering Design
Software Engineering
Software Engineering Design Tools
Human-Computer Interaction
Operating Systems
Computer Networks
Foundations of Computer Science
Analysis of Algorithms
Business Organization for Engineers

4.3.11 University of Houston-Clear Lake

Software engineering addresses critical issues across the life cycle of software intensive systems. The life cycle of such a system begins with the proposal to develop an application requiring the use of computing resources and continues through its development, testing, operation and maintenance until its retirement.

The Software Engineering Master's program, offered in the School of Natural and Applied Sciences at UHCL, is designed to give students an understanding of the latest technologies and tools developed to address these issues. The program provides a careful balance between theory and practice.

The program in Software Engineering leads to a Master of Science (MS) degree. Studies address the foundations, methodologies and tools used in the development and evolution of software intensive systems. By providing a careful balance between theory and practice, the program prepares students for key roles in industry, government agencies and other agencies and other institutions where software is important and provides a basis for an academic career related to software engineering.

University of Houston-Clear Lake: Core Courses

Requirements Engineering
Software Project Management
Software Construction
Software Architecture
Software Engineering Process

4.3.12 ConGESE

According to ConGESE, Software engineering is a collection of principles, models, methods, and techniques for the development, maintenance, evolution, and reuse of software that meets the functional, performance, and quality requirements in an economic and competitive manner.

The collaborative program in Software Engineering is offered as part of the Consortium for Graduate Education in Software Engineering (ConGESE).

ConGESE is a cooperative effort between ten departments in seven Ontario Universities (Carleton, Ottawa, Queen's, Toronto, Waterloo, Western, and York) and several industry partners. The two collaborating departments at the University of Toronto are Electrical and Computer Engineering (ECE) and Computer Science (CSC). This part-time collaborative program is specially structured for software professionals currently working in the field. The courses are commonly offered on-site with cooperating industrial sponsors and are designed to fit into the working schedule of professionals who might otherwise find it difficult to attend regular, on-campus courses.

ConGESE: Core Courses

Software Requirements and Specification
Software Architecture and Design
Software Quality Assurance
Software Project Management

4.3.13 University of Calgary

The MSc degree with a specialization in Software Engineering is offered jointly through the Department of Computer Science and the Department of Electrical and Computer Engineering, with funding provided by the Province of Alberta through the Access Fund.

The learning objectives of the Software Engineering specialization are to enable students to: become familiar with research and methodologies in industrially relevant areas of software engineering; carry out an applied research project in software engineering.

University of Calgary: Core Courses

Requirements Engineering
Managing the Software Lifecycle
Software Process Management
Software Quality Management
Trends in Software Engineering

4.3.14 Carleton University (Certificate of Software)

This program is designed primarily for Science and Engineering graduates (either recent graduates or graduates with professional experience) who intend to re-train in the software field and obtain employment as a computer programmer.

Carleton University (Certificate of Software): Core Courses

Introduction to Object-Oriented Programming
Abstract Data Types and Algorithms
Introduction to Systems Programming
Programming in C++
Software Engineering
Database Management Systems

4.3.15 Carleton University (Bachelor of Engineering)

Software Engineering is the driving force behind the new technologies that are transforming the way we live and work: distributed computing and the internet, multimedia applications, new telecommunication systems, etc. The phenomenal growth in computing and the related information technology industry has resulted in a tremendous demand for qualified people who can develop reliable, economical, high-quality software systems. Carleton University has responded with the new B.Eng.

in Software Engineering offered through the Department of Systems and Computer Engineering.

Software Engineering is much more than computer programming. Correctness, security, reliability, timeliness of responses, presentation in an understandable format, and meeting the true needs of the end user are all concerns of the software engineer. Languages and design notations to capture the needs and the solutions are part of the subject. Rapid design with sufficient flexibility for reusability and for future changes are additional concerns, as are due regard for safety, economy, and efficiency and speed of execution.

As offered in the Faculty of Engineering, the Software Engineering program also provides a broad foundation in basic mathematics, physical science, the engineering sciences and technology, and communications, as well as fundamental computing theory and practice, processes, methods, and tools for developing software systems, and regulatory and social issues. Emphasis is placed on developing expertise in object-oriented and real-time computer systems. A co-op option gives students access to real-world experience at one of Carleton's industry partners right at the centre of Canada's high-tech capital.

Carleton University (Bachelor of Engineering): Core Courses

Calculus for Engineering Students
Introductory Physics
Orientation to Engineering for Software Engineering Students
Introduction to Computing Communication Skills for Engineering Students
Introduction to Object-Oriented Computing
Linear Algebra for Engineering Students
Object-Oriented Software Development
Foundations of Systems Programming
Differential Equations and Infinite Series for Engineering Students
Multivariable Calculus for Engineering Students
Mechanics I
Foundations of Computer Systems
Algorithms and Data Structures
Circuits and Signals
Introduction to Discrete Structures
Chemistry for Engineering Students
Introduction to Real-Time Systems
Object-Oriented Programming and Design Laboratory
Systems Analysis and Design
Operating Systems and Databases
Database Management Systems
Engineering Economics
Probability and Statistics
Discrete Simulation and Applications
Professional Practice
Programming Languages
Real-time Concurrent Systems
Software Engineering
Engineering Project
Software Validation, Verification, and Testing
Electronic Materials, Devices, and Media
Software Product Management
Architecture of Computer Systems
Elective: Systems and Simulation or Switching Circuits

4.3.16 Queen's University

Department of Electrical and Computer Engineering offers Computer Engineering Program with Computer Engineering Option and Software Engineering Option, Electrical Engineering Program with Electrical Engineering Option and Communications Engineering Option. The discipline of Computer Engineering includes Computer Architectures, Computer Networks, Digital Systems and VLSI, Microprocessor Applications, and Software Engineering. Computer Engineering provides a basis for entering a variety of interesting and challenging careers. The shortage of skilled computer engineers is holding back Canada's high technology companies. Software Engineers mostly do applications design (multimedia, user interface), parallel and distributed systems (internet, databases), real-time and embedded systems (robotics, medical), CAD tools (VLSI).

The "Software Engineering" option focuses on the creation of well-engineered software. The fall term of the second year is the same for the two options. It is recommended that students entering Computer Engineering have a 60% average in first year (MATH, PHYS and CISC). The Computer Engineering program is challenging and requires a considerable amount of hard work.

Queen's University: Core Courses

Electric Circuits
Digital Systems
Electrical/Computer Eng. Laboratory
Introduction to Computing Science
Differential Equations for ECE's
Electronics I
Discrete Mathematics
Computer Architecture
Foundations of Info Structures & Software Eng.
Fundamentals of Electromagnetics
Microprocessor Systems
Software Specifications
Database Management Systems
Operating Systems
Technical Communications
Digital Systems Engineering
Discrete Mathematics & Logic II
Software Architecture
Algorithms I
Probability for ECE's
Project Management and Economics
Software Engineering Project
Solid State Devices

4.3.17 McGill University

Modem engineering graduates increasingly find themselves working on design projects in which software plays a primary role. Software finds use both as an integral component of products and as an integral part of the design process. Neither traditional computer science nor engineering curricula adequately teach engineering principles of software design. There is a need for the training of a new breed of engineer, in all engineering disciplines, that is capable of applying engineering design practices to the development of software-based products.

Through this Minor an engineering student can get an in-depth and comprehensive education in software engineering. It will provide a solid foundation in both basic computer science, computer programming, and software engineering practice. This minor will prepare engineers for careers in software-oriented engineering enterprises. The minor requires some basic courses in computer science in order to ensure that the students have a firm grounding in computer science and computer programming skills. The students will develop engineering software design skills through the two required courses in software engineering.

McGill University: Core Courses

Introduction to Computing II
Introduction to Computer Science
Introduction to Computer Engineering I
Introduction to Computer Engineering II
Introduction to Software Engineering
Software Engineering Practice

4.3.18 University of Victoria

In September 1998, the Faculty of Engineering at the University of Victoria introduced new degree options in Software Engineering. It will be offering:

B.Eng. Computer Engineering (Software Engineering Specialization)

B.Sc. Computer Science (Software Engineering Option)

B.Sc. Computer Science with an Area of Emphasis in Software Engineering

University of Victoria: Core Courses

Introduction to Software Engineering
Human Computer Interaction
Object Oriented Software Development
Software Development
Computers and Society
Media Applications
Ergonomics
Software Evolution
Software Architecture
System Reliability
Object Oriented Design
Embedded Systems
Network-centric Computing
Distributed Systems and the Internet
Advanced Software Development
Management OF Software Development
Software Process
Topics in Software Engineering
Directed Studies

5. ANALYSIS

In this section, the raw data illustrated above will be discussed in more detail. Some useful hints and conclusion will be drawn according to the processed data. After this discussion, the current state of Software Engineering program should be clear and it will help the university administrators to improve their program in the future.

5.1 Summary Tables

5.1.1 Number of Universities for Each Course

The following table shows the number of universities for offering that particular course. As mentioned early, some courses have been combined according to the contents they have such that it makes the analysis clearer.

Table3 Number of Universities for Each Course

Software Engineering Courses	Total No.
Algorithms and Data Structures	7
Analysis of Software Artifacts	1
Architecture of Computer systems	5
Business Organization	1
Calculus / Linear Algebra	2
Communication Systems	1
Communications Network Management	1
Computer Communications	3
Computer Engineering	2
Computer Networks	3
Database Management	6
Design & Analysis of Algorithms	3
Digital Systems/Logic Engineering	3
Discrete Mathematics / Simulation and Applications Structures	7
Distributed Operating / Processing Systems	1

Document Processing	1
Electrical Engineering	3
Electronic Materials, Circuits, Devices, Media, and Magnetism	5
Embedded Systems	1
Foundations of Computer Systems	1
Foundations of Programming Languages	4
Foundations of Sequential Programs	1
Information Systems Security	1
Introduction to Computing Science	5
Introduction to Real-Time Systems	1
Management of Software Development	3
Mathematical Foundations / Differential Equations and Infinite Series	5
Mechanics	2
Media Applications	1
Metrics and Measurement in Software Development	1
Microprocessors and Microcomputers	3
Models of Software Systems	1
Network-centric Computing	1
Object-Oriented Programming	2
Object-Oriented Software Engineering/Development	4
Object-Oriented Systems	1
Operating Systems	4
Probability and Statistics	4
Programming Languages	2
Programming Methodology	3
Project Management in Software Engineering	8
Real-time Concurrent Systems	2
Signal Processing	1
Software Architecture and Design	7
Software Design	8
Software Development	3
Software Engineering	9
Software Engineering Project	8
Software Process / Maintenance & Evolution	7
Software Product Management	7
Software Quality Engineering	5
Software Requirements	9
Software Specification, Verification, Validation and Testing	6
Solid State Devices	1
Survey of Software Design Methods / Tools	1
System Hardware	1

System Reliability	2
Systems Analysis and Design	3
Technology and Human Values	1
Telecommunications Networks	1
Theory of Logic and Computation	2
User Interface Design and Techniques	4

5.1.2 Number of Software Engineering Courses for Each University

The following table shows the number of courses offered by each university. As mentioned early, some courses have been deleted and combined according to the contents they have such that it makes the analysis clearer.

Table4 Number of Course for Each University

University Names	Total Number of SE courses
Carnegie Mellon University	6
University of British Columbia	5
Concordia University	22
University of Waterloo	23
Seattle University	9
University of Texas	4
University of Newcastle	20
University of Colorado	4
George Mason University	3
University of Western Ontario	21
University of Houston-Clear Lake	5
ConGESE	6
University of Calgary	6
Carleton University (Certificate of Software)	6
Carleton University (Bachelor of Engineering)	23
Queen's University	19
McGill University	4
University of Victoria	16

5.2 Details

From the previous section, it becomes clear that not every course has the same frequency in Software Engineering program and similarly not every Software Engineering program has the same courses.

5.2.1 Course Included in Most Programs

Among the courses in Software Engineering program, some of the courses have the higher frequency than others. The top eighteen courses are listed in the following table.

Table5 Top Seventeen Courses Offered in Software Engineering Program

Software Engineering Courses	Total No.
Algorithms and Data Structures	7
Architecture of Computer systems	5
Database Management	6
Discrete Mathematics / Simulation and Applications Structures	7
Electronic Materials, Circuits, Devices, Media, and Magnetism	5
Introduction to Computing Science	5
Mathematical Foundations / Differential Equations and Infinite Series	5
Project Management in Software Engineering	8
Software Architecture and Design	7
Software Design	8
Software Engineering	9
Software Engineering Project	8
Software Process / Maintenance & Evolution	7
Software Product Management	7
Software Quality Engineering	5
Software Requirements	9
Software Specification, Verification, Validation and Testing	6

5.2.2 Courses Included in Fewest Programs

Among the courses in Software Engineering program, some of the courses have the lowest frequency than others. The last thirty-one courses are listed in the following table.

Table6 Last Twenty-Two Courses Offered in Software Engineering Program

Software Engineering Courses	Total No.
Analysis of Software Artifacts	1
Business Organization	1
Communication Systems	1
Communications Network Management	1
Distributed Operating / Processing Systems	1
Document Processing	1
Embedded Systems	1
Foundations of Computer Systems	1
Foundations of Sequential Programs	1
Information Systems Security	1
Introduction to Real-Time Systems	1
Media Applications	1
Metrics and Measurement in Software Development	1
Models of Software Systems	1
Network-centric Computing	1
Object-Oriented Systems	1
Signal Processing	1
Solid State Devices	1
Survey of Software Design Methods / Tools	1
System Hardware	1
Technology and Human Values	1
Telecommunications Networks	1

5.2.3 Core Courses for Software Engineering

Software Engineering requires certain theoretical underpinnings. As with traditional Engineering disciplines, Software Engineering deals with the application of science and theory in a rigorous disciplined manner, to ensure that the goals of producing reliable and useful systems are achieved [3]. As stated before, the Software Engineering program should therefore contain the following core courses:

- Analysis of Requirements;
- Allocation of Requirements and Design at various levels;
- Modeling and Analysis of Design;
- Implementation and System Construction;
- Testing, and System Analysis;
- Performance and Behavior Analysis & Modeling;
- Quality Assurance.

From the survey we did, some courses are offered in most Software Engineering programs. These are shown in the table below.

Software Engineering Courses	Total No.
Algorithms and Data Structures	7
Discrete Mathematics / Simulation and Applications Structures	7
Project Management in Software Engineering	8
Software Architecture and Design	7
Software Design	8
Software Engineering	9
Software Engineering Project	8
Software Process / Maintenance & Evolution	7
Software Product Management	7
Software Requirements	9

It is clear from the table above that some of the core functionality courses in Software

Engineering are included in most programs. For example, the courses for software requirements, software design, software management, and software maintenance /evolution and software implementation have been included in most Software Engineering programs. But at the same time, some core functionality courses have not been included in most programs. For example, software validation/verification, and software quality control have not been included in most Software Engineering programs.

Meanwhile, some foundation courses have been offered in most Software Engineering program such as mathematics, data structure and algorithm.

5.3 Issues

Computer and Software Engineering are relatively new disciplines that have gained much recognition in recent years. They encompass a broad spectrum of areas including software development, hardware implementation, systems integration and infrastructure design, etc. They address the enhancement of existing systems, the development of new technologies, and the automation of manual processing using computer technologies [9].

As Parnas points out, the profession of Engineering already has a well-established scheme and structure for self–governance, one that addressed the education of its members, the proper practice of their profession, and the duties of the members to employers and society at large. Such a structure would be well suited to tackle the challenges of establishing and upholding professional standards in the domain of

Software and Computer Engineering [3].

In the following sections, some issues will be discussed in the light of our survey. After this, it will be a little clearer as to which courses should be included as the core courses for Software Engineering program. Suggestions of a model curriculum for Software Engineering program could be developed to the development of Software Engineering education programs.

5.3.1 Software Engineering is Part of Computer Science

We can assume that Software Engineering is part of Computer Science but they are not equal. To say it more theoretically, Software Engineering is a kind of intersection between Computer Science and Engineering.

From the survey we did, it can be seen that most current Software Engineering programs contain some Computer Science courses such as mathematics, programming languages, databases, data structures, and etc. But at the same time, there are many topics in Computer Science that are interesting and challenging, but have not yet found practical application. It maybe true that one can build sound software systems without any knowledge of this field. In contrast, similar remarks could be made about neural computation, many parts of "artificial intelligence", and some aspects of computability and automata theory. Some discussion of those topics must be included in Computer Science programs while a graduate of an Software Engineering program should understand aspects of communication, control theory, and maintenance that are rarely seen in Computer Science programs.

There are many advanced courses in Computer Science where the most important thing learned by the students is how to invent new algorithms or design new tools. Many of these will not be as important in the Software Engineering program where the stress will be on selecting from known algorithms and applying tools and technology that were developed by others [2].

5.3.2 Projects and Teamwork are Common

There are about 8 universities amongst those that we have studied that provide the course of doing Software Engineering project. It should be noted that the teamwork and individual contribution to the team are both the key point for one project to succeed.

Like in Computer Science program, some courses are provided for the students to practice what they learnt from class and most importantly, to let the students work as a team (group). Why is it so necessary? Everyone who has gone through the formal education could answer it directly. As we know that in the beginning of study, it is good to study and do some research yourself. When the time goes on, at a certain point group working is needed, e.g. a project which is too big for only one student to finish. At that time, individual efforts towards the team are counted on. Everyone knows that even some students are so good to do the work but they maybe not good team workers. In that case, a project may be delayed due to the failed organization of the team.

That is why many Software Engineering programs include the course for project or team project. Student could gain the knowledge and experience of team working when they work with others. While Software Engineering programs must reflect the fact that their graduates will specialize in software design, they must also equip their graduates with enough knowledge about other areas of engineering that they know when to call for help from other engineers and can work well in a team with other types of engineers.

5.3.3 Design Course Is Popular But Not Enough

There are about 8 universities from our survey providing Software Engineering program that include the design course.

Engineers are professionals whose education prepares them to use mathematics, science, and the technology of the day to build products that are important to the safety and well being of the public.

Over the past three decades, it has become increasingly common to find that software is a major component of a wide variety of products – including many traditional engineering products. Further more, software is used by engineers whey designing others, non-computerized products; the correctness of their designs depends in part on the correctness of the software they use [2].

The increasing importance of software, combined with our increased knowledge about how to build it, has resulted in a need for students who like other engineers have received an education that focuses on how to design and manufacture reliable products, but specialize in designing, building, testing, and maintaining software products.

Software Engineers are not just good programmers. An engineer is a professional who is held responsible for producing products that are fit for use. To be sure that product is fit for use, requires an understanding of the environment in which it is used. Consequently, those Software Engineers need to know many things that are not part of Computer Science.

5.3.4 Mathematics is Important

From our survey, there are about 7 universities which provide a Software Engineering program including mathematics courses.

It should be clear that mathematics is a fascinating topic and obviously important in Software Engineering programs. It is essential to emphasize the use of logic to describe properties of systems and properties of states. It is also necessary to emphasize the role of logic in checking specifications and programs for completeness and consistency. Deduction or proof would be discussed, and students would be given opportunity to use theorem-proving software, but the difference between types of logic would not get much time.

5.3.5 Types of Software Engineering Program

In our survey, many types of Software Engineering programs have been covered. Among that there are Master of Software Engineering program, Certificate of Software Engineering program and Bachelor of Software Engineering program.

Certificate of Software Engineering program takes high percentage in the entire Software Engineering program.

It makes sense that if the person already got a degree before in Computer Science or whatever major, it is better for him/her to finish fewer courses to graduate for the Software Engineering program which the Certificate program provides. Similar to the Master of Software Engineering program, fewer courses are only needed to graduate. Bachelor of Software Engineering program is not as popular as the other two because a lot of traditional universities provide some related but not the exactly same program such as Computer Engineering program or Electrical Engineering program. Of course, these programs are not the one we discussed here but they occupy many university programs and will be there in future.

At some institutions, the debate about Software Engineering was treated as a jurisdictional dispute between the Computer Science and Electrical /Computer Engineering department. The dispute is sometimes resolved by including some courses from each department in a new program. But it is not good since this type of compromise will produce graduates that are neither scientists nor engineers.

5.3.6 Management is the Key Point

About 8 universities in our survey provide a course for project management. Products containing software and products designed by software are so important to the safety and well being of the public that we must have some assurance that those practicing software engineering have graduated from a program in which the most important basic material has been covered.

5.3.7 Quality Assurance Course Necessary

Among the universities in our survey, only about 5 of them include quality assurance engineering course in the Software Engineering program. As we know, Information processing is probably the most significant industry in the world economy today and in the foreseeable future. It has expanded and continues to expand at rapid rate and this in part related to the increase in cost-effectiveness of computer hardware [10]. Cost-effectiveness has increased by a factor of about 1000 every decade and, as long as this rate of change continues, the range of tasks that can be handled most economically by computing is likely to grow rapidly. Since software is the major part of most computer systems, the field of software engineering can expect similar rapid growth.

Most information system customers are businesses. The increasing level of business competition that they face makes them acutely aware of their needs in regard to software products. Since there is also more competition among software producers, software customers are more aware of the products and services available to them. These customers, once relatively naïve and dependent on their suppliers, have become increasingly sophisticated and demanding. Software producers must understand their needs thoroughly and precisely and three of the most significant needs are level of quality required, time of delivery, and cost.

With the cost and schedule pressures discussed, it is becoming impossible to create a software product that is "generous" in the sense of simultaneously providing high quality and low expense. The management of information system development has become more complex as the size and complexity of the systems have increased. Many systems are now divided into components that are developed by different companies. A definite need exists for the clear identification of the characteristics of the system and its components and for indicators of progress during development. Quantitative measures exist for cost and schedule, but the quantification of quality has been more difficult. However, because the absence of a concrete measure for software quality generally means that quality will suffer when it competes for attention against cost and schedule.

Therefore, the course of software engineering quality is much needed to make the product good enough for the customers. If it is absent, it will be the main principle reason for the existence of quality problems in many software products.

6. CONCLUSION

Every educational program is a compromise. Any topic that is essential in one of these programs would be of potential interest to the students in the other programs. However, the limited length of university programs will force us to make choices based on priorities. In the Software Engineering program, the priority will be usefulness and applicability; for the Computer Science 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 [2].

A question could be asked about what the current state of Software Engineering education? From the previous discussion and illustration of the data we obtained, it is clear that the current state of Software Engineering program is not entirely satisfactory. There are some reasons for this:

- Many famous universities/institutes do not provide Software Engineering program, instead, they offer other kind of programs such as Computer Engineering program, which are not the same as Software Engineering program.
- Some universities provide Software Engineering program but the courses are not standard. As we mentioned before, some core courses are need like the course software requirements, software design, software implementation, software management, software quality control, software verification and validation, and so on.

- ➢ As is seen from the table created, the courses in different universities maybe have different course names but the same contents and vice versa. It should have some general course names for the core courses.
- ➢ The size of Software Engineering programs is variable, depending on the kind of programs. For example, certificate or diploma programs for Software Engineering have such distinct courses compared with bachelor programs.
- ➢ The goals for Software Engineering programs also vary between universities. While it is true that the need for software engineers continued to increase, the graduates maybe have different skills even they were enrolled in Software Engineering program.
- ➢ We mainly focused on North American universities but the reason for the choice in that, when surfing the Internet, it is really hard to find related information for other countries. Software Engineering programs should be provided as the international level in the future.

As a conclusion, it could be stated that the Software Engineering programs still need much more improvement as the information technology changes faster and faster. Software Engineers will be needed more and more as a new kind of position compared with the traditional Computer Scientists and Engineers.

Software Engineers will need to know not only the engineering stuff but also the computer theory and programming to better suit for the changed positions in the market. According to this, the formal education for Software Engineering will be necessary and compulsory. Standard curriculums and course names/contents should be provided.

7. REFERENCES

[1] Bagert, Donald J., Thomas B. Hilburn, Grey Hislop, Michael Lutz, Michael McCracken, and Susan Mengel, "*Guidelines for Software Engineering Education Version 1.0*", Technical Report, CMU/SEI-99-TR-32, ESC-TR-99-002, October 1999.

URL: http://faculty.erau.edu/hilburn/se-educ/99tr032.pdf

[2] Parnas, David Lorge, "*Software Engineering Programs are not Computer Science Programs*", Annals of Software Engineering, 19-37, June 1998.

[3] Botman, Pieter, "*Vision Document – Regulation of the Software Engineering Discipline. Draft*", 17th June 1999.

URL: http://www.apeg.bc.ca/cse/vision.pdf

[4] Calvert, Draft T., "*Computer and Software Engineering Task Force, White Paper, Academic Overview*", 16th March 1999.

URL: http://www.apeg.bc.ca/cse/academic.pdf

[5] Botman, Pieter, "*Brief Summary of Software Engineering Regulation In Other Jurisdictions (Texas, Cal., Ont.)*", 1st February 1999.

URL: http://www.apeg.bc.ca/cse/cse_overview.htm.

[6] Parnas, David Lorge, "*Software Engineering: An Unconsummated Marriage*", Communication of the ACM, September 1997/Vol.40. No.9.

URL:

http://www.acm.org/pubs/articles/journals/cacm/1997-40-9/p128-parnas/p12

8-parnas.pdf

[7] Denning, Peter J., "*Computer Science and Software Engineering: Filing for Divorce*", Communication of the ACM, August 1998/Vol.40. No.8.

[8] Lethbridge, Timothy C., "*The Relevance of Education to Software Practitioners: Data from the 1998 Survey*", University of Ottawa Computer Science Technical Report TR-99-06, July 1999.

[9] Ng, Claudia, "*Experience Qualification Guidelines for Registration through the Computer and Software Engineering Disciplines, Draft*", 29th March 1999.

URL: http://www.apeg.bc.ca/cse/experience.pdf

[10] Musa, John D., Anthony Iannino, Kazuhira Okumoto, "*Software Reliability: Measurement, Prediction, Application*", McGraw Hill, 1987.

8. APENDIXES

See attached file.

Software Engineering Courses	Carnegie Mellon University	University of British Columbia	Concordia University	University of Waterloo	Seattle University	University of Texas
Algorithms and Data Structures			V	V	V	
Analysis of Software Artifacts	V					
Architecture of Computer systems	V				V	V
Business Organization						
Calculus / Linear Algebra				V		
Communication Systems						
Communications Network Management				V		
Computer Communications					V	
Computer Engineering						
Computer Networks						
Database Management			V	V		
Design & Analysis of Algorithms			V			
Digital Systems/Logic Engineering				V		
Discrete Mathematics / Simulation and Applications Structures			V	V		
Distributed Operating / Processing Systems						
Document Processing			V			
Electrical Engineering						
Electronic Materials, Circuits, Devices, Media, and Magnetism				V		

Embedded Systems						
Foundations of Computer Systems						
Foundations of Programming Languages			V	V		
Foundations of Sequential Programs				V		
Information Systems Security			V			
Introduction to Computing Science			V	V		
Introduction to Real-Time Systems						
Management of Software	V					
Mathematical Foundations / Differential Equations and Infinite					V	
Mechanics				V		
Media Applications						
Metrics and Measurement in Software Development			V			
Microprocessors and Microcomputers						
Models of Software Systems	V					
Network-centric Computing						
Object-Oriented Programming						
Object-Oriented Software Engineering/Development						
Object-Oriented Systems						
Operating Systems			V			
Probability and Statistics				V		
Programming Languages						

Programming Methodology	V		V		V	
Project Management in Software Engineering			V	V		V
Real-time Concurrent Systems				V		
Signal Processing						
Software Architecture and Design		V	V			
Software Design			V	V	V	
Software Development						
Software Engineering				V		V
Software Engineering Project			V			V
Software Process / Maintenance & Evolution	V	V	V			
Software Product Management		V	V		V	
Software Quality Engineering			V	V	V	
Software Requirements		V	V	V	V	
Software Specification, Verification, Validation and Testing		V		V		
Solid State Devices						
Survey of Software Design Methods / Tools						
System Hardware			V			
System Reliability				V		
Systems Analysis and Design			V	V		
Technology and Human Values						
Telecommunications Networks						

Theory of Logic and Computation				V		
User Interface Design and Techniques			V	V		
Total No.	6	5	22	23	9	4

Software Engineering Courses	**University of Newcastle**	**University of Colorado**	**George Mason University**	**University of Western Ontario**	**University of Houston-Clear Lake**	**ConGESE**
Algorithms and Data Structures	V			V		
Analysis of Software Artifacts						
Architecture of Computer systems						
Business Organization				V		
Calculus / Linear Algebra						
Communication Systems						
Communications Network Management						
Computer Communications				V		
Computer Engineering	V					
Computer Networks	V			V		
Database Management	V					
Design & Analysis of Algorithms				V		
Digital Systems/Logic Engineering				V		
Discrete Mathematics / Simulation and Applications Structures	V			V		
Distributed Operating / Processing Systems						

Document Processing						
Electrical Engineering	V			V		
Electronic Materials, Circuits, Devices, Media, and Magnetism	V			V		
Embedded Systems						
Foundations of Computer Systems						
Foundations of Programming Languages						
Foundations of Sequential Programs						
Information Systems Security						
Introduction to Computing Science				V		
Introduction to Real-Time Systems						
Management of Software						
Mathematical Foundations / Differential Equations and Infinite	V			V		
Mechanics						
Media Applications						
Metrics and Measurement in Software Development						
Microprocessors and Microcomputers	V			V		
Models of Software Systems						
Network-centric Computing						
Object-Oriented Programming						
Object-Oriented Software Engineering/Development	V			V		

Object-Oriented Systems						
Operating Systems	V			V		
Probability and Statistics				V		
Programming Languages						
Programming Methodology						
Project Management in Software Engineering	V					V
Real-time Concurrent Systems						
Signal Processing				V		
Software Architecture and Design	V				V	V
Software Design		V	V	V		V
Software Development			V		V	
Software Engineering	V			V		
Software Engineering Project	V			V	V	
Software Process / Maintenance & Evolution	V	V				
Software Product Management		V			V	
Software Quality Engineering						V
Software Requirements		V	V		V	V
Software Specification, Verification, Validation and Testing	V					V
Solid State Devices						
Survey of Software Design Methods / Tools				V		
System Hardware						

System Reliability						
Systems Analysis and Design						
Technology and Human Values	V					
Telecommunications Networks	V					
Theory of Logic and Computation	V					
User Interface Design and Techniques				V		
Total No.	20	4	3	21	5	6

Software Engineering Courses	**University of Calgray**	**Carleton University (certificate of Software)**	**Carleton University (bachelor of engineering)**	**Queen's University**	**McGill University**	**University of Victoria**
Algorithms and Data Structures		V	V			
Analysis of Software Artifacts						
Architecture of Computer systems			V	V		
Business Organization						
Calculus / Linear Algebra			V			
Communication Systems			V			
Communications Network Management						
Computer Communications				V		
Computer Engineering					V	
Computer Networks						V
Database Management		V	V	V		
Design & Analysis of Algorithms				V		

Digital Systems/Logic Engineering				V		
Discrete Mathematics / Simulation and Applications Structures			V	V		V
Distributed Operating / Processing Systems						V
Document Processing						
Electrical Engineering				V		
Electronic Materials, Circuits, Devices, Media, and Magnetism			V	V		
Embedded Systems						V
Foundations of Computer Systems			V			
Foundations of Programming Languages		V	V			
Foundations of Sequential Programs						
Information Systems Security						
Introduction to Computing Science				V	V	
Introduction to Real-Time Systems			V			
Management of Software	V					V
Mathematical Foundations / Differential Equations and Infinite			V	V		
Mechanics			V			
Media Applications						V
Metrics and Measurement in Software Development						
Microprocessors and Microcomputers				V		

Models of Software Systems						
Network-centric Computing						V
Object-Oriented Programming		V	V			
Object-Oriented Software Engineering/Development			V			V
Object-Oriented Systems	V					
Operating Systems				V		
Probability and Statistics			V	V		
Programming Languages		V	V			
Programming Methodology						
Project Management in Software Engineering			V	V		V
Real-time Concurrent Systems			V			
Signal Processing						
Software Architecture and Design				V		V
Software Design						V
Software Development						V
Software Engineering		V	V	V	V	V
Software Engineering Project			V	V	V	
Software Process / Maintenance & Evolution	V					V
Software Product Management	V		V			
Software Quality Engineering	V					
Software Requirements	V					

Software Specification, Verification, Validation and Testing			V	V		
Solid State Devices				V		
Survey of Software Design Methods / Tools						
System Hardware						
System Reliability						V
Systems Analysis and Design			V			
Technology and Human Values						
Telecommunications Networks						
Theory of Logic and Computation						
User Interface Design and Techniques						V
Total No.	6	6	23	19	4	16