EVALUATING THE GAP BETWEEN THE SKILLS AND ABILITIES OF SENIOR UNDERGRADUATE COMPUTER SCIENCE STUDENTS AND THE EXPECTATION OF INDUSTRY

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Alex David Radermacher

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

May 2012

Fargo, North Dakota

North Dakota State University
Graduate School

Title

EVALUATING THE GAP BETWEEN THE SKILLS AND ABILITIES

OF GRADUATING COMPUTER SCIENCE STUDENTS AND THE EXPECTATIONS OF INDUSTRY

By

ALEX RADERMACHER

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia	
Chair (typed)	Chair (signature)
Dr. Dean Knudson	
Dr. Brian Slator	
Dr. Myron Eighmy	

Approved by Department Chair:

4/3/12	Dr. Kendall Nygard
Date	Signature

ABSTRACT

A large emphasis is placed on improving student education. Every year researchers propose new teaching methods and suggest improvements to existing methods. However, recent computer science and software engineering graduates continue to face difficulties when beginning their professional careers. This thesis reports the results of a systematic literature review that examined the knowledge deficiency among graduating students beginning work in the software industry. Following the literature review, interviews with industry managers and surveys of graduating students were undertaken to examine additional knowledge deficiencies among graduating students. The results from the literature review and the additional studies show that students lack proficiency with software development tools, knowledge of software testing, and teamwork and communication skills. The results of this research can be used by both educators and industry managers to identify potential areas of improvement for students and new hires.

ACKNOWLEDGMENTS

I would like to thank all of the numerous individuals who took time from their schedules to participate in this research. I would also like to thank Dr. Dean Knudson for assisting in contacting various industry managers and Dr. Gursimran Walia for his assistance in designing and conducting the literature review and studies that composes much of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

One of the goals of computer science and software engineering educators is to prepare undergraduate students for their future careers in industry or advanced studies. For this reason it is necessary to evaluate the education that students are receiving in order to ascertain its quality. Additionally, it is important to determine the areas in which students are lacking in order to raise awareness of these shortcomings and understand where improvement is most needed. Many researchers, educators, and industry professionals have indicated that graduating computer science students are deficient in their skills and understanding of concepts that will be important in their future careers [7, 22, 5].

In this paper, knowledge deficiency is defined as:

> *Any skill, ability, or knowledge of concept which a recently graduated computer science student lacks based on employer expectations.*

Less formally stated, knowledge deficiencies can be thought of as any knowledge or skills which graduating computer science students lack when first beginning a job in industry, but are expected to have by their employers. Knowledge deficiencies can range from a lack of understanding of computer science concepts (e.g. object orientation, UI design principles, etc.) to poor personal skills (e.g. written or oral communication) to the inability to use software development tools. In this paper, the term knowledge deficiency is used to refer to any of these areas.

Previous researchers have conducted studies to determine which skills graduating computer science and software engineering students lack. This predominantly includes case studies of newly hired software engineers [3], surveys of industry professionals [22], and empirical assessments of student knowledge [7]. Their research has found evidence in knowledge deficiencies such as testing ability [22], tool usage

[7], and concepts such as data structures [30]. However, a large number of findings related to knowledge deficiencies are based on anecdotal evidence or hearsay [5, 42]. This can make it difficult to determine whether these knowledge deficiencies are unsubstantiated claims or have been continuously shown to exist through more formal experimentation. Therefore, to gain a better understanding of knowledge deficiencies, researchers only need to focus on the knowledge deficiencies that have been empirically proven to exist in graduating computer science and software engineering students.

As an initial step to gain a better understanding of knowledge deficiencies, a systematic literature review was conducted to identify and classify the knowledge deficiencies previously identified and empirically identified by other researchers. A *systematic literature review* is a formalized, repeatable process in which researchers systematically search a body of literature to document the state of knowledge on a particular subject. The benefit of performing a systematic review, as opposed to using the more common ad hoc approach, is that it provides the researchers with more confidence that they have located as much relevant information as possible. This approach is more commonly used in other fields such as medicine to document high-level conclusions that can be drawn from a series of detailed studies [19]. To be effective, a systematic review must be driven by an overall goal. The goal of this research is to:

> *Identify and classify knowledge deficiencies in graduating computer science students for the purpose of better preparing students for their future careers.*

The remainder of the paper is structured as follows: section 2 presents related background work and motivations for undertaking a systematic literature review. The methods of the systematic literature review are detailed in Section 3. Section 4 reports the results of the literature review. Next, additional research studies performed to gain further insight into knowledge deficiencies are described in section 5. Section 6 presents a discussion of overall results of the literature review and additional research studies and a discussion of the applications of this research follows in section 7. Finally, a conclusion is presented in section 8.

CHAPTER 2. BACKGROUND WORK AND STUDY MOTIVATION FOR PERFORMING SYSTEMATIC LITERATURE REVIEW

This section outlines the motivation for conducting a systematic literature review and describes relevant background work to help provide context for the research presented in the remaining sections of this document.

2.1. Motivation

Over the past several years, North Dakota State University has been working with several industry companies to provide students with real-world project experience as part of the computer science program's capstone course [20]. One of the companies that had worked with the university on these projects voiced concerns with the instructor of the capstone course about some of the recently graduated students who had applied for positions as their company. The company mentioned that these students lacked the ability to use tools and conceptual knowledge that was necessary for employment at that company. This lead to the motivation for further examining knowledge deficiencies as well as making modifications to address these problems.

In order to better understand issues related to knowledge deficiencies, a brief ad hoc literature review was performed to determine if other researchers had studied this issue or have compiled a more extensive list of skills and abilities that graduating computer science students commonly lack. Several papers mentioned this problem, but in a majority of cases, the assertion that students possessed knowledge deficiencies was based on anecdotal evidence. Some of the papers uncovered in the review did provide good empirical evidence for the existence of knowledge deficiencies, either through large-scale surveys of industry managers and software professionals, or through studies designed to evaluate student understanding of concepts.

In order to better understand knowledge deficiencies, a systematic literature review was planned. To ensure that the review produced high-quality results, it was decided that only those papers which reported deficiencies based on empirical experimentation would be included.

The main motivation behind this research is to provide educators and employers with a better understanding of existing knowledge deficiencies. This can assist educators in improving courses to incorporate concepts and tools to which students may not have been adequately exposed. This research can also provide employers with a list of areas where recently hired graduates may require additional training. Furthermore, this research will also provide a good baseline for future investigations into knowledge deficiencies among graduating students.

2.2. Related Work

The idea of examining knowledge deficiencies in graduating computer science students is not a new idea. During the ad hoc literature review, the primary researcher of this work found different papers from the 70's and 80's [28, 29, 40]. Even then researchers were interested in ensuring that computer science education was relevant to industry needs [28] and identifying that newly hired graduates required further training to become productive at their new jobs [29].

Much research from the 90's onward was also uncovered [5, 6, 22, 34, 42]. Byrne, et al. researched the similarities and differences between Computing Curriculum 91 [33] and actual software industry managers [6]. A Microsoft employee provided a personal account of multiple areas in his career for which he felt that this college education did not adequately prepare him [5]. Ruff, et al. examined and created recommendations for the lack of communication abilities in engineers [34]. Winbladh reported on some of the misconceptions that newly hired requirements engineers possess [42]. A large survey of software developers was conducted by Lethbridge, et al. to examine areas in which they needed to increase their competency past their college education [22].

In general, there is a multitude of research directly targeted at identifying knowledge deficiencies in graduating students, but many publications are less than useful for a number of reasons. For example, publications contain anecdotal information or are based on a personal account making the identified deficiencies suspect or non-generalizable [5, 42]. Other publications are sufficiently old that the information that they contain may no longer be relevant to the current academic and industry climate [28]. Many publications are also more interested in solving perceived knowledge deficiencies than determining to what extent they actually exist [34]. Another issue is that some research focuses on other disciplines such as MIS (management information systems) and is not directly related to computer science students [40].

Despite these problems, there are several excellent publications that either explicitly examine knowledge deficiencies or can be used to derive areas where graduating computer science students are not meeting industry expectations [3, 6, 22, 26]. McGill reported on the gap between the curriculum for game development at several universities and actual industry needs [26]. The surveys of industry managers and professionals conducted respectively by Byrne, et al. and Lethbridge, et al. also provided a good deal of insight into knowledge deficiencies. Begel and Simon conducted a case study of recently hired developers at Microsoft and reported on the struggles and other issues that these developers frequently experienced [3].

It was these publications based on empirical research that presented the best information on knowledge deficiencies. Not only was there clear evidence for the existence of these deficiencies, but in many cases the results could be quantified and compared with the results from other research to determine if a particular knowledge deficiency was widespread or highly prevalent. In order to find additional publications based on empirical evidence, the authors of this research decided to conduct a systematic literature review.

CHAPTER 3. SYSTEMATIC LITERATURE REVIEW

This section describes the process used for performing a systematic literature review of knowledge deficiencies. This includes a description of the review protocol, which describes the high-level research questions, the sources to be included in the literature review, various criteria used for conducting the study, and the data that was extracted from each research paper included in the review.

3.1. Research Approach

The systematic review is based on guidelines established by Kitchenham in *Procedures for Undertaking Systematic Reviews* [17, 18]. The purpose of performing a systematic literature review is similar to that of performing any scientific experiment. Procedures are established, followed, and reported on so that other researchers are capable of replicating the work. Following a systematic review process also provides a high degree of control over the type and quality of reference works that will be included in the review and helps to provide support for the conclusions of the literature review.

In accordance with the guidelines for a systematic literature review established by Kitchenham, [18] the following steps were implemented:

1. Formulate a review protocol.
2. Execute the review based on the established protocol (identify the primary studies, evaluate those studies, extract and synthesize data from those studies).
3. Analyze the results of the review.
4. Disseminate the results of the review.
5. Discuss the findings of the review.

The review protocol specifies the research questions to be addressed, establishes a list of databases, conference proceedings, journals, etc. from which primary sources will be selected, and establishes criteria for including sources and evaluating their quality. The subsequent steps closely mirror those of any other experiment in that the protocol is executed, the results of the review are analyzed to address the research questions, and the results are presented and discussed.

3.2. Research Questions

A high-level research question (What are the knowledge deficiencies which exist among graduating computer science students and how can they be classified?) was decomposed into three more specific research questions and related sub-questions. A list of these research questions and the motivation for those questions is available in Table 1.

Table 1. Research Questions and Motivation

Research Question	Motivation
1 Is there empirical evidence of knowledge deficiencies in graduating computer science students? 1.1 What are the most common knowledge deficiencies in students? 1.2 Are there trends or changes in knowledge deficiencies in students over time?	Determine where graduating students are most commonly knowledge deficient.
2 How do the knowledge deficiencies that have been identified by academia and industry differ? 2.1 How do the methods for determining and evaluating knowledge deficiencies differ from each other? 2.2 How are the deficiencies identified by each group similar or different? 2.3 Are there other differences or similarities between identified deficiencies within academia or industry?	Determine if industry managers and college professors perceive knowledge deficiencies among graduating students similarly and how their approach and methods differ.
3 Can a classification system for the identified knowledge deficiencies be created? 3.1 Have classification systems been created or examined in previous research?	Create a taxonomy for knowledge deficiencies.

The first research question required searching for empirical evidence of knowledge deficiencies. The purpose of this question is to separate knowledge deficiencies that have been empirically validated from those which are based on anecdotal evidence or other hearsay. The second question required examining differences in knowledge deficiencies identified by industry managers and college professors. This was done to provide better understanding of how these groups identified knowledge deficiencies and to determine if there were large differences between the deficiencies identified by each group. The third question classifies the identified deficiencies based on the information gathered from questions 1 and 2. The purpose of this question was to explore additional trends and connections among identified knowledge deficiencies.

3.3. Source Selection and Search

Initially, an ad hoc review was performed in order to assist with the development of search strings and to provide a list of potential conference proceedings and journals to be manually searched. Using the results of the ad hoc review as a guideline, selection criteria were developed to establish a list of initial databases to be searched and more relevant conference proceedings or journals to be searched manually. References from primary sources were also included if they were relevant. A summary of the criteria used is described as follows:

- Based on the results of the ad hoc review and early results from the systematic review, papers from Information Technology (IT) and Information Science (IS) fields were also included for consideration if results focused on CS or SE students or programming roles.

- Databases were limited to those primarily containing publications pertaining to computer science and software engineering education.

- Journal and Conference proceedings which focus on computer science or software engineering education were searched manually.
- References from primary sources were examined if they appeared to be relevant or were used to identify the existence of knowledge deficiencies.

Table 2. Source List

Type	Sources
Databases	ACM Digital Library IEEE Explore
Conference Proceedings	ACM Special Interest Group on Computer Science Education (SIGCSE) IEEE-CS Conference on Software Engineering Education and Training (CSEET) ACM International Computing Education Research (ICER) Innovation and Technology in Computer Science Education (ITiCSE) Australasian Conference on Computing Education (ACE)
Journals	Computers and Education Computer Science Education Computing Sciences in Colleges
Other Sources	Reference list from primary studies

Based on these guidelines, the initial list of databases was created and then reviewed to remove any redundant databases. The final source list appears in Table 2.

In order to search the selected databases, a search string was developed based on the research questions identified in the previous section. The following search string contains all of the relevant keywords and synonyms used to search the databases:

(Knowledge **OR** Skill **OR** Ability **OR** Education **OR** Competence **OR** Qualification) **AND** (Gap **OR** Lack **OR** Deficiency **OR** Expectations)

OR

(New **OR** Recent) **AND** (Hire **OR** Graduates **OR** Developer **OR** “Software Engineer” **OR** Employees)

OR

(Manager **OR** Industry **OR** Job **OR** Professional **OR** “Real World”) **AND** (Needs **OR** Expectations **OR** Demands **OR** Qualifications)

In the event that the database was not able to handle the entirety of the search string, the string was broken down in order to fit and return an appropriate number of results. No more than 300 results were considered for each query. This was done after a test investigation revealed that those results after the first 300 were not related to the topic area, and that any that were appeared within the first 300 results of another search string.

3.4. Study Inclusion and Exclusion Criteria

The database search resulted in an extensive list of papers, some of which were clearly not related to the research questions. To narrow down the results from search, a set of inclusion and exclusion criteria were developed in order to assist the selection of appropriate papers to be included in the literature review. These criteria were applied in multiple steps, starting with using the title to exclude papers not related to our research focus, then proceeding based on the papers’ abstracts, and finally concluding with the papers’ contents in their entirety. A list of the criteria used can be found in Table 3 and is discussed in this section.

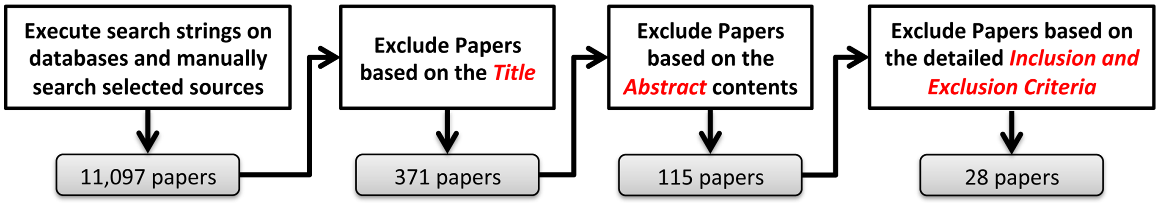
Only papers with empirically validated results were included in this review. This was done in order to provide an accurate understanding of knowledge deficiencies that exist without resorting to anecdotal evidence or other hearsay. Initially, the researchers had intended to examine knowledge deficiencies that were not validated empirically separately, but decided that having scientifically unsupported deficiencies would not add much value to the review and would only increase the amount of work to be done in performing the review.

Table 3. Study Inclusion and Exclusion Criteria

Inclusion Criteria	Exclusion Criteria
Publications that directly address any of the research questions Publications that contain empirical results Publications that discuss knowledge deficiencies in computer science students	Publications that are not in English. Publications that do not focus on graduating students or newly hired software engineers. Publications before 1995. Publications that do contain results about computer science students or results that can be generalized to computer science students. Publications that contain unclear or ambiguous results.

Because the field of computer science progresses at a faster rate than many other fields it was necessary to eliminate older studies which are no longer relevant. 1995 was chosen as a cut-off point for several reasons. In 1991, the ACM and IEEE released Computing Curricula 1991, an updated recommendation for university computer science curriculum [33], the first update since the previously published curriculum recommendations made in 1978 [1]. By 1995, institutions should have had ample time to modify their curriculums in response to the new recommendations and graduate students who received their education based on these recommendations. Also, using 1995 as a cut-off point allows a large window to better examine and understand the trends in knowledge deficiencies among graduating students.

Because early results from the systematic literature review were not yielding a large number of quality results, the researchers decided to extend the search to include publications from the information technology and information systems fields after finding several possible candidate papers from those fields. IT and IS papers were included if was clear that the results focused on programming, software development, or other common roles performed by computer science and software engineering graduates. Results which could be generalized to include computer science students


Execute search strings on databases and manually search selected sources
11,097 papers
Exclude Papers based on the Title
371 papers
Exclude Papers based on the Abstract contents
115 papers
Exclude Papers based on the detailed Inclusion and Exclusion Criteria
28 papers
Figure 1. Literature Review Execution Results

were also considered for inclusion.

3.5. Study Execution

After executing all search strings on all databases and manually searching through all selected sources, 11,097 papers were found. After applying the inclusion and exclusion criteria based on the title of those papers, 371 remained. The abstracts for these papers were read and also subjected to the same inclusion and exclusion criteria, leaving 115 papers remaining. Each of the remaining papers was read in its entirety. After reading each of the selected papers, only 28 remained.

These 28 papers were published in 11 leading journals and 5 conferences in computer science education. A distribution of the paper sources can be found in Table 4, including the number of papers from each source along with references.

3.6. Quality Assessment

After selecting the final list of papers, a quality assessment was performed to assess the study design, bias, validity, and generalizability of results for all the publications. Each paper was read in its entirety and evaluated using a quality assessment checklist that was developed in accordance with the guidelines published by Kitchenham et al [17, 18]. Table 5 contains the questions used to construct the checklist and is discussed in the following subsections.

3.6.1. Quality Assessment Methodology

The first question is used to address whether or not the purpose of the study was to identify knowledge deficiencies in graduating computer science students or

Table 4. Paper Distribution

Source	Count	References
Journal of Computing Science in Colleges	5	[37, 10, 9, 41, 13]
ACM Special Interest Group on Computer Science Education	4	[25, 8, 4, 39]
Australian Computing Education Conference	3	[15, 14, 21]
IEEE Conference on Software Engineering Education & Training	2	[22, 7]
ACM Internation Computing Education Research	2	[38, 3]
Journal of Systems of Software	2	[11, 23]
Computers and Education	1	[6]
Internation Conference on Foundations of Digital Games	1	[26]
ACM Special Interest Group on Information Technology Education	1	[27]
Information Systems Frontiers	1	[12]
ACM Special Interest Group on Computer Personnel Research	1	[2]
Journal of Computer Information Systems	1	[43]
Communications of the ACM	1	[16]
IEEE Computer	1	[24]
Australasian Conference on Information Systems	1	[36]
ACM Conference on Innovation and Technology in Computer Science Education	1	[30]

newly hired graduates. Also, the question considers whether or not researchers used a similar definition of knowledge deficiencies in their research. The second question assesses whether or not the primary focus of the research is graduating or recently graduated computer science and software engineering students or industry professionals responsible for hiring those individuals. Similarly, the third question addresses whether or not the study subjects are a good representation of current or future industry professionals and if there were a sufficient number of subjects in the study to suggest that the results are generalizable. The fourth question is used to address whether or not the identified knowledge deficiencies are relevant to computer science or software engineering careers. Finally, the fifth question assesses additional quality attributes reported in the study that provide additional support and validity to the results.

Table 5. High Level Quality Assessment Questions

#	Question
1	Was a purpose of the study to identify knowledge deficiencies and is the definition of deficiencies used in the study similar to the one used in this paper?
2	To what degree do the published results focus on senior computer science students, recent graduates with degrees in computer science or software engineering, or recently hired industry professionals; or hiring personnel or industry managers responsible for hiring recently graduated students?
3	Are the study subjects a good representation of the industry professionals (future or current) and were there a large number of subjects to support the results?
4	Are the identified knowledge deficiencies well-defined and primarily relevant to computer science and software engineering students or careers?
5	Does the study use and report experimental procedures that help to further assess quality or report and address validity threats to suggest quality research was performed?

Table 6 contains the quality checklist questions and their mapping to the high-level questions shown in Table 5. To develop this list, the authors of this paper used recommendations from Kitchenham [18]. All check-list items were treated as binary data (when applicable to a given paper) in order to simplify the quality assessment process. A summary of the quality assessment results is provided in the following subsection.

3.6.2. Quality Assessment Discussion and Additional Details

Because the quality assessment checklist items were binary data, a hard line was drawn for whether or not a paper met the required criteria for each item. For check-list items related to high-level question 1, it was sufficient for the paper to explicitly mention or contain data that measured the performance of newly hired graduates against an employer's expectations. Similarly, it was sufficient to satisfy high-level question 4 check-list items if deficiencies such as programming were contained in the

Table 6. Quality Assessment Checklist Items

High-Level Question	Checklist Item
2	Are the study subjects primarily CS/SE students, recent graduates, or managers or hiring personnel for job positions commonly filled by CS/SE graduates?
1	Is a purpose of the study to identify knowledge deficiencies?
1	Does the study measure knowledge deficiencies similarly to this research? (i.e. areas where new hires are lacking)
4	Are the identified knowledge deficiencies (or other issues) relevant to roles commonly performed by CS/SE graduates?
3	Is the study population a good representation of the industry or future industry professionals?
5	Was a control group used in the study?
5	Is a response rate given for the study?
3	Was the sample size adequate for the study or otherwise justified?
4	Does the study contain self-describing knowledge deficiencies or provide descriptions for non-obvious knowledge deficiency categories?
5	Does the study describe or justify statistical methods used for data analysis?
5	Are scoring or ranking systems used in the study described?
5	Does the study list and address potential validity threats of the study?
5	Does the study present most or all of its data or findings?
5	Does the study attempt to analyze or provide a detailed discussion of its findings?

paper or if the paper explicitly mentioned computer science or software engineering students. Check-list items related to high-level question 2 were the most stringent. Although all papers contained results from computer science students or software industry professionals, only those papers which specifically focused on senior-level computer science students, recently hired graduates, or hiring personnel and managers were counted as meeting the criteria. For high-level question 3 check-list items, it was sufficient to meet the requirements if the subjects came from multiple universities, institutions, etc. and if a large sample population (100 for surveys, 40 for experiments) was used. In case the paper did not meet these size requirements, it was considered

	Is a purpose of the study to identify knowledge deficiencies?	Does the study measure knowledge deficiencies similarly to this research? (i.e. areas where new hires are lacking)	Are the study subjects primarily CS/SE students, recent graduates, or managers or hiring personnel for job positions commonly filled by CS/SE graduates?	Is the study population a good representation of the industry or future industry professionals?	Was the sample size adequate for the study or otherwise justified?	Are the identified knowledge deficiencies (or other issues) relevant to roles commonly performed by CS/SE graduates?	Does the study contain self-describing knowledge deficiencies or provide descriptions for non-obvious knowledge deficiency categories?	Was a control group used in the study?	Is a response rate given for the study?	Does the study describe or justify statistical methods used for data analysis?	Are scoring or ranking systems used in the study described?	Does the study list and address potential validity threats of the study?	Does the study present most or all of its data or findings?	Does the study attempt to analyze or provide a detailed discussion of its findings?
[6]	y	y	y	y	y	y	y	-	-	-	y	n	y	y
[11]	n	n	n	y	y	y	y	-	y	-	y	n	y	y
[22]	y	y	n	y	y	y	y	-	n	-	y	y	y	y
[38]	y	n	y	y	y	y	y	y	y	y	y	n	y	y
[8]	n	n	y	y	y	y	y	n	-	-	y	n	y	y
[25]	n	n	y	y	y	y	y	n	-	-	y	n	y	y
[26]	y	y	y	y	y	n	y	-	n	y	y	y	y	y
[27]	y	n	y	y	n	y	y	-	n	-	-	n	n	y
[12]	y	n	n	y	y	n	y	-	n	-	y	y	y	y
[2]	n	n	n	n	y	y	y	-	n	-	y	y	y	y
[3]	y	y	y	y	y	y	y	-	-	-	-	y	y	y
[4]	y	y	y	y	y	y	y	-	-	-	-	y	y	y
[43]	y	y	n	n	n	y	y	-	y	n	y	n	y	y
[39]	n	n	n	n	n	y	y	-	n	-	y	n	y	y
[16]	n	y	y	n	n	y	y	y	y	-	y	n	y	y
[37]	y	y	n	y	y	y	y	-	n	-	n	n	n	y
[10]	y	y	n	y	y	y	y	-	y	-	y	n	n	y
[9]	n	n	y	y	y	y	y	-	n	-	y	n	y	n
[36]	y	y	n	y	n	y	y	-	n	-	y	n	n	y
[41]	y	n	y	n	y	y	y	-	-	-	y	n	n	y
[13]	y	y	y	n	y	y	y	-	-	-	n	y	n	y
[15]	y	y	y	y	n	y	y	-	y	-	y	n	y	y
[14]	y	y	n	y	y	y	y	-	y	-	n	n	n	y
[21]	y	y	y	y	y	y	y	-	n	n	n	n	n	y
[30]	y	n	y	y	y	n	y	-	-	y	y	n	y	y
[23]	n	y	n	y	y	y	y	-	n	-	y	n	y	y
[24]	y	y	n	y	y	y	y	-	n	-	y	n	y	y
[7]	y	n	y	y	n	y	y	-	-	y	y	n	y	y

Figure 2. Quality Assessment Results

good if proper justifications for the generalizability of the results were provided.

In general, the results of the quality assessment were good. Only 2 of the 28 selected papers satisfied 3 or fewer check-list items for high-level questions 1 through 4 and no paper satisfied fewer than 2 check-list items. Similarly, only 6 papers satisfied fewer than 50% of applicable check-list items related to high-level question 5.

3.7. Data Extraction and Synthesis

Once all of the papers had undergone quality assessment, data extraction was performed on all papers. A data extraction form was developed to ensure consistent extraction across all papers. Table 7 lists the fields of the data extraction form as well as a description of each field.

Table 7. Data Extraction Forum

Field	Description
Study ID	A unique identification number for each publication.
Title	The title of the publication.
Date	The date when the publication was published.
Publication Type	The type (e.g. journal article, conference paper, technical report) of the publication.
Bibliographic Reference	Citation including author, year, title, and source of the publication.
Type of Study	The type of research (e.g. survey) performed in the study.
Number of Participants	The number of participants in the study.
Identified Knowledge Deficiencies	The knowledge deficiencies identified by the publication.
Knowledge Deficiency Inclusion Criteria	The criteria used to select which knowledge deficiencies were extracted from this publication.
Study Point of View	The point of view of the study. This can be managers, instructors, students, etc.
Knowledge Deficiency Taxonomy	The knowledge deficiency classification system (if any) used by the publication.

Consistent with the process followed in previous systematic reviews (e.g., [19]), the primary researcher reviewed all papers and extracted data. Then, a second researcher independently reviewed and extracted data from a sample of the papers. Researchers then compared the data extracted by each reviewer for consistency. The researchers found that data had been consistently extracted from the sample of papers, suggesting that the second author did not need to review the remainder of papers in detail and that the information extracted by the primary author was sufficient. The data extracted from all papers was synthesized to answer each question as described in the following section.

CHAPTER 4. SYSTEMATIC LITERATURE REVIEW RESULTS

This section reports the findings of the systematic literature review and provides answers for the research questions listed in Section 3.2

4.1. Question 1: Is There Empirical Evidence of Knowledge Deficiencies in Graduating Computer Science Students?

A review of the literature indicates that there was empirical evidence of knowledge deficiencies in graduating computer science students. Knowledge deficiencies

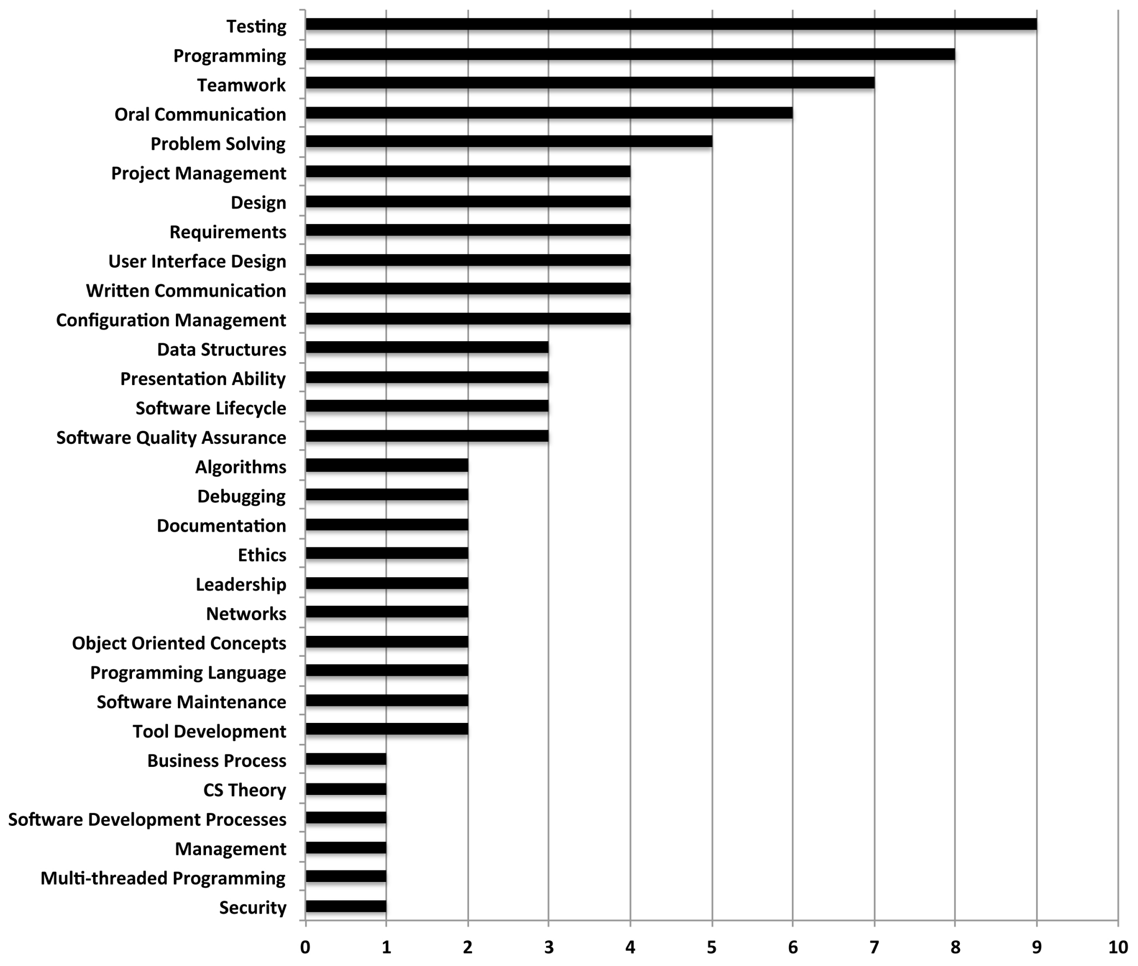


Figure 3. Knowledge Deficiencies Identified in Literature Review

identified as most prevalent or most important in each paper were placed into thirty different categories. Of these thirty categories, twenty-five were identified in more than one paper. Figure 3 shows the frequency of knowledge deficiencies across all papers. A total of twenty-eight papers were used to address this question and related sub-questions.

This section discusses the findings regarding the eleven most common knowledge deficiencies identified in the selected papers. A detailed description of each knowledge deficiency category along with a summary of study settings and major findings of each of the twenty-eight papers is provided in Appendix A.

4.1.1. Question 1.1: What Are the Most Common Knowledge Deficiencies in Students?

The review identified several common knowledge deficiencies. The most common deficiencies (i.e., the deficiencies appearing in at least 4 papers), along with additional details, are listed below.

- *Software testing* was the most commonly found knowledge deficiency. Not every paper identified specific areas of software testing where graduating students were deficient, but those that did cited system testing as a specific type of testing [12] and another paper indicated that students lacked the ability to use test-coverage tools effectively [7]. One paper also pointed out that lack of expertise was viewed as a major factor in baring the adoption of testing tools and methodology in software companies [11].

- *Programming ability* was the second most frequently found knowledge deficiency. Most papers did not mention any specific types of programming rather merely categorized the deficiency as general programming ability or something similar. Two papers were more specific, mentioning procedural and multi-threaded programming as knowledge deficiencies [26, 39]. Language independence was also mentioned in one paper [27].

- *Teamwork* was the third most listed knowledge deficiency. If additional information was provided, the ability to get along with others or to check one's own ego were listed as important facets of teamwork [16]. Having experience working as part of a team or group was also cited as being important [27]. Another paper stated that it was also necessary to be able to work as part of cross-disciplinary teams [36].

- *Oral communication* was also identified in several papers. The ability to communicate with customers and listening skills were cited as important parts of oral communication when detailed explanations were given [2, 14]. Another paper expressed that recently hired graduates also struggled to adequately communicate when they needed assistance with an issue [4].

- *Written communications* was also identified as a common knowledge deficiency, though not as frequently as oral communication. Most papers did not provide specific examples, but two papers specifically mentioned technical writing as a common knowledge deficiency [22, 15]. Some papers also specifically mentioned that graduating students lacked the ability to write and produce documentation [6, 37].

- *Requirements gathering and analysis* was a commonly identified knowledge deficiency. In some papers, more emphasis was placed on a particular sub-area such as requirements elicitation [37]. Another paper classified requirements specification as a knowledge deficiency [41].

- *Problem solving ability* was another commonly identified knowledge deficiency. Most papers did not provide additional details, but one paper indicated that students lacked the ability to generate alternative solutions to problems [13].

- *Software design* was also frequently identified as a knowledge deficiency. Multiple papers indicated that students tended to produce very minimalist designs and that they often left out important details or that portions of their design were largely incomplete [25, 8]. Another paper also indicated that graduating students lacked the ability to describe their designs using formal or semi-formal modeling techniques [41].

- *Project management* was also identified as a knowledge deficiency in multiple papers [22, 14, 23, 24]. None of the reviewed papers provided any indication of whether recently graduated students lacked project management skills necessary for their jobs or if they merely did not have an adequate understanding of project management. As such, it is not entirely clear exactly how graduating students are considered knowledge deficient in this area.

- *User interface design* was a commonly listed knowledge deficiency. One paper specifically identified object oriented user interface design as a knowledge deficiency, whereas other papers did not provide additional distinction about the type of user interfaces [27].

- *Configuration management* was also a knowledge deficiency that was identified frequently. One of the papers indicated that several recently hired developers lacked experience using the configuration management tools used by the company [3].

Out of the remaining 19 knowledge deficiencies, seven appeared in three papers, eight of them appeared in two papers, and each of the remaining six appeared in a single paper. The complete list of all the knowledge deficiencies identified during the review along with their sources is listed in Appendix B. This information of the knowledge deficiencies served as input to the knowledge deficiency taxonomy.

4.1.2. Question 1.2: Are There Trends or Changes in Knowledge Deficiencies of Students over Time?

Because many of the reviewed papers tend to focus on one specific area and over half of the reviewed papers were published in the last five years, it is difficult to establish whether or not there are significant trends in knowledge deficiencies over a period of time for many of the categories. Table 8 provides a list of identified knowledge deficiencies along with their distribution across three periods of time. These time periods are somewhat arbitrary, but each spans six years and the ACM published their updated curriculum recommendations in 2001, suggesting a reasonable breakpoint for evaluating potential trends in knowledge deficiencies.

Software testing was listed frequently in both reviewed papers published recently and those published over a decade ago. A larger percentage of the papers from 1995 – 2000 list software testing a deficiency; however this is largely a side effect of low number of papers from this time period that focused on knowledge deficiencies among students. Additionally, recently published research papers have indicated that students still lack an understanding of how to effectively and efficiently conduct software testing [7]. Another possible explanation is that industry expectations of the testing ability of newly hired graduates are less stringent than they once were.

Object oriented concepts and ethics are also deficiencies that only appear in papers from the 1995 – 2000 time period [23, 24]. The first is likely due to the industry going through a transition where many companies began using OO languages. It is likely that many of the developers at the time did learn a substantial amount of object oriented concepts in their education. Since that time, the ACM/IEEE curriculum recommendations have placed a much higher emphasis on teaching object oriented languages and concepts to CS majors [31].

One growing trend that has been identified is that students lack the ability to

Table 8. Knowledge Deficiencies and Distribution Across Time

Deficiency	Total	1995-2000	2001-2006	2007-2012
Testing	9	[6, 22, 23, 24]	[36]	[11, 12, 41, 7]
Programming	8	[15]	[2, 10, 9]	[27, 12, 39, 37]
Teamwork	7	[6]	[9, 36]	[27, 3, 4, 16]
Oral Communication	6		[2, 9, 14]	[12, 3, 4]
Problem Solving	5		[2, 43, 36, 13]	[41]
Configuration Management	4	[22]	[43]	[3, 4]
Design	4	[36]	[8, 41]	[25]
Project Management	4	[22, 23, 24]	[14]	
Requirements	4	[22, 23]	[36]	[37]
User Interface Design	4	[22, 23, 24]		[27]
Written Communication	4	[6, 22]	[9, 36]	
Data Structures	3		[9, 30]	[16]
Software Debugging	3		[2]	[3, 4]
Software Lifecycle	3	[6]	[43]	[27]
Networking	3		[43, 10]	[37]
Presentation Skills	3	[6, 23, 24]		
Software Quality Assurance	3	[6, 23, 24]		
Tool Development	2		[36]	[26]
Algorithms	2		[9]	[39]
Documentation	2	[6]		[37]
Ethics	2	[22, 23]		
Programming Languages	2			[26, 16]
Leadership Ability	2	[6]		[26]
Software Maintenance	2	[22]	[43]	
Object Oriented Concepts	2	[23, 24]		
Experience (Interships)	2		[9]	[21]
Business Processes	1		[14]	
CS Theory	1			[39]
Management Skills	1	[22]		
Multi-threaded Programming	1			[26]
Software Development Processes	1			[37]
Security	1		[43]	

effectively use software tools intended to assist in software development (i.e. code coverage tools, debuggers, etc.) [41, 7]. Papers from each period of time specifically mentioned configuration management as a knowledge deficiency [22, 3, 4, 43]. The ability to debug programs has also been reported as an emerging knowledge deficiency [2, 3]. In some cases, this was identified as a lack of understanding or experience with using debugging software [4].

4.2. Question 2: How Do the Knowledge Deficiencies that Have Been Identified by Academia and Industry Differ?

Papers that were published from the point of view of hiring managers or other software professionals differed from those published by college professors. In general, the papers from industries' point of view tended to focus on a broad range of knowledge deficiencies, whereas those papers from an academic setting tended to be focused on one topic in much greater detail.

Twenty-eight total papers were used to address this question and related subquestions. Of them, three contained knowledge deficiencies from the perspective of industry managers, fifteen contained knowledge deficiencies from the perspectives of software professionals in industry, four contained knowledge deficiencies from the perspective of hiring personnel at companies in the software industry, and seven contained knowledge deficiencies from the perspective of recently graduated students or students who were in the final year of their program. The papers that contained the knowledge deficiencies identified by each group is shown in Table 9 and discussed in the following subsections.

4.2.1. Question 2.1: How Do the Methods for Determining and Evaluating Knowledge Deficiencies Differ for Each?

Papers which focus on experienced industry professionals or the expectations of managers and hiring personnel were all based on the results of surveys and interviews.

Table 9. Knowledge Deficienices and Distribution Across Perspectives

Deficiency	Manger/Hiring Personnel	New Professionals	Old Professionals	Students
Testing	[6]	[36]	[11, 22, 12, 23, 24]	[41, 7]
Programming	[27, 9]	[15]	[12, 2, 39, 37, 10]	
Teamwork	[6, 27, 9]	[3, 4, 36]	[16]	
Oral Communication	[9, 14]	[3, 4]	[12, 2]	
Configuration Management	[43]	[3, 4]	[22]	
Design		[36]		[25, 8, 41]
Problem Solving	[43]	[36]	[2, 13]	[41]
Project Management	[14]		[22, 23, 24]	
Requirements		[36]	[22, 37, 23]	
User Interface Design	[27]		[22, 23, 24]	
Written Comm.	[6, 9, 14]		[22]	
Data Structures	[9]		[16]	[30]
Software Debugging		[3, 4]	[2]	
Software Lifecycle	[6, 27, 43]			
Networking	[43]		[37, 10]	
Presentation Skills	[6]		[23, 24]	
Software Quality Assurance	[6]		[23, 24]	
Tool Developments	[26]	[36]		
Algorithms	[9]			[41]
Documentation	[6]		[37]	
Ethics			[22, 23]	
Programming Language	[26]		[16]	
Leadership Ability	[6, 26]			
Software Maintenance			[22, 2]	
Object Oriented Concepts			[23, 24]	
Experience (Internships)	[9]			[21]
Business Procedures	[14]			
CS Theory			[39]	
Management Skills			[22]	
Multi-threaded Programming	[26]			
Software Development Processes			[37]	
Security	[43]			

However, when distinguishing between recently hired professionals and experienced professionals, it was common for researchers to use case studies where researchers would follow new employees throughout their work day to evaluate knowledge deficiencies found in recently hired professionals. Three of the five papers that focus on newly hired professionals used case studies, whereas the other two papers used surveys and a combination of surveys and interviews.

The most likely explanation of these results is that the time of managers and experienced professionals is valuable, making it difficult for these individuals to commit large amounts of time to external studies. Surveys also make it possible for researchers to gather data from a large number of subjects, helping improve the generalizability of the results. Papers which report the results of knowledge deficiencies in students predominantly use controlled experiments to support the results of the paper. Five of the seven papers which focus on knowledge deficiencies in students used some form of experiment to measure student knowledge or performance, whereas the other two papers used surveys.

4.2.2. Question 2.2: How Are the Deficiencies Identified by Each Group Similar or Different?

Both academia and industry have identified several common knowledge deficiencies including knowledge of data structures, the ability to solve problems, and software testing. However, there are also significant differences in the types of deficiencies that are identified by academia and industry.

The most common difference is the industry identification of knowledge deficiencies for a large number of non-technical abilities, commonly referred to as soft skills. Both oral [12, 2, 3, 4, 9, 14] and written [6, 22, 9, 36] communication abilities have been identified in a large number of industry focused research, but have not been identified in any academic papers. Leadership ability [6, 26], presentation skills

[6, 23], and the ability to work as part of a team [27, 16, 36] have also been identified as areas in which graduating students are deficient.

Another knowledge deficiency that has been commonly identified by industry, but not at all by academia is programming ability. Some papers have identified specific types of programming such as multi-threaded programming [26] or procedural programming [39], but most mention general ability [12, 2, 39, 10, 9, 15]. Being language independent was mentioned in one paper [27], but others also mentioned ability in specific languages such as Lua and C++ [26, 16].

Another major difference is that academia has identified that students lack the ability to design software [Loftus11, Eckerdal06, Wang08], however industry managers and hiring personnel have not identified this as a major knowledge deficiency among graduating students. This suggests that industry expectations may be low in regards to the ability of recently hired graduates to design software or that the roles of these new employees do not commonly involve designing software.

4.2.3. Question 2.3: Are There Other Differences Between Identified Deficiencies within Academia or Industry?

When more carefully examining the research papers containing an industry perspective, several additional differences in identified knowledge deficiencies can be observed. These differences exist between the various positions in a company, such as newly hired developers, experienced developers, and managers or other hiring personnel. Differences can also be observed based on whether or not the company focuses more on software engineering or identifies itself as an information technology company.

Recently hired software developers more frequently reported debugging as a knowledge deficiency than either more experienced professionals or managers [2, 3]. A possible explanation of this phenomenon is that at some companies, newly hired graduates are primarily given tasks that involve debugging code and making bug reports [4]. Managers, however, were the only group of professionals to report deficiencies in the understanding of the software lifecycle [6, 27, 43]. The most obvious explanation for this is that individual developers are not likely involved in more than small portion of a software product's lifecycle or development process.

Differences also exist depending on whether or not the companies describe themselves as IT companies or more traditional software development companies. Companies in the IT field more frequently reported knowledge deficiencies in both oral communication [12, 2, 9, 14] and that graduating students lacked necessary real world experience [9, 21]. In addition to this, IT companies also accounted for a strong majority of the cases where general programming ability was identified as a knowledge deficiency [27, 12, 2, 43, 9]. Alternatively, companies involved in software development were more likely to identify software testing as a major knowledge deficiency [6, 11, 22, 41, 23].

Some key differences also exist between the general software development industry and the game development industry. The game development industry placed a strong emphasis on knowledge deficiencies in specific languages such as C++ [16] and software tool development [26] which were not indicated in other papers from an industry perspective.

4.3. Question 3: Can a Classification System for the Identified Knowledge Deficiencies be Created?

A majority of the knowledge deficiencies identified in this literature review can be categorized with a relatively simple classification system. This system is largely based off of an amalgamation of several classification systems used in previous papers (covered in additional detail in the following subsection) examining knowledge deficiencies in graduating students.

Table 10. Knowledge Deficiency Taxonomy

Soft Skills	Software Engineering Practices	Computer Science Concepts	Software Tools
Oral Communication	Design	Theoretical CS	Debuggers
Written Communication	Testing	Data Structures	Configuration Management
Leadership	Requirements	Programming	Development Tools
Presentation	Software Lifecycle	Networking	Misc. Software Tools
Teamwork	Software Development Processes	Security	
Ethics	Maintenance	Object Orientation	
	Project Management	User Interface Design	

The majority of knowledge deficiencies can be placed into one of four categories: *Soft Skills*, which cover people skills; *Software Engineering Practices*, which cover knowledge areas related to software engineering and activities performed during software development; *Computer Science Concepts*, which cover core computer science topics; and *Software Tools*, which cover the development and usage of software tools designed to support software development activities. Table 10 shows this categorization and how identified knowledge deficiencies fit within these categories.

4.3.1. Question 3.1: Have Classification Systems Been Created or Examined in Previous Research?

The vast majority of the selected papers did not contain any explicit taxonomy for knowledge deficiencies, in some cases because the paper only examined one specific knowledge deficiency in detail. In the cases where a classification system was used, it was predominantly a system that classified deficiencies into two or three high-level general categories, a technical skills category; a non-technical or soft skills category;

and another category such as business concepts [Byrne97, Bailey01, Woratschek02, Haddad02]. Other papers used more robust classification systems, most of which provided further sub-classification of technical skills. One paper categorized knowledge deficiencies into four categories: mathematics, software, other engineering, other topics [Lethbridge98]. Another provided yet further classification of these topics into additional categories such as theoretical computer science concepts, mathematical topics widely used in computer science, other mathematics, software engineering processes, software design core, software subsystem design, and other software [Lethbridge00]. Other papers used categories such as software tools [McGill09], managing customers [Goles08], and time management [Haywood00] to categorize knowledge deficiencies.

4.4. Discussion of Findings

This section provides a summary of the principal findings of the systematic literature review, discusses the strengths and weaknesses of the gathered evidence, relates how this information can be used by both educators and industry professionals, and presents future work that will evaluate and extend this research.

4.4.1. Principal Findings

The purpose of this literature review was to identify knowledge deficiencies found in graduating computer science and software engineering students. To ascertain this information, a systematic literature review was designed and executed on popular publication databases related to computer science and software engineering education, as well as a wide variety of journals and conference proceedings related to the topic area. The identified deficiencies were analyzed and categorized to explore trends in the data. The principal findings of this review are as follows:

- Graduating computer science and software engineering students are commonly found to be knowledge deficient in a large number of different areas, ranging from technical knowledge and computer science to concepts to soft skills such as communication ability. A list and description of the most common knowledge deficiencies can be found in Section 4 and a brief description of all deficiencies can be found in Appendix B.

- Academia and industry have different methods of identifying knowledge deficiencies and also identify different types of knowledge deficiencies. This suggests that there is some disconnect between the educational goals of college universities and the actual needs of software development companies in industry. A list of the different deficiencies identified by these groups can be found in Table 9 and a description of the differing methods used to identify knowledge deficiencies can be found in Section 4.

4.5. Strengths and Weaknesses

This section discusses the strengths and weaknesses of the evidence collected in the literature review by examining the source selection, source quality, and validity of the evidence.

4.5.1. Source Selection

A wide variety of sources were selected from in order to produce a large list of candidate papers for inclusion in the literature review. Multiple literature databases covering relevant journals, proceedings, and other publications were searched and a manual search of conference proceedings and journals associated with the topic area was also conducted. Unfortunately, there were not a large number of empirically validated sources restricted to the precise topic area of this report. In order to increase the number of available sources and evidence of knowledge deficiencies, some constraints of the inclusion/exclusion criteria were relaxed. Specifically, sources from the information technology and information systems fields were considered, especially if they made some indication that the research was focused on knowledge deficiencies for students who would be employed in programming positions or other software development roles. Furthermore, it is likely that there are additional sources related to information technology and information systems that were published in journals related to those fields that were not found from the database searches. However, including those journals in the manual search would have resulted in an excessive amount of additional work for a minimal amount of return.

4.5.2. Source Quality

To ensure that only quality evidence was included for consideration in this literature review, numerous quality metrics were identified and a quality assessment system (described in Section 3.6) was developed to evaluate all sources. In general all sources can be considered to have a quality, at least insofar as the data in the sources is based on an empirical study.

Due to the low number of sources immediately related to the topic area, it was necessary to relax constraints to improve the overall number of sources. In order to ensure that selected papers about information systems were still relevant to the review, effort was taken to ensure that these papers focused on roles (e.g. programming) that computer science and software engineering students could expect to fill in their future careers. While this does lower the quality of the overall results, it was believed to be better to sacrifice some quality to ensure that a sufficient number of results were available for analysis.

4.5.3. Validity of Evidence

Stringent inclusion and exclusion criteria were used to ensure that only appropriate papers were included in this literature review. To reduce potential bias, data extraction forms were used to ensure that the same information was extracted from each source. After extracting relevant information from all sources, the researchers of this work compared a subset of selected papers to ensure that the data being extracted was similar and consistent.

Additionally, the presence of certain knowledge deficiencies in a large number of independent sources provides further strength to the validity of evidence.

CHAPTER 5. FURTHER STUDIES OF KNOWLEDGE DEFICIENCIES

In order to validate the data collected during the systematic literature review, to expand the qualitative information on the identified knowledge deficiencies, and to gather results that were more up to date, additional research was conducted. To determine which research methods would produce the most valuable results while still being practical to implement, various potential options were evaluated to determine which would be feasible. Table 11 contains a list of potential subjects, research methods, costs, and result values that were evaluated.

Software managers and hiring personnel were determined to the best source of information as they are directly responsible for interviewing and hiring recently graduated computer science students. Two possible methods for collecting data were considered: surveys and interviews. Surveys would have allowed for a larger number of potential responses, but the results would have been largely limited to a predefined list. It would also be difficult for respondents to elaborate on information or to provide additional details about identified deficiencies. Interviewing managers and hiring personnel would be more time consuming from both the perspective of the researchers and interviewed subjects, however the results would be more detailed and there was also the potential of uncovering additional knowledge deficiencies that had not been identified in the literature review. In both cases, responses may be limited as software professionals' time is valuable and it may be difficult to convince subjects to participate.

Software developers also presented another valuable perspective. Similar to managers and hiring personnel, the time of these individuals is limited and valuable, so finding a large number of subjects willing to participate can be challenging. Furthermore, it can be difficult to find software developers who had graduated from

Table 11. Potential Research Methods, Costs, and Result Values

Subject	Method	Costs and Issues	Result Value
Managers and hiring personnel	Surveys	Compiling a survey. Managers' time is valuable. Results mostly limited to predefined selections.	Most valuable source of information. Potential to target largest number of managers.
Managers and hiring personnel	Interviews	Time consuming. Managers' time is valuable. May not cover all deficiencies.	Most valuable source of information. Potential to get new information and identify new knowledge deficiencies.
Software developers	Case study	Time consuming. Developers' time is valuable. Getting access to developers. Limited number of results.	Results based on real-world occurrences. Results not biased by developers' opinions.
Software developers	Surveys	Compiling a survey. Developers' time is valuable. Finding appropriate developers to survey.	Potential for large number of results. Large variety of developers and education experiences.
Students	Testing	Time consuming. Finding testing methods for many knowledge deficiencies. Finding a large sample size. Testing at multiple universities not feasible. Does not provide information about unknown deficiencies.	Best measure of deficiencies in students.
Students	Surveys	Compiling a survey. Results mostly limited to predefined selections. Limited response rate. Students may be biased or not accurate portray their skill levels.	Surveying students from multiple universities is feasible. Can cover existing most or all existing deficiency categories.

college within the past two or three years. These developers would be ideal research subjects as they would have recent, and fresh knowledge of areas in which they had struggled during their jobs, as well as how their college education had prepared them for work in industry. A case study of developers in their actual work environment was identified as the best method for studying knowledge deficiencies in software developers. This would allow researchers to observe developers and identify areas where developers actually struggled as opposed to relying on developer perception of their own knowledge deficienices. However, conducting such a study would be immensely time consuming and require having a large amount of access to several suitable subjects, making it impractical. Surveying developers was also a possibility, but it would be necessary to survey developers at a large number of companies in order to find a substantial number of developers who had graduated with in the past two or three years and to ensure that a wide enough variety of companies was included to have results which could be generalized. The poor response rates from similar studies identified in the literature review made this approach unappealing [22].

Students present a third potential source for identifying knowledge deficiencies. Students were deemed to be a less valuable source of information compared to managers and developers as they may not have a good understanding of industry expectations or the ability to accurately rate their abilities based on industry expectations. However, students are available in the largest numbers of any of these groups and are generally more reachable. The best method to evaluate knowledge deficiencies in students would be to test their knowledge in the areas of previously identified deficiencies. This is impractical as in many cases there are not good tests for evaluating these abilities (e.g. teamwork) and the amount of time it would take to cover the even the most prevalent deficiencies would be too large. Surveys were another potential method for gather data from students, but presented several disadvantages. First, the results would be limited to predefined categories, but more importantly, it would be difficult to determine if a student accurately represented his skills or knowledge.

The different approaches were examined to determine which would provide the best qualitative information about knowledge deficiencies and be relatively easy to implement. After analyzing the different approaches, the best methods were identified as interviewing managers and hiring personnel along with surveying students. Interviews were chosen as they provided the ability to get instantaneous feedback about identified deficiencies as well as the potential to uncover additional deficiencies, and because it was felt that the response rate for interview requests would be better than the response rate to fill out a survey. Software developers were not chosen because it was considered too difficult to get a large enough sample size from a wide enough variety of companies to produce meaningful results, even though the information would be valuable. Additionally, a majority of the results from the systematic literature review were from the perspective of developers, so there was less of a need from additional results from this groups perspective. Surveys of graduating senior-level students were also chosen as it would be easy to develop a survey and ask students to participate. Even if no new knowledge deficiencies were identified, the results of the survey could serve as additional support for knowledge deficiencies identified in the literature review and through interviews with the managers and hiring personnel. Additionally, none of the previous research identified in the literature review had attempted this method so there existed the possibility of gaining useful information that would otherwise be difficult to identify.

5.1. Interviews with Managers and Hiring Personnel

The following subsections provide details of a study where multiple industry managers and hiring personnel were interviewed about knowledge deficiencies in recently graduated students. First the high level study goals and research questions are described, following by a description of the study design. Next the study participants and data collection process are detailed. Then the results of the study are given and a brief discussion of the results follows. Finally, threats to validity are addressed.

5.1.1. Study Goals and Research Questions

The main goal of this study was to provide a support for knowledge deficiencies identified in the literature review and to gain additional qualitative data about those deficienices. Several of the studies from previous research into knowledge deficiencies had been conducted a decade or more prior to this research and there was reason to believe that some of the results may have been outdated. Therefore, this study would provide a more updated set of knowledge deficiencies experienced by recently graduated students working in industry. It was also important to differentiate between deficiencies that were encountered after recently graduated students had begun their new jobs and those deficiencies that generally prevented them from getting jobs in the first place, as previous research had not attempted to distinguish between the two. To that end, the following research questions were developed:

1. Are there differences in the knowledge deficiencies identified during manager interviews with recently graduated students and those identified after those students begin working?

2. What knowledge deficiencies prevent recently graduated students from being hired for jobs?

5.1.2. Study Design and Process

To receive quality information, a semi-structured, open-ended interview with managers and hiring personnel was used. To better understand how knowledge deficiencies both prevented recently graduated students from receiving a job and how those deficiencies affected them in their jobs, interview participants were asked two main questions: 1) knowledge deficiencies that they screened for during the interview and job application process that would prevent a recently graduated student from receiving a job; and 2) areas in which recently graduated students struggled and did not meet expectations after beginning their jobs. Participants were also asked to provide some basic information about their interview process in order to establish additional context for how knowledge deficiencies were identified during their interviews with potential candidates as well as how some deficiencies would slip through the screening process. In some cases, the managers and hiring personnel worked at companies located in or near Fargo, North Dakota. To ensure that the results were generalizable, participants were reminded that in identifying deficiencies, it was not necessary to limit their responses to only deficiencies observed in applicants or employees who had graduated from NDSU.

The interviews were open-ended in that participants were encouraged to speak freely with minimal guidance from the interviewer. In general, the interviewer attempted to avoid direct questions about any particular knowledge deficiencies, but would ask about broad categories such as software tools, soft skills, etc. if the participant had not mentioned any knowledge deficiencies from that category. If asked to elaborate on a particular category or provide examples, the interviewer attempted to give additional details without directly specifying any one previously identified deficiency. If a participant identified a particular deficiency that could be somewhat ambiguous, such as software testing, the interviewer would ask for additional information about the deficiency (e.g. was it a particular testing method, does it have more to do with tool usage, is it a basic lack of knowledge about testing, etc.) to gain more insights into some of these categories.

The interview was designed to take approximately twenty to thirty minutes as to minimize the amount of time commitment necessary from participants. In most cases, only one individual from each company was interviewed, but in a few cases multiple people were present at the interview. This occurred if the company had multiple large divisions within the company that undertook different roles or in one case where human resources handled the majority of the candidate screening and interview process.

5.1.3. Participants

Participants in this study were fourteen managers or hiring personnel at various companies located predominantly in the Midwest United States. The companies ranged in size from smaller organizations that employ under one hundred people to large organizations that employ over one thousand people. All of the chosen companies had previously served as a sponsor company for the senior capstone course of the computer science department or had worked with NDSU in some other capacity.

Twelve of the fourteen participants worked as managers or team leads who were responsible for both overseeing other employees and interviewing new candidates for positions at their company. One of the participants was a manager who did not participate in the interview process at their company. One of the participants worked in the company's human resources department. Although this person was primarily responsible for interviewing candidates at the company, they were still able to provide some feedback about issues that newly hired students experienced based on performance reviews and feedback from other managers at the company.

5.1.4. Data Collection

Study participants were interviewed by the primary researcher. The majority of interviews were conducted over the phone or Skype, but in a small number of cases where the participants were located in Fargo, the interview was conducted in person at one of the company's facilities.

During the interview, the interviewer took notes on the different knowledge deficiencies identified by the study participants. If multiple participants were present during the interview, the researcher took care to distinguish one participant's statements from any other participant's statements. Qualitative information about knowledge deficiencies was also recorded.

After the interview had ended, the notes were examined to map identified knowledge deficiencies to existing categories identified in the literature review. In the majority of cases, the deficiencies identified by the study participants fit into an existing category. In the event that no suitable existing category could be found, a new separate category was created to describe the knowledge deficiency.

5.1.5. Results

Thirty-six separate knowledge deficiencies were identified during the interview process. The most commonly identified deficiencies are listed in table 12. The results from the table are categorized by whether the knowledge deficiency was commonly identified in recently graduated students who were interviewing for a position or if it was only encountered after a recently graduated student began employment at the company. In some cases, a participant identified a knowledge deficiency as something that they specifically looked for during the interview process as well as something that was still an issue for their new employees. In this case, the deficiency is listed in both categories, but is only counted once for the total number of times it was identified.

Oral communication and project experience were the two most frequently identified knowledge deficiencies, with both being mainly identified during the interview process. When asked to provide additional details, participants usually mentioned that recently graduated students had difficulties effectively communicating their ideas or seemed afraid of asking questions. One participant also stated that students interviewing at the company often used too much jargon when attempting to describe

Table 12. Knowledge Deficiencies Identified in Interviews with Managers and Hiring Personnel

Deficiency	Identified in Interview Process	Encountered during Employment	Total
Oral communication	8	2	9
Project experience	9	0	9
Configuration management tools	0	8	8
Problem solving	6	2	8
Ability to see the big picture	4	2	6
Commenting and documenting code	0	5	5
Software development processes	0	5	5
Teamwork and collaborative skills	2	4	6
Working with and understanding customer needs	0	5	5
Databases	1	3	4
Testing	2	4	5
Written communication	1	5	5
Ability to self-manage	1	4	4
Software design	2	2	4
Ability to learn new skills	2	3	4
Knowing when to ask for help	0	4	4
Understanding job expectations	0	5	5
Software maintenance	0	3	3
Understanding business aspects	1	2	3

their work. The majority of participants also listed project experience as another major factor of their interview process that could disqualify a candidate. Many candidates described this deficiency as some previous experience working as part of a team on a large project, whether this was through previous work experience, an internship or co-op, or a capstone project done at the student's university. One participant stated that their company liked to see an example of a large project that a student had worked on individually.

Ten study participants indicated a lack of knowledge or experience in using at least one type of software tool as a knowledge deficiency. The most commonly identified tool category was configuration management or version control software. Multiple different version control programs were listed, but the most frequently identified were SVN and git, but many participants indicated that knowing a particular version control program was not as important as having experience with using the tools. Two interviewees specifically mentioned that recently graduated students were unfamiliar with the concept of branching and merging code and predominantly struggled in this area, while one interviewee indicated that students did not make good commit comments or understand the importance of doing so. Other software tools that were less frequently indicated included debuggers, bug tracking tools, code and run-time analysis tools, IDEs (integrated development environments), database tools such as DBMS (database management systems), and testing software.

Problem solving ability and critical thinking skills were also identified by a majority of interviewees. Multiple participants stated that some recently graduated students appeared to lack even basic problem solving abilities and would be unable to solve basic problems (e.g. simple tree traversal) given to them during an interview. One interviewee indicated that many students who exhibited this issue lacked understanding of the problem solving approach and did not appear as though they even knew how to begin to tackle the problem or where to start. Although most responses indicated that deficiencies in problem solving were usually identified in the interview process, two participants reported that even the recently graduated students that they hired still experienced issues with problem solving. One indicated that it was just a general problem with developing algorithms to solve a problem, whereas another indicated that although recently graduated students could usually solve a given problem, their solutions were generally not good and in many cases created other problems.

Another knowledge deficiency that was commonly sited was the inability for recently graduate students to see the “big picture” of a project and to understand how they fit in to that project. When asked to provide additional details, one interviewee responded that recently graduated students had a tendency to not consider how their software designs would impact other parts of the project such as testing. Another described the issue as recently graduated students not understanding how other developers or teams will need to interface with their code. Some of the participants indicated that this was closely related to a lack of experience working on a large team project where much of the work was not done by any individual person.

Commenting or documenting code was indicated as a knowledge deficiency by five participants. One interviewee stated that recently graduated students tended to produce fairly useless comments that were largely unnecessary or produced comments for code that needed additional explanation. One interviewee also indicated that the problem was not limited to code, but that new hires were generally not good at documenting the work that they were doing either. Another interviewee stated that in addition to this, recently graduated students also had problems following coding standards.

Another soft skill that was identified by multiple interviewees was written communication. One interviewee stated that recently graduated students often struggled with producing quality documentation whereas another indicated that a growing trend among new hires was poor grammar in memos and other written communications. Another participant stated that evaluating candidates technical writing ability was becoming a part of the interview process. One interviewee also indicated that recently graduated students had difficulties expressing their thoughts in written form and that memos and proposals from these individuals were often difficult to read and comprehend. Another interviewee indicated that new hires often did not have good technical writing skills, but that they were unnecessary as the company had a department staffed by technical writers.

Another deficiency that was identified by six participants was collaborative skills and teamwork. Two interviewees indicated that their company generally tried to find people who exhibited good team work skills as part of the interview process. One interviewee indicated that this was done by looking at previous work they had done as part of a group and asking about that individual's role with in the group as well as by determining if the individual had a personality that would fit in well with the company. One interviewee indicated that recently graduated students don't have enough experience as working as part of a team and often have issues communicating effectively with other team members. Another interviewee indicated that often their new hires would be too absorbed in their own portion of the work to the detriment of the team.

Software testing and software design were knowledge deficiencies that were identified as both being important in order to make it through the interview process as well as an area in which newly hired students often struggled. One participant indicated that part of the interview process was producing simple unit tests for code that they had just written. Two other interviewees indicated that their new hires were generally poor at writing unit tests for code that they had developed and one indicated that the new hires often lacked experience with unit testing frameworks and other testing tools. Another interviewee stated that recently graduated students had problems developing tests from an end-users perspective and developing meaningful test cases. One interviewee also mentioned a lack of knowledge of or experience with test-driven development. Two participants indicated that part of their companies interview process was to assign candidates hypothetical design problems in order to gauge their ability to design software and to evaluate how they approached the design process. One interviewee expressed the tendency for recently graduated students to want to dive in to coding without taking time to consider the design of the software and another interviewee indicated that their new hires generally did not have a good knowledge of design patterns. Another indicated that new hires were generally not familiar with regression testing or regression testing tools.

Several personal skills that may be closely related were also identified. The most frequent were the ability to learn new skills and tools on the job, and the ability to self-manage. Interviewees stated that some recently graduated students who had begun their new jobs did not do an adequate job of self-learning or managing their work. One interviewee indicated that newly hired students were not proactive and thought that this was most likely due to the nature of education vs. work where students were used to waiting for new assignments before doing additional work. Interviewees also indicated that recently graduated students had some issues with learning new tools or skills that were a necessary part of their job. One interviewee specifically mentioned adapting to new programming languages and said that while many students could eventually become proficient in the language that it took them longer than expected to do so. Another interviewee described that as part of their company's interview process it was important for them to see how students could apply what they've learned and their existing experiences to new problems.

Another set of related deficiencies identified were that recently graduated students did not always have a good understanding of job expectations and did not know when they should ask for help. One interviewee indicated that many of their new hires didn't always have a reasonable understanding of what was expected of them and may try to do too much work initially and had difficulties knowing when they should ask for help. Another interviewee indicated that new hires often had difficulties determining when they were stuck and needed help and attempting to solve the problem by themselves. Another interviewee indicated that because new hires didn't wish to appear unskilled or unknowledgeable to their boss, they would often avoid asking questions and getting help in situations where they clearly needed it.

Databases were also mentioned by four different participants. One interviewee stated that database knowledge and skills were an important part of their interview process and that candidates needed to have a good understanding of database design, whereas most other interviewees indicated that although database knowledge as important, they did not screen out applicants if they had a poor understanding of database concepts. One interviewee stated that recently graduated students also lacked experience with database management systems and other tools for interacting with databases.

5.1.6. Discussion of Results

The results from this study helped to provide additional insight into several categories of knowledge deficiencies as well as to expose additional knowledge deficiencies that were not present or prevalent in the literature review. The study also helped to distinguish how managers and hiring personnel attempt to screen for some important knowledge deficiencies during their interview processes. The remainder of this section addresses the research questions found in section 5.1.1.

Are there differences in the knowledge deficiencies identified during interviews with recently graduated students and those identified after students begin working?

The results from table 12 indicate that there are several knowledge deficiency categories with large differences in where they are most commonly identified by managers and hiring personnel. Deficiencies in oral communication are most commonly experienced and identified during the interview process. This is somewhat obvious as most of the interviewed companies have either lengthy or multiple interviews that are driven by verbal communication between the interviewer and candidate. Of the two study participants that identified issues with oral communication during the course of a newly hired recently graduated students job, one indicated that the issue primarily centered around the new hires staying in communication with their boss and proving regular updates about their work. The other participant was not responsible for interviewing candidates. Other issues related to communication may also occur once a recently graduate student begins employment. Difficulties in knowing when to ask for help and communication deficiencies that adversely affect the collaborative ability of recently graduated students were two frequently given deficiencies exhibited by those students. Similarly, a large number of study participants indicated that they tended to identify deficiencies in problem solving ability during the interview process. Of those who identified such deficiencies in newly hired recent graduates, one indicated that those new hires did not carefully approach problems and often created additional problems through their solutions.

Having experience working with a large team project was one deficiency that was identified entirely in the interview process. A related deficiency, the ability to the big picture in a project and for an individual to understand their role within that project, was also identified frequently during the interview process. One of the interviewees who stated that this was a problem for newly hired recent graduates in their jobs, indicated that those new hires who did not have a good understanding of how other parts of the project worked often made poor design decisions in their own parts of the project. This seems to indicate why such a large number of companies prefer candidates to have internship experience or to have completed a large team project as part of a capstone course.

On the other hand, deficiencies related to software tools were only identified after recently graduated students had begun their jobs. The most likely explanation for this is that most of the interview processes do not involve working with specific tools that the company uses and creating specialized environments to test the abilities of candidates would be infeasible. Similarly, deficiencies such as ineffective code commenting, inability to self-manage, and understanding customer needs are deficiencies which are not easy to test for in an interview setting.

What knowledge deficiencies prevent recently graduated students from being hired for jobs?

The two most prominent deficiencies that may prevent recently graduated students from obtaining jobs are poor oral communication skills and a lack of experience with large team projects. Of the companies interviewed, only one indicated that an internship or co-op was absolutely necessary for employment, as most companies were merely interested in seeing some project experience, whether it was a large personal project or a team project done as part of a capstone course. Personality was also tied in closely with oral communication skills and some study participants specifically mentioned an outgoing personality was something that they looked for in their interviews in addition to being able to effectively communicate.

Problem solving was another frequently given response related to deficiencies that could prevent an applicant from being hired. Multiple participants indicated that presenting applicants with simple problems (e.g. breadth-first tree traversal or determining if a year is a leap year) was a common part of the interview process. Applicants who had difficulties solving these problems or effectively showing that they could at least attempt to solve the problem were generally not hired or asked back for additional interviews.

Other deficiencies were also given, but were nowhere near as prevalent. These include understanding objected oriented concepts (e.g. inheritance and polymorphism), being able to produce a design for a small software project and evaluate the design choices made against possible alternatives, the ability to write unit tests for small samples of code, and being able to express a lot of passion for the area or job in which they will be working. Two companies also indicated that having a high GPA was important and that applicants with lower GPAs might not even be interviewed; however, several other companies indicated that GPA generally wasn't considered when determining which candidates to interview.

5.1.7. Threats to Validity

Because the companies that were interviewed were predominantly located in the Midwest of the United States, there is the possibility that the results do not generally represent software development companies throughout the country. However, the business areas of the different companies were varied enough to suggest that the results are not too specific to any particular business domain (e.g. banking software) and can not be generalized to software development roles in general.

Because study participants were not asked about specific knowledge deficiencies, there is a possibility that some deficiencies are underrepresented in the results of the study. However, this was considered preferable to specifically asking about individual deficiencies and potentially biasing the results. There is also the possibility that areas in which recently graduated students struggle in their new jobs were not identified if the interviewed manager had expectations that their new hires would struggle in that area and merely expected them to learn this part of their job.

None of the interviews were recorded, which in some instances lead to small ambiguities or vagueness for some deficiencies. In some cases, detailed information was not provided for knowledge deficiencies, either because the interviewee did not provide additional details or because the interviewer was not able to record all of the details of the conversation. In this case, the identified deficiencies are still counted as being identified, but no qualitative information is given.

5.2. Surveys of students

The following subsections provide details of a study where students were surveyed about their educational experiences and asked to evaluate their knowledge and skills in several areas that have been identified as knowledge deficiencies by previous researchers. First the high level study goals and research questions are described, following by a description of the study design. Next the study participants and data collection process are detailed. Then the results of the study are given and a brief discussion of the results follows. Finally, threats to validity are addressed.

5.2.1. Study Goals and Research Questions

The purpose of this study was to gain additional insight into knowledge deficiencies from a student perspective. Because the survey format would preclude gaining information about knowledge deficiencies beyond those for which specific questions existed, researchers felt it would be worthwhile to include open-ended long-response questions for subjects to provide additional details or information. This would provide qualitative information to enable a better understanding of knowledge deficiencies from students' perspectives. An additional goal of the study was to identify parts of their education that students did not feel were particularly useful or valuable to them. The following research questions were developed based on these goals:

1. Is there any support for previously identified knowledge deficiencies based on students' ratings of their own abilities and knowledge?

2. In what ways do students not feel as though their education has provided them with the necessary knowledge and skills for their future careers?

3. Are there parts of students' education that students feel is not worthwhile or useful for their future careers?

5.2.2. Study Design and Process

Based on the results of the literature review, a forty question survey (the survey instrument is available in Appendix C) was designed to cover many of the knowledge deficiencies identified in the literature. Qualitative information from interviews with managers and hiring personnel discussed in section 5.1 was also used to add additional information to some of the questions that covered relatively broad topics when applicable. In addition to questions related to knowledge deficiencies, subjects were also asked about whether or not they had already secured a job upon graduation and what different careers they were interested in pursuing. Subjects were also asked if they felt as though their education had prepared them for their future careers and were given two long response questions asking them to list areas where they felt as though their education had not adequately prepared them for obtaining a job and also areas where they felt as though their education would not be useful in their future careers.

In order to ensure that the survey was well designed and useful, a pilot version of the survey was designed and given to students enrolled in the capstone course in the computer science department at NDSU. A limited subject response (30%) on the pilot version of the survey lead researchers to adjust the design of the study to remove a large number of long response questions from the survey. This was done because a majority of the subjects either skipped the questions or did not respond with meaningful information. Also, a large number of partial responses lead researchers to feel that subjects may have abandoned the survey because they felt as though it was too long.

The majority of survey questions were based on a five-point Likert scale with the median response corresponding to neutral. Some questions, such as those that asked whether or not a student had already accepted a job position were yes/no responses.

Questions related to communication ability were based on a four-point scale where subjects rated their oral communication ability as poor or good and whether or not they had taken courses specifically aimed at improving those abilities.

5.2.3. Participants

Participants in this study were senior-level students enrolled in the computer science program at NDSU. An initial pilot study was given to students in the spring semester of 2011 academic year one week prior to graduation. Ten subjects completed the entire survey, whereas sixteen subjects completed only part of the survey. The revised version of the survey was given to students in the spring semester of the 2012 academic year. Nine subjects completed the entire survey and there were no subjects who only partially completed the survey.

5.2.4. Data Collection

Survey data was collected using LimeSurvey, an open source web survey tool. After collecting the data, Minitab, a statistical analysis software tool, was used to analyze the data.

5.2.5. Results

The response rate to the survey was relatively low in both the initial pilot study (30%) and the second study (25%). The low response rate made it difficult to perform adequate statistical analysis on the survey results to find correlations between the data.

To determine if the results of the student survey provided support for the existence of knowledge deficiencies in any of the included categories, a one-sample t-test was used to analyze the data. Table 13 shows the mean response rate and p-values for a one-sample t-test for both groups of subjects who participated in the survey. Based on the results of this analysis, the 2011 subjects' responses for questions about unit testing (2.3) and test automation tools (1.9) were lower than the midpoint

Table 13. Results of Study Survey

	2011 Survey		2012 Survey	
Topic	Mean	p-value	Mean	p-value
Configuration Management	3.10	.594	3.00	0.500
Debugging Tools	2.80	.084	3.44	0.888
Unit Testing	2.30	.001	2.56	0.173
SW Dev. Process	3.40	.865	3.56	0.911
Test Automation Tools	1.90	.000	2.56	0.173
Software Maintenance			3.11	0.600
Networking	3.50	.952	2.56	0.156
Parallel/MT Programming	3.50	.974	3.56	0.952
UI Design	3.90	.979	3.67	0.879
Data Structures	3.90	.991	3.89	0.995
Databases			3.56	0.893
SW Lifecycle	3.90	.991	3.89	0.982
SW Testing	3.50	.991	2.89	0.391
Requirements Elicication	3.30	.783	3.00	0.500
SW Design	3.80	.995	3.78	0.978
OO Programming	4.20	1.000	4.56	1.000
Language Independence	3.80	.989	4.67	1.000
Problem Solving			4.00	0.991
Teamwork	3.70	.934	3.89	0.990
Presentation Ability			3.56	0.975

of the scale with statistical significant ($p \leq .001, .000$, respectively). The results for the 2012 subjects for these questions was also low, but not statistically significant ($p = .173, .173$, respectively).

Because there were too few respondents from both groups to analyze the results for correlations, it was necessary to combine the two groups before performing the analysis. The primary focus of this analysis was to determine if there was a correlation between any of the knowledge deficiency categories and whether or not a subject already had a job or felt prepared for their future career. There were no strong correlations between any of the knowledge deficiency categories and whether or not a subject already had a job or internship after graduation. There was a strong positive correlation between software maintenance and and whether the subject felt prepared for their future ($r^2 = .577$, $p = .019$). There were also some moderate correlations for configuration management and debugging tools ($r^2 = .23$, $.24$, respectively) with near statistical significance ($p = .083$, $.070$ respectively).

Although the quantitative results were not as worthwhile as hoped, some of the qualitative information provided by the subjects helps to provide support for existing knowledge deficiencies. In addition to this, subject feedback about areas of their education that were not particularly useful may make it easier for educators to determine which courses could be cut or deemphasized when determining how to provide better coverage of the topics and areas where students are commonly identified as being knowledge deficient.

Twelve subjects responded to the long-response question related to areas in which they felt their education was lacking. The most common response was a desire for more programming, with some respondents emphasizing additional experience in languages such as C++. One subject seemed angry that some of his or her peers could not explain what a pointer was because they did not have C/C++ experience and Java had not done a good job of exposing them to that concept.

Four subjects specifically indicated a lack of experience with version control software or experience with other software tools. One subject indicated a general desire for more exposure to development tools. Another subject stated that they had not been exposed to version control until their last semester in college and felt that it should have been introduced earlier.

Four subjects also expressed a desire for an increased number of projects in their courses. One subject stated that a majority of their course projects were too short and not wortwhile. Another respondent indicated more course-work should be hands on and that it would be better if the the junior and senior years were more devoted to project-based work.

Two subjects mentioned testing, with one indicating that he or she had not learned any testing skills at all and felt as thought they did not even have a good basic understanding of testing. One respondent listed web technologies, but did not provide any specifics. Another subject database concepts and data structures. One subject indicated that they felt as though they were underprepared, but learned many of the skills that they needed in the capstone course.

Ten subjects responded to the long-response question related to parts of their education that they did not feel would be useful to them in their future careers. The most common response was computer theory, which was given by six subjects. Two of those respondents indicated that while the course was fun or enjoyable, that they did not feel as though it had any practical value. One suggested that the course should focus on programming a parser and compiler so that students could apply what they learned in the course to useful problems.

Three subjects indicated that they felt as though the ethics course was not very valuable. One suggested that the course should be restructured to provide useful information that was not common sense. Another subject suggested that it should probably be incorporated into a lower-level course such as the second semester programming course.

One respondent felt as though there were too many useless general education courses and that four semesters of science courses were too many. Two subjects indicated that statistics courses were not particularly useful.

5.2.6. Discussion of Results

The results from this study helped to provide additional additional strength for the existence of knowledge deficiencies in students, or at least those at this university. The study also provided additional insight about aspects of their education that subjects did not find worthwhile and in some cases, suggestions for improvement.

The remainder of this section addresses the research questions found in section 5.2.1.

Is there any support for previously identified knowledge deficiencies based on students' ratings of their own abilities and knowledge?

Based on the results of the student survey, there seems to be a good deal of support indicating that students may not be proficient at software testing, especially when it comes to using tools or in specific areas of software testing such as unit testing.

In what ways do students not feel as though their education has provided them with the necessary knowledge and skills for their future careers?

A large number of responses to an open-ended question about where students felt least prepared suggests that students may not be as confident in their programming skills or project experience.

Are there parts of students' education that students feel is not worthwhile or useful for their future careers?

The overwhelming response to the relevant survey question was that students did not feel as though theoretical computer science provided much practical value. There were also several students who did not feel as though the social implications (ethics) course was very valuable either.

5.2.7. Threats to Validity

The subjects who were surveyed as a part of this study were from a single university, making it unlikely that the results can be generalized to all students. Furthermore, the response rate was low enough to suggests that it is possible that those students who responded to the survey do not even accurately represent the student population at NDSU. Initially it was thought that the low response rate was due to the length of the survey and the large number of open-ended response questions. However, even after removing most of these questions, the response rate was still rather low. One other possible explanation is that those students who generally would have rated their abilities and knowledge poorly may have been intimidated by the survey and stopped responding.

The limited number of data points also made statistical analysis more difficult. In several cases, the mean subject response was below the mid-point of the scale, but because there were very few data points, it made it difficult to effectively apply statistical tests. The low number of data points also made it difficult to find correlations in the data. There were instances of strong corelations (i.e. r^2 ¿ .35) but with p-values that did not indicate statistical significance.

Another potential issue is that there's no indication that subjects are able to accurate measure their own knowledge and abilities in a given area. For instance, it is entirely possible for a subject who would consider himself or herself to have an excellent knowledge of some category (e.g. software testing), to actually be considered rather lacking by other standards simply because that subject does not fully understand the category or realize that there is significantly more about it that they need to learn.

CHAPTER 6. DISCUSSION AND APPLICATION OF RESEARCH

This section provides a discussion of the major results from the literature review and study results reported in the previous sections. Further discussion of the application of these results is presented by comparing the knowledge deficiencies identified in this research to the curriculum recommendations of the ACM as well as ABET guidelines.

6.1. Discussion of Major Findings

In order to determine which knowledge deficiencies are most well supported, it is necessary to consider results from all of the different research that has been conducted in this work. Results from the systematic literature review (as reported in section 4) as well as those from the additional research studies (see sections 5.1.5 and 5.2.5) are considered when evaluating the existence of knowledge deficiencies among graduate computer science students. Table 14 contains a list of the different knowledge deficiencies discussed in this section as well as a general assessment of the strength of the support for that deficiency from each

Software testing was one knowledge deficiency that was identified consistently in each area of the research. Nine of the twenty-eight research papers included in the systematic literature review paper listed software testing as a prominent knowledge deficiency among graduating computer science students [6, 22, 23, 24, 36, 11, 12, 41, 7]. Five of the fourteen managers interviewed by the researchers of this work also reported that testing was an issue. NDSU students who took part in a survey also rated their abilities in unit testing and automated testing tools as the lowest two categories on the survey. This corroborates with the qualitative information received from multiple managers during interviews and a 2011 study published by Carver, et

Table 14. Knowledge Deficiencies with Strong Support

Knowledge Deficiency	Source & Strength		
	Prior Work	Manager Interviews	Student Survey
Software Testing	Strong	Moderate	Strong
Oral Communication	Strong	Strong	Weak
Written Communication	Moderate	Moderate	Weak
Programming	Strong	Weak	Moderate
Software Tools	Moderate	Strong	Moderate
Project Experience	Weak	Strong	Moderate
Teamwork	Moderate	Moderate	Weak
Problem Solving	Moderate	Strong	Weak

al. that indicates that students struggle with software testing tools [7]. Garousi, et al. indicate that a lack of knowledge and expertise is baring the adoption of many testing tools and techniques in industry [11].

Oral communication (and to a lesser extent written communication) is another knowledge deficiency that is strongly supported in both the literature review [2, 9, 14, 12, 3, 4] and through the interviews with mangers discussed in section 5.1. However, responses from the student survey did not provide good support to indicate that NDSU students felt as though they had communication problems. Aspects of communication that were identified in both the existing literature and brought up during the interviews with managers were issues communicating with customers [14], knowing when to ask for assistance [4], and difficulties with producing quality documentation [37].

Programming ability was a knowledge deficiency that was identified by many papers in the literature review [15, 2, 10, 9, 27, 12, 39, 37] but was not frequently identified in our interviews with managers. However, programming was the most frequent response to one of the questions on the student survey that asked students which parts of their education were viewed by them as inadequate. Miller, et al. identified one area of programming knowledge deficiencies as the inability to be language independent [27], but the students who participated in the survey rated their ability to apply concepts learned in one programming language to another highly (see Table 13). Responses to the student survey seemed to indicate a desire for more programming exercises in general along with more in-depth exposure to other programming languages.

Software tool usage was a knowledge deficiency category has strong support from all areas of this research. Configuration management tools [22, 43, 3, 4] and debuggers [2, 3, 4] were the most cited software tools in the literature review. Configuration management tools were also identified by eight out of the fourteen managers interviewed as an area that recently graduated students struggled with. Although the results from the student survey did not indicate that students felt as though their ability to use configuration management tools was inadequate, two students did indicate that prior to their capstone course they had had no experience with version control. Students also indicated that they were inexperienced with automated testing tools, an issue that has been identified in previous research [11] and was mentioned by two managers who were interviewed by the researchers of this work.

One knowledge deficiency that was not present to a large degree in the literature review, but was one of the most identified deficiencies by interviewed managers and a common response on the student survey was a lack of project experience. Nine of the fourteen interviewed managers indicated that experience working on a large project was a major factor in screening candidates out during the interview process. One-third of the students who responded to a survey question about what areas of their education they considered lacking indicated that they did not feel as though they had enough experience working on large projects. One student even described a majority of the projects as too short and not worthwhile. Two papers in the literature review identified work experience as important, but were mainly referring to internship or co-op experience [9, 21]. Project management was also identified in previous research [22, 14] but additional information was not provided about exactly how this category was used, but it may be related in some degree to the idea of project experience.

A closely related knowledge deficiency that had a moderate amount of support in both the existing literature as well as the interviews with managers and hiring personnel was teamwork and collaborative skill. Teamwork was the third most frequently identified knowledge deficiency in the literature review [6, 9, 36, 27, 3, 4, 16] and was one of the top six mostly commonly identified deficiencies by interviewed managers. However, both groups of students who participated in the survey ranked their teamwork and collaborative abilities among the highest out of all items on the survey.

One final knowledge deficiency that is well supported is problem solving. Eight out of the fourteen interviewed managers and hiring personnel indicated that a lack of problem solving ability was one of the biggest reasons why potential applicants were not hired or found that newly hired recent graduates had difficulties solving problems. Lack of problem solving skills was also reported in the literature by multiple sources [2, 43, 36, 41].

CHAPTER 7. APPLICATIONS OF RESEARCH

This section provides additional discussion on the applications of this research for various third parties, chiefly educators and managers in industry. The primary intended use for this research is to guide course and curriculum design at universities and to be used in the assessment of those courses and curricula. To that end, existing guidelines are evaluated and recommendations are given for both academia and industry.

7.1. Evaluation of Curricula Guidelines

Many colleges and universities base their curriculum around guidelines published by organizations such as the ACM or rely on accreditation by organizations such as ABET to ensure that their computer science or software engineering program meets some standard of quality. The following subsections examine the recommendation of various curriculum guidelines to evaluate how or why it is possible that certain knowledge deficiencies might exist even when universities follow those various guidelines or are accredited through an organization such as ABET.

7.1.1. Computing Curricula 2001

The ACM and IEEE's *Computing Curricula 2001* for Computer Science is considered to be among the best guidelines for computer science curriculum design. First published in 2001 [31] and given an iterim revision in 2008 [32] it provides recommendations for course design and other suggestions useful for computer science departments. The ACM-IEEE Joint Task Force will be updating their curriculum recommendations in 2013 [35].

The wide-scale existence of evidence of knowledge deficiencies in software testing areas is most likely due to a minimal amount of recommended course hours in testing topics. Computing Curricula 2001 only requires three core hours of software validation

[31, p. 17], most of which is not scheduled to occur until courses in the second year in some of the sample curricula. Without sufficient instruction in testing, students may develop naive or ad hoc testing strategies that they continue to employ throughout their academic career. In order to combat this effect, it may be necessary to expose students to proper testing techniques and tools much earlier in the program in order to instill good testing habits that will be carried across their courses.

Although the guidelines recommend communication skills as a general requirement and emphasizes the importance of both oral presentation skills and technical writing ability as well as indicates the importance of incorporating course work designed to enhance those skills [31, p. 42], it does not mandate any particular course specifically aimed at teaching those skills. It is possible that some universities do not specifically require technical writing courses or even offer them. This leaves the burden of introducing good technical writing skills, presentation abilities, and effective oral communication proficiency to computer science and software engineering instructors who may lack formal training in these areas or may not be inclined to foster the development of these skills in their students. In such cases students do not receive adequate feedback regarding their communication skills and develop bad habits.

Computing Curricula does an excellent job of providing guidelines for general programming ability as well as a describing multiple alternatives to introductory programming such as an objects-first approach or an algorithms-first approach. It also lists several important sub-parts of programming that should be covered in first year courses for this different approaches. A potential explanation for the reason that programming ability was frequently identified as a knowledge deficiency is that the guidelines do place any particular emphasis on ensuring that later courses incorporate programming. If a large number of second and third year courses only require minimal amounts of programming, it is possible for the student's skills to atrophy.

Problem solving is another area that the guidelines cover fairly well. The guidelines also list several pitfalls for problem solving [31, p. 23] such as focusing too heavily on coding in introductory programming courses at the expense of design, analysis, and testing. Much like with programming, it is possible that there is not enough focus on problem solving in upper level courses. If these courses mainly focus on learning additional and specialized information rather than applying that information and solving problems with their domains, it could lead to underdeveloped problem solving skills among students.

Similar to testing, software tools are an area that are not given much time under the Computing Curricula 2001 guidelines. Only three hours are recommended and these are generally suggested for inclusion in the initial programming courses. While it is good to include these in the earliest courses in order to familiarize students with their use and functionality, it is likely that an insufficient amount of time is being devoted to covering many tools that are essential in the software development industry, such as configuration management systems, debugging tools, etc. Additionally, there are several other important topics that must be covered in introductory programming courses and tool usage may be passed up so that lectures can be spent on other topics.

Teamwork and projects are two areas that often overlap in the various sections of Computing Curricula 2001. Projects are cited as an excellent opportunity to improve teamwork skills [31, p. 42] while giving students the opportunity to engage in professional practice and exercise many of the other skills they've developed. The guidelines also recommend a capstone course with a focus on large team projects. Additionally, the guidelines state the value of working in teams does not become evident when working on smaller projects. If courses are merely using small, token projects in place larger ones as recommended by the guidelines, it is possible that students will have underdeveloped collaborative skills which may explain the prevalence of this knowledge deficiency in both the existing literature as well as the interviews with managers and hiring personnel. It may also explain the results of the student survey where one student indicated that course projects were too short and were not worthwhile.

7.1.2. ABET Accreditation

ABET (Accreditation Board for Engineering and Technology) is an organization that provides accreditation for universities in the areas of computing, engineering, and other academic programs. Unlike the ACM-IEEE curriculum recommendations, ABET only provides loose guidelines related to educational objectives, student outcomes, and a few, brief requirements. This makes it difficult to directly compare the presence of knowledge deficiencies with ABET requirements, as in many cases, they are not explicitly stated.

For example, one ABET guideline suggests that students should be able to obtain "an ability to use current techniques, skills, and tools necessary for computing practice" but does not make any specific recommendations as to which tools, techniques, or skills are necessary or even how this will be measured. This makes it possible that even if a university is ABET accredited, its students would still be knowledge deficient in areas such as configuration management if in the process of the accreditation review, configuration management tools were not deemed to be necessary.

Similar problems exist with the other knowledge deficiency categories. Without a more concrete set of guidelines, any further comparison is impossible.

7.2. Recommendations for Dealing with Knowledge Deficiencies

The following sections provide brief recommendations for dealing with knowledge deficiencies for both academia and industry. This advice has not been empirically validated or shown to work and should be considered as a set of general guidelines rather than specific suggestions.

7.2.1. Recommendations for Academia

Dealing with existing knowledge deficiencies can prove to be challenging. Often, there is very little room in the curricula for additional required courses and attempting to fit additional material or objectives into existing courses can prove similarly difficult. To best address the most predominant knowledge deficiencies it is necessary to examine their reasons for existence.

Issues with student programming and problem solving can arise if students do not receive regular and challenging course work that requires the use of programming and critical thinking to solve. Ensuring that courses during the sophomore and junior years contains these type of assignments should help to produce seniors who are more confident in their programming abilities and posses better problem solving abilities.

Similarly, ensuring that students have early exposure to proper testing tools and techniques will enable them to build good testing habits. Further exposure to additional testing concepts and tools along with a continued emphasis on testing will also help to improve students' testing ability. Introductory programming courses can begin requiring unit tests to accompany each assignment, and more advanced courses can introduce ideas such as regression testing for use on a large project.

Adequate and good project experience can be achieved by ensuring that course projects are appropriately large and suited for groups. More frequent deliverables can be used to ensure that teams are working on the project throughout the course, rather than attempting to complete the entire project shortly before the deadline. Domain related software tools can be worked in to the projects to expose students to their use, and other tools such as version control software can be used to more easily facilitate collaborative programming and ensure that all team members are participating. Appropriate teamwork skills should develop as a consequence.

Communication skills are something that can be developed throughout the curriculum, but as with many things, it may be necessary to have a course targeted at technical writing and presentation abilities. This allows students to formally learn appropriate writing techniques and gain opportunities to improve their oral communication skills before putting them to use in later courses. Adoption of department wide writing standards may also help to provide some consistency across courses.

Other knowledge deficiencies can be dealt with in a similar manner. By identifying the likely cause of a given deficiency, it becomes possible to choose a general course of action to remove that knowledge deficiency. Furthermore, developing a set of criteria for measuring the prevalence of a knowledge deficiency provides the ability to determine if proposed solutions are having the desired effect. In general, most knowledge deficiencies can be addressed by including early and dedicated instruction designed to prevent the knowledge deficiency from emerging, or by ensuring that activities designed to build student skills are present throughout the curriculum and that courses are appropriately implementing these activities rather than simply going through the motions of doing them.

Other potential uses of this research to academia is to use the results to influence future curricula design. For example, testing appears to be a widespread and often reported issue. This suggests that it may be necessary to include more testing in future curricula recommendations. Some care should be taken, as the results of the student survey may not be generalizable to other universities. For example, these include teamwork and programming, which may be issues unique to NDSU. However, issues such as testing and software tools may still be valid since ACM's curricula recomendations only include a minimal amount of core hours in these areas.

7.2.2. Recommendations for Industry

Industry already does a good job at identifying knowledge deficiencies in order to ensure that their employees are both productive and valuable parts of their companies. However, based on the results from the interviews conducted with industry managers, it is clear that there are some knowledge deficiencies that cannot be easily tested for in advance.

For examples, such as specific tools, short of asking a candidate to demonstrate the use of those tools, simply asking some general question about their basic understanding of that or similar tools may be helpful in determining how much additional training the candidate will require. However, many other deficiencies may be difficult to catch with such an approach. Code documentation, for example, is not something that generally needs to be observed in order to determine if a person has good practices in that area.

Otherwise, companies should focus on developing training material for new employees that is designed to help remove knowledge deficiencies or to minimize their impact. Including some time spent pair engineering or shadowing experienced developers may be another possible solution. By allowing newly hired, recently graduated students to observe experienced developers, it may help them to better understand the processes that these individuals follow. However, care should be taken to ensure that this does not become a crutch for new hires.

Industry can also help to reduce knowledge deficiencies by reaching out to academia and working with educators to identify knowledge deficiencies in students at various universities. This can help to ensure that graduating students are better prepared for their future careers. Industry companies can also collaborate with universities through capstone projects, enabling students to receive experience working on a real project.

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

This section presents a conclusion to the thesis and presents potential future work to be done in this area in order to further the understanding of knowledge deficiencies.

8.1. Conclusions

Based on a systematic review of existing literature, there is ample empirical evidence to support the existence of several knowledge deficiencies among graduating computer science and software engineering students. Further support for these knowledge deficiencies has been provided through further empirical investigations that examined these deficiencies from the point of view of industry managers and hiring personnel as well as from the perspective of students themselves.

Curriculum guidelines were also evaluated to see how the recommendations for courses and course structures can influence the prevalence and magnitude of knowledge deficiencies. Some advice is also given to mitigate or outright eliminate those knowledge deficiencies where possible.

8.2. Future Work

Although this work provides a comprehensive look at knowledge deficiencies and attempts to collect results from a large set of studies, the fields of computer science and software engineering are rapidly changing, necessitating the need for continued research into knowledge deficiencies in order to spot new and emerging deficiencies. Although suggestions for eliminating knowledge deficiencies are given, these methods have not been empirically validated and may be no better than other existing methods of education that have already solved these problems.

The research conducted in this work should also be thought of as a starting point for continued future work. By interviewing additional managers, a better consensus of knowledge deficiencies can be built and analyzed for differences across business focus or geographic regions. Similarly, students at other universities can be surveyed to help determine if certain curricula are better at producing graduates with fewer knowledge deficiencies.

REFERENCES

[1] Richard H. Austing, Bruce H. Barnes, Della T. Bonnette, Gerald L. Engel, and Gordon Stokes, *Curriculum '78: recommendations for the undergraduate program in computer science a report of the acm curriculum committee on computer science*, Commun. ACM **22** (1979), no. 3, 147–166.

[2] Janet L. Bailey and Greg Stefaniak, *Industry perceptions of the knowledge, skills, and abilities needed by computer programmers*, Proceedings of the 2001 ACM SIGCPR conference on Computer personnel research (New York, NY, USA), SIGCPR '01, ACM, 2001, pp. 93–99.

[3] Andrew Begel and Beth Simon, *Novice software developers, all over again*, Proceedings of the Fourth international Workshop on Computing Education Research (New York, NY, USA), ICER '08, ACM, 2008, pp. 3–14.

[4] ———, *Struggles of new college graduates in their first software development job*, Proceedings of the 39th SIGCSE technical symposium on Computer science education (New York, NY, USA), SIGCSE '08, ACM, 2008, pp. 226–230.

[5] Eric Brechner, *Things they would not teach me of in college: what microsoft developers learn later*, Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (New York, NY, USA), OOPSLA '03, ACM, 2003, pp. 134–136.

[6] Declan Jerome Byrne and Jeffrey L. Moore, *A comparison between the recommendations of computing curriculum 1991 and the views of software development managers in ireland*, Comput. Educ. **28** (1997), no. 3, 145–154.

[7] J.C. Carver and N.A. Kraft, *Evaluating the testing ability of senior-level computer science students*, Software Engineering Education and Training (CSEE T), 2011 24th IEEE-CS Conference on, may 2011, pp. 169–178.

[8] Anna Eckerdal, Robert McCartney, Jan Erik Moström, Mark Ratcliffe, and Carol Zander, *Can graduating students design software systems?*, Proceedings of the 37th SIGCSE technical symposium on Computer science education (New York, NY, USA), SIGCSE '06, ACM, 2006, pp. 403–407.

[9] Ernest Ferguson, *Changing qualifications for entry-level application developers*, J. Comput. Sci. Coll. **20** (2005), no. 4, 106–111.

[10] John D. Fernandez and Phyllis Tedford, *Evaluating computing education programs against real world needs*, J. Comput. Sci. Coll. **21** (2006), no. 4, 259–265.

[11] Vahid Garousi and Tan Varma, *A replicated survey of software testing practices in the canadian province of alberta: What has changed from 2004 to 2009?*, J. Syst. Softw. **83** (2010), no. 11, 2251–2262.

[12] Tim Goles, Stephen Hawk, and Kate M. Kaiser, *Information technology workforce skills: The software and it services provider perspective*, Information Systems Frontiers **10** (2008), no. 2, 179–194.

[13] Hisham Haddad, *Post-graduate assessment of cs students: experience and position paper*, J. Comput. Sci. Coll. **18** (2002), no. 2, 189–197.

[14] Dianne Hagan, *Employer satisfaction with ict graduates*, Proceedings of the Sixth Australasian Conference on Computing Education - Volume 30 (Darlinghurst, Australia, Australia), ACE '04, Australian Computer Society, Inc., 2004, pp. 119–123.

[15] Elizabeth Haywood and Jane Madden, *Computer technology students - what skills do they really need?*, Proceedings of the Australasian conference on Computing education (New York, NY, USA), ACSE '00, ACM, 2000, pp. 139–144.

[16] Michael Hewner and Mark Guzdial, *What game developers look for in a new graduate: interviews and surveys at one game company*, Proceedings of the 41st ACM technical symposium on Computer science education (New York, NY, USA), SIGCSE '10, ACM, 2010, pp. 275–279.

[17] Barbara A. Kitchenham, *Procedures for undertaking systematic reviews*, Tech. report, Computer Science Department, Keele University, 2004.

[18] Barbara A. Kitchenham and Stuart Charters, *Guidelines for performing Systematic Literature Reviews in Software Engineering*, Tech. Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[19] Barbara A. Kitchenham, E. Mendes, and G.H. Travassos, *Cross versus within-company cost estimation studies: A systematic review*, Software Engineering, IEEE Transactions on **33** (2007), no. 5, 316–329.

[20] Dean Knudson and Alex Radermacher, *Software engineering and project management in cs projects vs. "real-world" projects: A case study*, Proceedings of the IASTED Internation Conference on Software Engineering and Applications, SEA '09, 2009.

[21] Tony Koppi, Sylvia L. Edwards, Judy Sheard, Fazel Naghdy, and Wayne Brookes, *The case for ict work-integrated learning from graduates in the workplace*, Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103 (Darlinghurst, Australia, Australia), ACE '10, Australian Computer Society, Inc., 2010, pp. 107–116.

[22] Timothy C. Lethbridge, *A survey of the relevance of computer science and software engineering education*, Proceedings of the 11th Conference on Software Engineering Education and Training (Washington, DC, USA), CSEET '98, IEEE Computer Society, 1998, pp. 56–66.

[23] ______, *Priorities for the education and training of software engineers*, J. Syst. Softw. **53** (2000), no. 1, 53–71.

[24] ______, *What knowledge is important to a software professional?*, Computer **33** (2000), no. 5, 44 –50.

[25] Chris Loftus, Lynda Thomas, and Carol Zander, *Can graduating students design: revisited*, Proceedings of the 42nd ACM technical symposium on Computer science education (New York, NY, USA), SIGCSE '11, ACM, 2011, pp. 105–110.

[26] Monica M. McGill, *Defining the expectation gap: a comparison of industry needs and existing game development curriculum*, Proceedings of the 4th International Conference on Foundations of Digital Games (New York, NY, USA), FDG '09, ACM, 2009, pp. 129–136.

[27] Craig S. Miller and Lucia Dettori, *Employers' perspectives on it learning outcomes*, Proceedings of the 9th ACM SIGITE conference on Information technology education (New York, NY, USA), SIGITE '08, ACM, 2008, pp. 213–218.

[28] J. L. Mize, *Making an academic curriculum relevant to business requirements*, SIGCSE Bull. **8** (1976), no. 2, 24–27.

[29] Freeman L. Moore and James T. Streib, *Identifying the gaps between education and training*, Proceedings of the twentieth SIGCSE technical symposium on Computer science education (New York, NY, USA), SIGCSE '89, ACM, 1989, pp. 52–55.

[30] Laurie Murphy and Josh Tenenberg, *Do computer science students know what they know?: a calibration study of data structure knowledge*, Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education (New York, NY, USA), ITiCSE '05, ACM, 2005, pp. 148–152.

[31] The Joint Task Force on Computer Curricula, *Computing curricula 2001*, J. Educ. Resour. Comput. **1** (2001), no. 3.

[32] ______, *Computing curricula 2008*, (2008).

[33] The Joint Task Force on Computing Curricula, *Computing curricula 1991*, Commun. ACM **34** (1991), no. 6, 68–84.

[34] S. Ruff and M. Carter, *Communication learning outcomes from software engineering professionals: A basis for teaching communication in the engineering curriculum*, Frontiers in Education Conference, 2009. FIE '09. 39th IEEE, oct. 2009, pp. 1–6.

[35] Mehran Sahami, Mark Guzdial, Andrew McGettrick, and Steve Roach, *Setting the stage for computing curricula 2013: computer science – report from the acm/ieee-cs joint task force*, Proceedings of the 42nd ACM technical symposium on Computer science education (New York, NY, USA), SIGCSE '11, ACM, 2011, pp. 161–162.

[36] G. Scott and D. N. Wilson, *Tracking and profiling successful it graduates: An exploratory study*, Proceedings of the 13th Australasian Conference on Information Systems, ACIS '02, 2002, pp. 1185–1195.

[37] Chris B. Simmons and Lakisha L. Simmons, *Gaps in the computer science curriculum: an exploratory study of industry professionals*, J. Comput. Sci. Coll. **25** (2010), no. 5, 60–65.

[38] Leigh Ann Sudol and Ciera Jaspan, *Analyzing the strength of undergraduate misconceptions about software engineering*, Proceedings of the Sixth international workshop on Computing education research (New York, NY, USA), ICER '10, ACM, 2010, pp. 31–40.

[39] Sami Surakka, *What subjects and skills are important for software developers?*, Commun. ACM **50** (2007), no. 1, 73–78.

[40] R. H. Trent, *Perspectives on the academic preparation of mis professionals*, Proceedings of the ACM SIGCPR conference on Management of information systems personnel (New York, NY, USA), SIGCPR '88, ACM, 1988, pp. 119–119.

[41] Taehyung (George) Wang, Diane Schwartz, and Robert Lingard, *Assessing student learning in software engineering*, J. Comput. Sci. Coll. **23** (2008), no. 6, 239–248.

[42] Kristina Winbladh, *Requirements engineering: closing the gap between academic supply and industry demand*, Crossroads **10** (2004), no. 4, 4–4.

[43] Charles R. Woratschek and Terri L. Lenox, *Information systems entry-level job skills - a survey of employers*, Journal of Computer Information Systems **2** (2002), no. 2, 119–143.

APPENDIX A. DESCRIPTION OF STUDIES

This section provides a brief description of the studies included in this literature review.

A comparison between the recommendations of Computing Curriculum 1991 and the views of software development managers in Ireland [6]

This study presents the results of a semi-structured interview of sixteen software development managers from various software companies in Ireland. Researchers interviewed these managers about both technical and non-technical topics that these managers felt required additional emphasis in college. The most frequent responses were that graduating students required more knowledge in the areas of software testing and a better understanding of the software development lifecycle. The managers also indicated that a larger emphasis on team work skills and writing ability were necessary.

A replicated survey of software testing practices in the Canadian province of Alberta: What has changed from 2004 to 2009?[11]

This study replicates the work of a previous study investigating software testing practices in Canadian software companies and examining how testing practices have changed over time. Fifty-three professional software engineers from various companies in the province of Alberta were surveyed to gain information about the types of testing used at different companies as well as issues or limitations that were preventing companies from adopting new testing methods. The results of this research indicate that system and unit testing are the most prevalent forms of testing used and that a lack of training is a major factor in hindering companies from adopting new testing methods and tools.

A Survey of the Relevance of Computer Science and Software Engineering Education[22]

This study presents the results of a survey of over one hundred sixty software professionals from multiple companies. To complete the survey, subjects were asked to rate their current level of knowledge for multiple computer science and software engineering topics and to also rank their level of knowledge in those areas when they graduated college. The areas where respondents had the largest difference in present knowledge as opposed to when they graduated college were: configuration management, testing and quality assurance, software maintenance, project management, and UI design. Non-technical areas just as ethics and professionalism and writing ability were also noted as areas where software professionals showed large differences since graduating college.

Analyzing the strength of undergraduate misconceptions about software engineering[38]

This study presents information about misconceptions concerning software engineering that are held by senior level college students majoring in computer science. Researchers surveyed forty-five students and twenty-nine software professionals to determine if students held misconceptions related to software engineering by asking a series of questions related to twelve predetermined misconceptions about software engineering. The results indicated that over half of the surveyed students held mistaken beliefs about encapsulation and software defects, but understood the importance of development processes, teamwork, and system requirements.

Can Graduating Students Design Software Systems?[8]

This study presents the results of a multi-institutional experiment to evaluate the ability of senior-level students to produce quality software design. One hundred fifty students from colleges in the United States, the United Kingdom, Sweden, and other countries worked in teams to complete a small design assignment. Researchers used a rubric to grade the designs and reported that a majority of the designs produced by the student teams were of poor quality or inadequate. The researchers also indicated that students lacked a general understanding of the type of information that should be present in a software design and that these students had difficulties communicating information in their designs because they did not use UML diagrams, sequence diagrams, etc. to model software behavior.

Can Graduating Students Design: Revisited[25]

This study replicates a previous study investigating whether or not senior-level college students are capable of producing good software designs. Approximately sixty subjects worked on teams to complete a simple design assignment. Researchers used a rubric a score the student designs and found that most groups did not produce satisfactory designs, affirming the results of the previous study. Researchers also asked the student groups to rank the different designs in order of which was best and found a high degree of correlation between the student rankings and the scores produced by the rubric used to grade the designs. The researchers concluded that although graduating students may not be capable of producing good design, they possess the ability to recognize good design.

Defining the Expectation Gap: A Comparison of Industry Needs and Existing Game Development Curriculum[26]

This study explores the gap between the needs of the game development industry and the curriculum for game development at multiple universities. Twenty-six hiring personnel at game development companies were surveyed to establish baseline expectations for newly graduated developers. These results were then compared against the results of a survey completed by fifteen universities with game development programs. Researchers reported that the largest gap between employer expectations and the college curriculum were in the areas of multi-threaded programming and tool development. Game development companies also placed a strong emphasis on programming languages such as C++, Lua, and Perl. The results also indicated that game development companies felt that colleges did not spend enough time preparing students to assume leadership roles.

Employers' Perspectives on IT Learning Outcome[27]

This study reports the results of a survey of ten IT professionals who are responsible for hiring programmers, database administrators, and software architects, etc. at their respective companies. The surveyed professionals were asked to give both free-form answers and to respond to questions related to a small set of pre-selected competencies. The results indicated that the respondents overwhelmingly agreed that the pre-selected competencies such as the ability to apply abstraction and understanding object-oriented interface design were important skills. Multiple respondents also indicated that teamwork experience and language-independent programming abilities were important as part of the free-form responses.

Information technology workforce skills: The software and IT services provider perspective[12]

This research presents the results of a web-based survey of IT professionals designed to replicate and extend an earlier study. There were one hundred four responses to the survey, the vast majority of which came from individuals in North America, but approximately twenty-five percent were from respondents in India, Russia, Australia, and other locations. The results of the survey indicated that communication, system testing, and programming were among the most important skills for new hires to possess. The study also compared importance rankings with a previous study to evaluate how the importance of different skills was changing over time. The results show that abilities such as project planning and managing customer relationships have become more important to businesses.

Industry perceptions of the knowledge, skills, and abilities needed by computer programmers[2]

This study presents the results of a web-based survey of IT professionals and interviews with top-level managers, supervisors, and directors about necessary skills and knowledge for computer programmers. The results include responses from over three hundred professionals who responded to the online survey as well as responses from the site interviews conducted at five separate companies. Participants were asked to rank skills based on their importance. Among the technical skills ranked as the most important were the ability to modify programs written by others, the ability to debug programs, and coding ability. Non-technical skills such as problem-solving ability and good listening skills were also highly ranked.

Novice Software Developers, All Over Again[3]

This research presents the results of a case study of eight recently graduated college students beginning employment at Microsoft. Researchers monitored the new employees over a period of six months, looking for common issues that these employees experienced and other problems that they struggled with. The study reported that the most common struggles for these new developers were using the revision control system, debugging software, interacting with team members, and communicating a need for help.

Struggles of New College Graduates in their First Software Development Job[4]

This research presents results from a case study of the same eight subjects as [3]. In addition to examining the technical and non-technical areas in which these subjects had difficulties, this research also examined common misconceptions held by these new developers. These included notions such as immediately fixing any software defects that they found and feeling as though they needed to be seen as knowledgeable and independent in the eyes of their manager in order to succeed. Researchers also noted that new software developers have a difficult time determining that they are stuck on a problem and should seek assistance from others.

Information Systems Entry-Level Job Skills: A Survey of Employers[43]

This research presents the results of a survey of companies that hire information systems graduates. Twenty-four employers provided responses to a variety of questions such as the importance of certifications, preferences for programming languages, and how the knowledge and abilities of recently graduated students compared to employer expectations. The areas with the largest reported gaps in technical areas included understanding the system development lifecycle, networking concepts, and data modeling. The results also indicated large gaps in non-technical areas such as written communication, problem solving, time management, attention to detail, and oral communication.

What Game Developers Look for in a New Graduate: Interviews and Surveys at One Game Company[39]

This study presents the results of surveys and interviews with game developers, artists and managers at one video game development company. Nine employees participated in an interview process where researchers asked participants about important skills and abilities for new developers. After examining responses and identifying the most common elements, researchers created a survey that was given to thirty-two other employees at the company. The results from the study indicate that there are several important skills and abilities for new graduates, including: a strong knowledge of C++, a good understanding of data structures, the ability to work well with others, and the ability to write clean, maintainable code.

What subjects and skills are important for software developers?[16]

This research serves as a quasi-replication of a previous study looking at the skill requirements and knowledge deficiencies among software professionals [22]. This study reports on the results of eleven Finnish software developers as well as nineteen professors and lecturers at Finnish universities and twenty-four graduate students from one university in Finland. The results were then analyzed to determine which areas had become more important since the original study and to determine if developers, educators, and students perceived differences in the importance of certain areas. Among the results, areas such as object oriented programming, data structures, and configuration management were rated as more important by surveyed developers than the original study. One other notable result was that the survey students placed much less importance on mathematics and theoretical computer science concepts than did software professionals and educators.

Gaps in the computer science curriculum: an exploratory study of industry professionals[37]

This study presents the results of interviews with IT professionals about how computer science curricula could be improved. Twenty IT professionals from a mixture of large Fortune 500 companies, small-to-medium businesses, and non-profit organizations responded to seven open-ended questions related to the educational needs of undergraduate computer science students. Among some of the most prevalent and important results were the ability to elicit requirements, writing skills, and a knowledge of process improvement frameworks.

Evaluating computing education programs against real world needs[10]

This research presents the results of a survey of sixty-eight businesses located in the southern United States that hire computer science graduates. The results from this study indicated that employers placed a much stronger emphasis on non-technical skills such as problem solving, critical thinking, and management ability than technical skills. Additionally, the results indicated that employers indicated that it was better to have good general programming skills than it was for any one single language. Other identified areas of importance were networking skills as well as knowledge of Microsoft's SQL server (as opposed to MySQL or Oracle databases).

Changing qualifications for entry-level application developers[9]

This study presents the results of a survey given to corporate CIOs and other hiring personnel hiring entry-level college graduates in application development roles. Respondents from one hundred companies were asked to participate in the study and answer questions related to the importance of different skills and abilities as well as how those skills and abilities were changing over time. The results of the study indicate that general programming ability, data structure knowledge, oral communication skills, team work ability, and internship experience were rated as the most important qualifications for college graduates. Respondents also noted a growing importance of other areas such as human computer interaction knowledge, software development methodologies, and ethics.

Tracking and Profiling Successful IT Graduates: an exploratory study[36]

This study presents the results of interviews and surveys conducted with recently graduated students who had been identified by IT employers as being successful in the beginning years of their careers. Thirty-four respondents were asked to rate the importance of various skills in their careers and to what extend their college education had prepared them for those abilities. The results of the surveys indicated that a willingness to learn from mistakes, the ability to work as part of a team, and being able to take responsibility were rated as the most important skills. Problem solving abilities were as cited as important. Respondents indicated that being able to openly take feedback and learn from errors as well as using previous knowledge to new problems were areas in which college did not adequately prepare students.

Assessing Student Learning in Software Engineering[41]

This study presents the results of an assessment performed by the computer science department to measure student understanding of software design and development principles and their ability to apply those principles to real world problems. The assessment was conducted in two phases over two years, with seventy-four students participating in the first phase and eighty-two students participating in the second phase. The results from the first phase of the assessment indicated that a majority of students had a less than adequate understanding of several software engineering concepts. When taken in conjunction with results from the second phase of the assessment, the researchers concluded that students had difficulties designing software before attempting to implement it and that students struggled with using formal or semi-formal modeling techniques to describe their designs. Researchers also found that students who had taken graduate level courses related to software design performed significantly better during the assessment than those who had not taken graduate-level courses.

Post Graduate Assessment of CS Students: Experience and Position Paper[13]

This research presents the results of a post-graduate case study of computer science students at a software development firm specializing in eCommerce. Twelve recent graduates who had been hired within the past year were evaluated through a series of interviews, discussions, and code reviews in order to assess their technical abilities and cognitive skills. The results of the technical assessment indicated that the developers were more likely to possess introductory-level skills in SQL as opposed to intermediate-level skills in HTML and programming. Researchers also reported that new developers had multiple cognitive issues including the inability to utilize computer science concepts in code development, an inability to analyze and design algorithms, an inability to find alternative solutions or designs for a given problem, and difficulties conducting thorough code testing.

Computer Technology Students - What skills do they really need?[15]

This research presents the results of a survey of graduating students who had been recently hired by various Australian companies. Twenty-two individuals responded to the survey, indicating the skills and abilities that were most important in their jobs and the applicability of the education that they had received in college. Results of the study indicated that programming and time management were identified as highly important by the majority of respondents. Technical writing abilities were also identified as important by many of the respondents.

Employer Satisfaction with ITC Graduates[14]

This study reports the results of a survey of employers of Australian ICT (Information and Communications Technology) graduates. Approximately two hundred companies completed the survey, responding to questions about their satisfaction with recently hired graduates and the shortcomings in the education of those graduates. Respondents indicated that it was important for students to have internship experience and that educators should provide students with better awareness of industry expectations. Project management, business processes, written communication skills, and the ability to communicate with clients were also cited as areas where colleges could improve student education. Researchers also indicated that respondents did not have a particular preference for any one programming language as some companies indicated that they preferred students to have knowledge of the latest programming languages, whereas others stressed an importance of legacy languages such as COBOL.

The Case for ICT work-integrated learning from graduates in the workplace[21]

This research presents the results of a large survey given to recent ICT (Information and Communications Technology) graduates in Australia. Seven hundred nineteen students who had received at ITC degree within the past five years responded to survey questions about how their education had prepared them for their current career, topics missing from their college education, and suggestions for improving college courses. Respondents indicated that theoretical knowledge was important in their work, but that technological knowledge was also important. Among the issues with their college education, respondents reported a lack of real-world experience and out of date technology and programming languages as the largest problems.

Do Computer Science Students Know What They Know? A Calibration Study of Data Structure Knowledge[30]

This research presents the results of an empirical study used to measure student knowledge of data structures and to evaluate whether students are able to adequately judge their own knowledge of a topic. Sixty-one students enrolled in two different U.S. universities were given a series of questions related to data structures taken from GRE (graduate record examination) and AP (advanced placement) practice tests. The results of the study indicate that the students' predictions of their performance on the test were strongly correlated with their actual results, indicating that students can accurately measure their own knowledge of knowledge areas. Researchers also found that the students performed mostly poorly on questions related to hash tables, recursive binary trees, and distinguishing between different searching and sorting algorithms.

Priorities for the education and training of software engineers[23]

This study presents the results of a survey given to one hundred eighty-six software professionals predominately from the United States and Canada. Participants were asked questions about their current knowledge of topic areas, how much they learned about a topic in their college education, the usefulness of a topic in their professional career, and how the learning of a topic has influenced their thinking. The results of the survey indicate that programming languages, data structures, and software design are viewed as the most important areas by software professionals. The results also show that software professionals required the largest amount of on-the-job training in the areas of configuration management, project management, and software maintenance.

What Knowledge Is Important to a Software Professional[24]

This research presents the results of a survey given to software programmers and managers to measure their knowledge of several skills, their perception of the importance of those skills, and the amount of additional knowledge that they have had to gain in those skills since graduating college. One hundred eighty-one software professionals from the United States, Canada, and various European countries were chosen for participation in the study. The results of the study revealed multiple areas in which respondents reported a great deal of on-the-job learning, including: configuration management, project management, software testing, software maintenance, and object-oriented concepts. The study also indicated that additional training for professionals may be necessary in areas such as technical writing, leadership ability, and user interface design.

Evaluating the testing ability of senior-level computer science students[7]

This research presents the results of a study used to evaluate the testing ability of senior-level college students. Forty-one subjects from two classes were given assignments to measure their ability to create test cases for a short computer programming both with and without tool assistance. The results of the study indicate that students were unable to reach complete test coverage without tool assistance and that when using software tools, students were able to significantly increase statement, branch, and conditional coverage. The researchers also indicated that even with tool assistance, students tended to produce a large number of redundant test cases, indicating a lack of proficiency with software tools and a lack of understanding in creating minimal test suites.

APPENDIX B. DESCRIPTION OF KNOWLEDGE DEFICIENCIES

Algorithm Development and Analysis: Deficiencies related to the ability to develop new or alternative algorithms as solutions to a problem, or possessing an understanding necessary to analyze the space and time complexity of an algorithm. Business Processes: Deficiencies related to the understanding of the procedures and activities related to the internal operation of a company.

Configuration Management: Deficiencies related to the use of configuration management tools or the underlying concepts of the importance of configuration management.

Computer Science Theory: Deficiencies related to topics generally classified as theoretical computer science, such as logic, automata theory, etc.

Data Structures: Deficiencies related to the understanding of data structures and selecting appropriate structures to solve a given problem.

Debugging: Deficiencies related to the use of software debugging tools or the ability to analyze error messages generated by compilers and produce appropriate fixes.

Documentation: Deficiencies related to the ability to produce documentation for software designs or other development activities.

Ethics: Deficiencies related to an understanding or expression of ethical and professional behavior.

Leadership: Deficiencies related to the ability to express characteristics related to leadership such as the ability to take on responsibility and working as part of a team.

Management: Deficiencies related to the ability to effectively manage teams of people and other resources.

Multithreaded and Parallel Programming: Deficiencies related to the understanding of parallel computation or ability to develop multithreaded code or use parallelization techniques or frameworks.

Networking: Deficiencies related to the understanding of networking concepts or the ability to work with computer networks.

Object Orientation: Deficiencies related to the understanding of object-oriented design, programming, or other object-oriented concepts.

Oral Communication: Deficiencies related to the ability to effectively

Presentation: Deficiencies related to the ability to present ideas to others or to conduct a formal presentation.

Problem Solving: Deficiencies related to the ability to produce solutions for a given problem.

Programming: Deficiencies related to the understanding of general programming concepts or the ability to program or apply concepts to different languages.

Programming Languages: Deficiencies related to the ability to effectively use a given programming language.

Project Management: Deficiencies related to the understanding of the project management concepts or the ability to oversee a project from initiation to completion.

Quality Assurance: Deficiencies related the understanding of quality assurance standards and activities or the ability to perform those activities.

Real World Experience: Deficiencies related to the ability to apply theoretical concepts to real world problems.

Requirement Gathering and Analysis: Deficiencies related to requirements elicitation and analysis or the ability to perform those activities.

Security: Deficiencies related to the ability to create secure software or the understanding of security concepts.

Software Design: Deficiencies related to the ability to produce software designs or an understanding of software design techniques and activities.

Software Lifecycle: Deficiencies related to the understanding of the software development process, the different phases of which it is comprised, and how those phases are connected.

Software Maintenance: Deficiencies related to the understanding activities and processes performed on shipping software products or the ability to perform those activities.

Software Development Processes: Deficiencies related to the understanding of software development processes such as a CMMI process or RUP, or the ability to follow a defined software development process.

Teamwork: Deficiencies related to the abilities necessary to successfully work as part of a group.

Testing: Deficiencies related to the understanding of testing methods and techniques or the ability to develop test cases or suits for software.

Tool Development: Deficiencies related to the ability to develop, maintain, or extend software development tools that will be used by other developers.

User Interface Design: Deficiencies related to the understanding of good user interface design principles or the ability to develop user interfaces for software projects.

Writing: Deficiencies related to the ability to clearly express ideas in writing or an understanding of technical writing concepts.

APPENDIX C. SURVEY INSTRUMENT

University from which you have received your degree:

In which of the following fields are you most interested? Check any that apply:

- Software Engineering
- Information Technology
- Computer Engineering
- Information Systems
- Computer Science
- Systems Analyst
- Other:

Have you already been hired for a job or internship?

- Yes
- No
- No answer

How would you describe your teamwork abilities and experience? Choose one of the following answers:

1. I have had almost no teamwork experience or feel that my teamwork skills are non-existent.
2. I have a small amount of teamwork experience or feel that my teamwork skills are below average.

3. I have a moderate amount of teamwork experience or feel that my teamwork skills are average.

4. I have had a lot of teamwork experience or feel that my teamwork skills are good.

5. I have had a significant amount of teamwork experience or feel that my teamwork skills are exceptional.

How would you describe your written communication and technical writing skills? Choose one of the following answers:

1. I feel that my written communication and technical writing skills are poor and I have not taken any courses focused on improving those skills.

2. I feel that my written communication and technical writing skills are poor, but I have taken at least one course focused on improving those skills.

3. I feel that my written communication and technical writing skills are good, but I have not taken any courses focused on improving those skills.

4. I feel that my written communication and technical writing skills are good, and I have taken at least one course focused on improving those skills.

5. I feel that my written communication and technical writing skills are good, and I have taken multiple courses focused on improving those skills.

How would you describe your presentation abilities and experience? Choose one of the following answers:

1. I have had almost no experience with presenting or feel that my presentation skills are non-existent.

2. I have a small amount of experience with presenting or feel that my presentation skills are below average.
3. I have had moderate experience with presenting or feel that my presentation skills are average.
4. I have a lot of experience with presenting or feel that my presentation skills are above average.
5. I have had significant experience with presenting or feel that my presentation skills are exceptional.

How would you describe your oral communication skills? Choose one of the following answers:

1. I feel that my oral communication skills and presentation skills are poor and I have not taken any courses focused on improving those skills.
2. I feel that my oral communication skills and presentation skills are poor, but I have taken at least one course focused on improving those skills.
3. I feel that my oral communication skills and presentation skills are good, but I have not taken any courses focused on improving those skills.
4. I feel that my oral communication skills and presentation skills are good, and I have taken at least one course focused on improving those skills.

How would you describe your problem solving ability? Choose one of the following answers:

1. I feel as though I am barely or not at all able to find solutions to problems or that my problem solving ability is poor.

2. I feel as though I struggle to find solutions to problems or that my problem solving ability is below average.

3. I feel as though I am generally able to find solutions to problems or that my problem solving ability is average.

4. I feel as though I am almost always able to find solutions to problems or that my problem solving ability is above average.

5. I feel as though I am usually able to find multiple solutions to problems or that my problem solving ability is exceptional.

How would you describe your familiarity with configuration management tools? Choose one of the following answers:

1. I am not at all familiar with configuration management tools and did not learn about them or use any in my courses.

2. I am not very familiar with configuration management tools, but recall learning about them in one of my courses.

3. I am familiar with configuration management tools and have used them in at least one of my courses.

4. I am familiar with configuration management tools and have used them in several of my courses.

5. I am very familiar with configuration management tools and have used them for my own personal projects in addition to my courses.

How would you describe your familiarity with debugging software tools? Choose one of the following answers:

1. I am not at all familiar with using debugging tools and do not use their features when debugging.

2. I am not very familiar with using debugging tools and feel they make debugging more difficult for me.

3. I am familiar with using debugging tools, but do not know many features beyond the basics.

4. I am highly familiar with using debugging tools and know some advanced features of these tools.

5. I am exceptionally familiar with using debugging tools and use many advanced features when debugging.

How would you describe your familiarity with unit testing frameworks? Choose one of the following answers:

1. I am not at all familiar with unit testing frameworks and did not learn about them or use any in my courses.

2. I am not very familiar with unit testing frameworks, but recall learning about them in one of my courses.

3. I am familiar with unit testing frameworks and have used them in at least one of my courses.

4. I am familiar with unit testing frameworks and have used them in several of my courses.

5. I am very familiar with unit testing frameworks and have used them for my own personal projects in addition to my courses.

How would you describe your familiarity with software development processes? Choose one of the following answers:

1. I am not at all familiar with software development processes and did not learn about them or use any in my courses.
2. I am not very familiar with software development processes, but recall learning about them in one of my courses.
3. I am familiar with software development processes and have used them in at least one of my courses.
4. I am familiar with software development processes and have used them in several of my courses.
5. I am very familiar with software development processes and have used them for my own personal projects in addition to my courses.

How would you describe your familiarity with test automation tools? Choose one of the following answers:

1. I am not at all familiar with test automation tools and did not learn about them or use any in my courses.
2. I am not very familiar with test automation tools, but recall learning about them in one of my courses.
3. I am familiar with test automation tools and have used them in at least one of my courses.
4. I am familiar with test automation tools and have used them in several of my courses.

5. I am very familiar with test automation tools and have used them for my own personal projects in addition to my courses.

How would you describe your familiarity with software maintenance? Choose one of the following answers:

1. I have little or no experience with maintaining existing software and did not learn about software maintenance in any of my courses.
2. I have little or no experience with maintaining existing software, but recall learning about software maintenance in one of my courses.
3. I have some experience with maintaining existing software and I learned about software maintenance in one of my courses.
4. I have some experience with maintaining existing software and I learned about software maintenance in several of my courses.
5. I have extensive experience with maintaining existing software and have learned about and practiced software maintanance on my own projects in addition to learning about it in my courses.

How would you describe your knowledge of computer networking? Choose one of the following answers:

1. I am not at all familiar with computer networking and did not learn about it any of my courses.
2. I am not very familiar with computer networking, but recall learning about it in my courses.
3. I am familiar with computer networking and learned about it in one of my courses.

4. I am fairly familiar with computer networking and have learned about it in several of my courses.

5. I am very familiar with computer networking and have studied it on my own time in addition to learning about it in my courses.

How would you describe your knowledge of multi-threaded or parallel programming? Choose one of the following answers:

1. I am not at all familiar with multi-threaded or parallel programming and did not learn about it any of my courses.

2. I am not very familiar with multi-threaded or parallel programming, but recall learning about it in my courses.

3. I am familiar with multi-threaded or parallel programming and learned about it in one of my courses.

4. I am fairly familiar with multi-threaded or parallel programming and have learned about it in several of my courses.

5. I am very familiar with multi-threaded or parallel programming and have studied it on my own time in addition to learning about it in my courses.

How would you describe your knowledge of user interface design? Choose one of the following answers:

1. I am not at all familiar with user interface design and did not learn about it any of my courses.

2. I am not very familiar with user interface design, but recall learning about it in my courses.

3. I am familiar with user interface design and learned about it in one of my courses.
4. I am fairly familiar with user interface design and have learned about it in several of my courses.
5. I am very familiar with user interface design and have studied it on my own time in addition to learning about it in my courses.

How would you describe your knowledge of data structures? Choose one of the following answers:

1. I am not at all familiar with data structures and did not learn about it any of my courses.
2. I am not very familiar with data structures, but recall learning about it in my courses.
3. I am familiar with data structures and learned about it in one of my courses.
4. I am fairly familiar with data structures and have learned about it in several of my courses.
5. I am very familiar with data structures and have studied it on my own time in addition to learning about it in my courses.

How would you describe your familiarity with designing and interacting with databases? Choose one of the following answers:

1. I am not at all familiar with databases and did not learn about them or use them in any of my courses.
2. I am not very familiar with databases, but do recall learning about them in one of my courses.

3. I am familiar with databases and have both used and learned about them in one of my courses.
4. I am fairly familiar with databases and have used and learned about them in several of my courses.
5. I am very familiar with databases and have studied and used them on my own outside of my courses.

How would you describe your knowledge of the software development lifecycle? Choose one of the following answers:

1. I am not at all familiar with the software development lifecycle and did not learn about it any of my courses.
2. I am not very familiar with the software development lifecycle, but recall learning about it in my courses.
3. I am familiar with the software development lifecycle and learned about it in one of my courses.
4. I am fairly familiar with the software development lifecycle and have taken several courses focusing on one or more aspect of it.
5. I am very familiar with the software development lifecycle and have taken courses that focused on most or all aspects of the software development lifecycle, or have worked on a project that spanned all phases of the software development lifecycle.

How would you describe your knowledge of software testing? Choose one of the following answers:

1. I am not at all familiar with software testing and did not learn about it any of my courses.
2. I am not very familiar with software testing, but recall learning about it in my courses.
3. I am familiar with software testing and learned about it in one of my courses.
4. I am fairly familiar with software testing and have learned about it in several of my courses.
5. I am very familiar with software testing and have studied it on my own time in addition to learning about it in my courses.

How would you describe your knowledge of requirements elicitation? Choose one of the following answers:

1. I am not at all familiar with requirements elicitation and did not learn about it any of my courses.
2. I am not very familiar with requirements elicitation, but recall learning about it in my courses.
3. I am familiar with requirements elicitation and learned about it in one of my courses.
4. I am fairly familiar with requirements elicitation and have learned about it in several of my courses.
5. I am very familiar with requirements elicitation and have performed requirements gathering for projects outside of school in addition to learning about it in my courses.

How would you describe your knowledge of software design? Choose one of the following answers:

1. I am not at all familiar with software design and did not learn about it any of my courses.
2. I am not very familiar with software design, but recall learning about it in my courses.
3. I am familiar with software design and learned about it in one of my courses.
4. I am fairly familiar with software design and have learned about it in several of my courses.
5. I am very familiar with software design and have designed software systems on my own time in addition to learning about it in my courses.

How would you describe your familiarity with object oriented programming languages? Choose one of the following answers:

1. I am not at all familiar with object oriented programming languages and did not learn about them or use any in my courses.
2. I am not very familiar with object oriented programming languages, but recall learning about them in one of my courses.
3. I am familiar with object oriented programming languages and have used one in at least one of my courses.
4. I am familiar with object oriented programming languages and have used one in several of my courses.
5. I am very familiar with object oriented programming languages and have used one for my own personal projects in addition to my courses.

How would you describe your ability to apply concepts from one programming language to another programming language? Choose one of the following answers:

1. I do not feel I am at all able to apply programming concepts from one programming language to another, new programming language.

2. I do not feel I am easily able to apply programming concepts from one programming language to another, new programming language.

3. I feel moderately able to apply programming concepts from one programming language to another, new programming language.

4. I feel quite able to apply programming concepts from one programming language to another, new programming language, but I have not had much experience doing so.

5. I feel extremely able to apply programming concepts from one programming language to another, new programming language, and I have had experience doing so.

How would you describe your ability to convert algorithms, constructs, etc. from natural language descriptions into code? Choose one of the following answers:

1. I do not feel I am at all able to convert algorithms, constructs, etc. from naturual language descriptions into code.

2. I do not feel I am easily able to convert algorithms, constructs, etc. from naturual language descriptions into code.

3. I feel I am moderately able to convert algorithms, constructs, etc. from naturual language descriptions into code.

4. I feel I am quite able to convert algorithms, constructs, etc. from naturual language descriptions into code.

5. I feel extremely able to convert algorithms, constructs, etc. from naturual language descriptions into code.

How many different programming languages would you say that you know well? Choose one of the following answers:

- One.
- Two.
- Three.
- Four or more.

Overall, do you feel your education and experiences at your school have prepared you for your future career? Choose one of the following answers:

- Yes
- No
- Unsure

Please list any skills, concepts, etc. that you feel you will need in your career, but were not prepared for in college:

Please list any skills, concepts, etc. that you feel you will not need in your future career, but were required to learn in college: