# THE EDUCATIONAL EXPERIENCES OF SOFTWARE DESIGNERS WORKING IN EDUCATION/INSTRUCTIONAL TECHNOLOGY RELATED FIELDS

Marisa E. Exter

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the Department of Instructional Systems Technology,
Indiana University
December 2011




Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

_____

Elizabeth Boling

_____

Barbara Bichelmeyer

_____

Kay Connelly

_____

Martin Siegel

July 20, 2011

Acknowledgements

I would like to acknowledge the following individuals for all of their assistance and understanding along the way. I would like to thank Elizabeth Boling for her availability, advice, and understanding at every step, and the rest of my dissertation committee, for their feedback on my proposal and at other times throughout the process. I would also like to thank Elizabeth Boling, Barbara Bichelmeyer, Nilufer Korkmaz, Nichole Turnage, and Dr. Andrew Gibbons for reviewing my instruments, and Miguel Lara, Rod Meyers, Micah Model, and Prashant Sabhnani, for assistance in testing the survey instrument. Thanks to Elizabeth Boling, Nilufer Korkmaz, and Nichole Turnage for reviewing my coding structure. Thanks to Ali Korkmaz and Simina Boca, for discussion and advice regarding the statistical procedures used. I would also like to thank my participants, especially the interviewees, who spent a considerable amount of time to give thoughtful responses and who were very encouraging about the study. A special thanks to those who participated in member checking.

Finally, I would like to thank my family for all of their support over the past years and for supporting me throughout my education. Thanks to my parents, Ben and Arlene Widrevitz, and my brother Dan Widrevitz and his wife Simina Boca for ongoing encouragement and reminders that I need to finish already. Thanks also to my mother-in-law Maureen Exter for proofreading nearly the entire manuscript!

Last but not least, much thanks to my husband Max Exter, for helping to revise and test instruments, proofreading, ongoing technical support to keep this laptop going until I finished, influx of baked goods when necessary, and for his general support throughout the process. Oh yes, and thanks to my son Ephrem Exter, without whom I would not be finishing my dissertation this particular year. Thanks for all the fun and funny times.

Marisa E. Exter

# THE EDUCATIONAL EXPERIENCES OF SOFTWARE DESIGNERS WORKING IN EDUCATION/INSTRUCTIONAL TECHNOLOGY RELATED FIELDS

As custom-built educational software becomes ever more complex, there is an increasing need for software design skills (including software architecture, business and technical requirements gathering, high- and low-level design, and programming) to produce high-quality software. However, it is unclear whether there is a typical educational path for computing professionals working on this area, or to what degree these software designers feel that domain-specific knowledge is required in order to succeed in this area. This three-phase mixed-methods study explores the formal (university) and non-formal (including work-sponsored, self-taught, and informal) educational experiences of software designers currently working in this field. Gaps between what is needed on the job and what is taught in school are highlighted, and participants' recommendations for improving educational programs to prepare students for entering this field are summarized. Implications for researchers, educators, and hiring managers are discussed.

Findings indicate that software design professionals come from variety of backgrounds, which include multiple formal educational paths and a wide variety of life experiences. Computing fields (such as Computer Science) and Instructional Design are two common starting points for professionals in this field. Regardless of formal educational background software designers typically play a number of roles over time, both within and outside of educational software development. Participants indicate that critical thinking, communication skills, and the ability to learn on one's own are among the most important competencies needed on the job, and

that these should be taught alongside Computing and/or Instructional Design foundations. Recommendations for educational programs focus on developing those skills through real-world experiences such as team projects.

# Table of Contents

The Educational Experiences of Software Designers working in Educational/Instructional Technology Related Fields

**Marisa Exter**

# 1 Chapter 1: Statement of the Problem

The purpose of this study is to explore the types of formal (university) and non-formal (including work-sponsored, self-taught, and informal) educational experiences software designers currently working in educational technology related fields have experienced, and the ways in which these software designers report these experiences have (or have not) prepared them for their current roles.

This study is part of a larger research agenda related to software design practitioners' reflections on their own formal and non-formal educational experiences, the ways in which these educational experiences prepare them for their professional roles, and the implications these professionals' recommendations for formal educational program improvement may have for design education in general and software design education in particular. A related exploratory study examined the formal and non-formal educational experiences of software designers working across many different fields (Exter & Turnage, 2011). The participants, especially the most experienced participants, described a number of attitudes which appear to guide their design processes as well as the approach they take towards learning what they need for each new project. These attitudes include a strong emphasis on self-learning, and a willingness to experiment with samples provided by others or their own earlier work. They also described strategies which help them as they move from project to project and from technology to technology or programming language to programming language. These include looking for commonalities with technologies, programming languages, or systems they already are familiar with, breaking projects into smaller pieces and testing each piece before going on to the next, creating small prototype systems or pieces of code to learn how a new technique works before

incorporating it into a larger design, learning better design practices by looking at others' code and design documents, and constantly watching websites, blogs, and journals for new ideas which may be incorporated into their own future designs.

Participants indicated disappointment that these types of attitudes and strategies were not central to their own formal educational experiences, which for the most part focused on math, science, and specific programming languages. The majority of participants, especially those with the most experience (who have been out of school the longest) no longer use many of the specific technologies or programming languages they learned during their university experiences, but they continue to value the underlying concepts and skills addressed in university courses. Participants indicate that courses outside of their major were of special value in helping them develop social, critical thinking, and writing skills.

When asked how the undergraduate university experience could be improved, the majority of the interviewees stressed the importance of real-world projects which are large and complex, and necessitate realistic aspects such as working in teams, working with materials created by others, and working with "fuzzy" problems. They believe that students need to learn theory, basic concepts, and one or more programming languages prior to embarking on a realistic project. Although future jobs may not require the use of these specific technologies and programming languages, they are necessary foundations for tackling real problems.

Although many of the examples given by participants were industry-specific and focused on technologies and techniques which were particular to the domain for which they were producing software (e.g. telecom, retail, financial, web design, etc.), the attitudes, strategies, and suggestions for improvement for educational systems were remarkably generic and appropriate for almost any type of software design. However, there were differences in participant attitudes.

At this point it is unclear if the differences are a result of the industry in which they were working, or differences in personality or educational background. Taking a detailed look at one or more specific domains will allow me to learn more about the degree to which domain-specific knowledge and skills are important on the job, as well as allowing me to ask whether domain-specific formal education appears to be useful. The current study will explore the experiences of one such group: software designers focused on the creation of educational software.

Both the instructional design and software design fields have relatively well-developed cultures which value both formal educational programs and opportunities for ongoing training and self-improvement in the field. As instructional design projects increasingly make use of technology, often including custom-built software, the need for software design (including software architecture, business and technical requirements gathering, high- and low-level design, and programming) within companies or educational institutions which produce this software is clear. However, little literature has been found that discusses how people are prepared for these roles within the domain of instruction and education. This study will examine the roles software designers play within educational software projects, and the formal and non-formal education they have received. Gaps or areas for improvement in educational preparation and continuing education as identified by software design professionals working in the field will be highlighted. Possibilities for improving educational support for software designers in this field will be discussed.

I have personally worked in multiple software-design environments. For seven years I worked as a software developer in the telecommunications industry, and have spent the last six years designing, developing, and most recently serving as project manager for a project involving a web-based software tool used by students across a range of ages and contexts. I have

observed and trained others in each of these environments. These personal experiences are not directly comparable. In my professional position at Lucent Technologies, I was a member of a team of several dozen members, which worked with numerous other teams of software developers, systems engineers, systems architects, quality assurance testers, technical documentation writers, and others. Although I experienced the full life-cycle of several projects and gave input at each stage in the process, my primary task was the design and development of software systems. For the first four years on the Critical Web Reader project here at Indiana University, I was the main designer and developer, with occasional support from hourly employees with whom I work closely to manage the design process. I was responsible for the entire technical design, development, implementation, and testing of the project from end to end. I had and continue to have an influence on curricular design decisions relating to the software tool as well. Since the project has grown, I have continued to be a key player in design discussions and decision-making, although my development responsibilities have lowered as my management responsibilities have grown.

Clearly, my experiences do not represent those of all software designers in either industry. However, discussions with others and initial findings of earlier studies on software designers across a range of industries indicate there are a few areas in which many educational software initiatives differ from organizations which create software within the telecommunications and similar industries. For example, in the telecommunications industry members of lower and middle management (those who actually interact with software designers) typically have experience as software designers or engineers themselves. In contrast, many educational software initiatives may be led by one or more instructional designers, or other types of specialists who do not have a formal background in software design (such as a degree in

Computer Science or Software Engineering or a related field). This may be particularly true in small educational technology firms and projects directed by university faculty in schools of education or other non-software-design related colleges. This distinction is likely to have an impact on the relationship between management and software designers. Without management who understand the software design process, one or more individual software designers may be required to contribute the expertise necessary to assess whether a desired solution is possible, choose the appropriate technologies, and fulfill the varied software design and development roles, each of which require education and years of practice to master. Managers without a formal software design background may not know the educational or work background necessary in an employee required to fulfill all of these roles. Therefore, they may also not know how to identify and attract personnel with the desired level of experience and assess the fit between an interviewee and the intended project.

These conjectures have led me to wonder whether there is a typical educational path for software designers working on educational software. Based on personal experience, I would suspect that Computer Science or Software Engineering programs would be more appropriate to prepare professionals for roles which involve a high level of complex software design than Instructional Technology related programs, although it is likely that self-taught individuals from a variety of backgrounds may play a role in software design, particularly in small companies and research-related projects conducted by small groups based in a university setting. Standards created by the professional organizations such as Association for Computing Machinery (ACM) and Institute for Electrical and Electronics Engineers (IEEE) and the Liberal Arts Computer Science Consortium tend to focus on general skills which the organizations believe all software designers need (Atlee, LeBlanc, Lethbridge, Sobel, & Thompson, 2006; LACS, 2007).

Standards addressing Instructional Designers, such as those developed by International Board of Standards for Training, Performance, and Instruction ( IBSTPI) , address competencies relating to Instructional Design, but only touch on certain types of technology (Richey, Fields, & Foxon, 2001). Each of these sets of standards includes at least some coverage of non-domain-specific skills such as those relating to communication, teamwork and project management. Do these standards and the educational programs based on them prepare students to work in a domain such as educational technology? If not, what additional types of experience do they need?

In addition to formal (university) education, ongoing non-formal education is important for professionals (Houle, 1980; Radcliffe & Colletta, 1989). It typically takes about 10 years for designers to gain expertise in their area of specialization (Cross, 2004). Ongoing education is especially important in a field such as software design in which underlying technologies, such as hardware platforms and programming languages, are rapidly evolving. Software designers appear to continue improving their theoretical as well as practical knowledge beyond their years of formal education (Lethbridge, 2000).

Self-directed learning appears to account for the majority of adult learning (Livingstone, 2001; Tough, 1989). It may be particularly important for software design professionals working under managers who do not have a software design background themselves to be able to direct their own learning program as well as their own learning projects. Studying those already working in the field will be a valuable way to determine whether and in what ways this is happening, since experts have greater understanding of their own learning process than novices (Daley, 2000). Looking at a range of software designers from newly graduated to highly experienced professionals will highlight the difference that non-formal education and work

experience make on their perception of their ability to perform in their role and their own insights on their educational experiences.

## 1.1 Research Questions

The central research question addressed by this study is:

What formal and non-formal educational experiences do software designers currently working on educational software report having experienced, and to what extent do they feel these experiences have prepared them for their current roles?

In order to explore this question, several sub-questions will be addressed:

1. *What are the primary role(s) played by the participants?*
2. *What formal education do the participants report having had? In what ways do they perceive these experiences have prepared them for their current role(s)?*
3. *Where are there gaps in the competencies (e.g. skills, knowledge, and attitudes) acquired through the formal educational experiences of these software designers? What topics are underemphasized by formal educational experiences?*
4. *What types of non-formal educational opportunities have these software designers sought or taken part in? How do these software designers seek and select these educational opportunities after they have joined the workforce? In what ways do they perceive these experiences have prepared them for their current role(s)?*
5. *What type of formal education do participants recommend for those planning to work in this field?*

## 1.2 Study Design

This study was conducted in three phases.

During Phase 1, interviews were conducted with software designers currently working in organizations which produce instructional or educational software. The primary purpose was to gain a rich understanding of the types of roles these software designers perform, the types of formal and informal educational opportunities they have pursued, and their perceptions of gaps in their own educational experiences. The semi-structured interview protocol was initially developed based on themes suggested by the literature and by an earlier study relating to the experiences of software designers working across a range of industries. As the interviews proceeded, questions were adapted to clarify them or to examine newly emerging themes. Phase 1 data was analyzed using the constant comparative method and was used to inform the questionnaire developed for use in Phase 2.

During Phase 2, software designers working in a variety of roles within organizations that produce instructional or educational software were invited to participate in a web-based survey questionnaire. The primary purpose of the data collected from this group was to determine the levels of competence software designers feel they had upon completion of their formal education. During the initial interview phase they were also asked to assess their current level of competence across a range of areas found to be important to software designers. Statistical analysis allowed me to ascertain whether or not there are gaps between what software designers learned during their formal education and the skills, knowledge, and attitudes they believe they need on the job. Demographic data obtained via the survey was also used to provide a picture of the typical profiles of software designers involved in the creation of instructional or educational software.

Phase 3 involved interviewing participants who represent profiles identified by the survey that were not represented during the initial interview phase via email. The third phase was also

intended to allow me to explore any open issues brought to light by analyzing the survey data and comparing Phase 1 and Phase 2 results.

A mixed-methods approach was used to analyze the data. More details are given on specific qualitative and quantitative techniques as well as the process of integrating the findings in Chapter 3: Methods.

## 1.3 Significance for Researchers

The results of this study may be of value to researchers interested in the training and education of designers. As mentioned earlier, this study is part of a larger research agenda related to software design practitioners' reflections on their own formal and non-formal educational experiences, the ways in which these educational experiences prepare them for their professional roles, and the implications these professionals' recommendations for formal educational program improvement may have for design education in general and software design education in particular. A comparison may be made to earlier studies conducted by myself and my colleagues, and possible subsequent studies on software designers working on other industries in order to gain an understanding as to whether software designers feel a need for more domain-specific formal education, or whether general software design education is more useful to them and can be supplemented as necessary by non-formal post-graduate education. These studies can also be seen as part of an effort across the design education community to understand ways in which various types of educational experiences prepare professionals for work in a variety of design fields.

It would be interesting to discover whether interactions with non-formal learning experiences (such as self-study, experimentation, and the use of various types of resources) are similar across design fields. This may have implications for the way designers of all types are

prepared at the university level. Specifically, this research agenda may lead to suggestions on the relative impacts of domain-specific and generalized design education at both the formal and non-formal levels. Another interesting strand is the impact of formal educational experiences on professionals' ability to pursue self-learning through non-formal education post-graduation to meet their job- and career- related needs.

## 1.4 Significance for Practitioners, Educators and Program Administrators

The primary groups who might be interested in the results of this study are educators and administrators in Software Design related programs (such as Computer Science, Software Engineering, and Human-Computer Interaction Design) and Instructional Design related programs (such as Instructional Systems Technology or Educational Technology). This study will in part address how university programs prepare software design professionals for their careers in instructional/educational technology development. Based on a review of literature (see Chapter 2), it is unclear whether there is a standard educational path for preparing students interested in designing educational or instructional software. This study may indicate ways that formal (university) programs and continuing education could be improved to support the development of these professionals. As stated earlier, this study is one in a series of studies. Looking across these studies will allow me to explore whether professionals may benefit from more domain-specific education, more generalized software design education (which appears to be the focus of current initiatives within the software design field), or general, cross-disciplinary design education. Findings across related studies may inform faculty and administrators in determining the direction of new programs, or providing focus to new programs.

Outcomes of this study may also inform instructional designers who manage, train, or interact with software designers. Outcomes of this study may indicate the value of learning to

recognize and understand the differences between software designers who have had different educational backgrounds and experiences. This would assist managers in identifying individuals with relevant skills for a given project or organization. It may also point out ways to provide an environment and resources which foster software designers' ability to augment or update their own skills. Understanding the background of software designers may improve communication between instructional designers and software designers.

Finally, the findings from this study and related studies may highlight the role that on-the-job learning plays in design fields. Administrators planning any type of design education program (including those preparing instructional designer practitioners and software design professionals) may be interested in exploring the implications these practices have on preparatory programs.

# 2 Chapter 2: Review of the Literature

## 2.1 Design Fields

Nelson and Stolterman describe design as the creation of new things, and state that the core of design activity is "to come up with an idea and to give form, structure and function to that idea" (Nelson & Stolterman, 2002, p. 1). They include many fields in their definition of "design", including architecture and graphic design but also information design, social systems design, educational systems design, and software design, and argue that a design culture which may cut across all of these fields has its own foundations or core concepts which are as relevant to it as "first principles" are to science (p. 3-4). Other authors coming from a range of fields including architecture, engineering, and instructional design agree that design fields share a common aim of creative endeavors focused on the needs of or carried out in service to a client, and constrained by characteristics of the real world, in contrast to arts or sciences, which have different foci and different constraints (Boling & Smith, 2007; Gibbons, 2000; Lawson, 1997; Rowe, 1991). Design is seen as a complex endeavor, in which designers approach each new problem as a unique case with no one optimal solution (Cross, 2001; Petroski, 1992; Rowe, 1991; Vincenti, 1990). Some skills and knowledge are seen as generic across all design practice, while others are specific to certain fields of design, although we may not yet know where these boundaries lie (Lawson, 1997; Rowe, 1991). Common characteristics across design fields may include: the focus on man-made (or "artificial") objects (Cross, 2001; Gibbons, 2000; Nelson & Stolterman, 2002); the intuitive nature of design processes (Cross, 2004; Nelson & Stolterman, 2002); thought processes typical of all designers (Lawson, 1997, 2004; Rowe, 1991); specific techniques used by designers (such as the use of artifacts as precedent materials and the use of

schemata and gambits to organize common patterns between ideas and solutions) (Boling & Smith, 2007; Lawson, 2004); and common categories or types of knowledge (although the specific knowledge may differ between design fields) (Gibbons, 2000; Vincenti, 1990). Although the authors cited do not all agree on all of these aspects of design endeavors, there are some common themes in thought about design - that design problems are complex; that there is no one correct answer to design problems; and that both knowledge and creativity play a role in design thinking. All strive to describe a "design culture" and language which allows designers to communicate about those things which all designers have in common.

### *2.1.1 Education for Design Professionals*

If design fields have common characteristics, it stands to reason that design educators have something to learn from one another. The majority of the authors cited in the previous passage explicitly or implicitly state that, because design is not the same as science or art, the traditional methods for teaching sciences and the arts may not be ideal for teaching design.

In his 1987 presentation on "Educating the Reflective Practitioner" Schön lamented the application of the "epistemology built into the university" which insists that theoretical knowledge (or "school knowledge") is the "highest" form of knowledge, and that professional knowledge be relegated to focusing on the application of research (Schon, 1987, pp. 1-3). He posited that this type of thinking about learning produces a regimented teaching style which focuses first on theory, with an underlying assumption that practice is "a confounding environment in which to experiment", resulting in a teaching style which focuses first on basic science, then on applied science, and only much later on "practica" which allow students to apply what they have learned to real problems (p.7-8). He recommended instead that education for those in the professions (including design fields) focus on "reflection-in-action" in which

students learn by doing in a "virtual world" which is a realistic but safe environment which allows for experimentation by students and teachers (who take on the role of coach, guiding students by observing and commenting on their "experiments" or teaching through demonstrations). He describes situations in which this type of education is used in design fields (such as architecture) and points out that it is often an uncomfortable technique in which students learn through experience and "try to educate themselves before they know what it is they're trying to learn", often becoming frustrated, out of control and even incompetent (p. 12), an observation that is reflected in Shulman's work on "Signature Pedagogies" used in educational programs for professionals.

Shulman (2005) discusses the difference between educational cultures in different fields. He stresses that in professions (which include design fields), education must take into account the standards of the professions themselves as well as the standards of the "academy". The signature pedagogies which develop in each field define what counts as "knowledge" in that field, how things are learned, and how knowledge is analyzed, criticized, accepted, and discarded. They are pervasive and even routinized part of the education culture, and cut across topics and courses, making it easier for professionals to learn highly complex subject matter within an increasingly familiar framework throughout their period of study. Unfortunately, Schulman explains that it is difficult to learn about the signature pedagogies used in professional education because "once they are learned and internalized, we don't have to think about them" (p. 56). However, he warns that there is a danger in unwittingly perpetuating the habits of signature pedagogies, as each will distort learning by focusing it to certain techniques or approaches. Therefore, he suggests that members of every profession examine the signature pedagogies used in other professions and ask themselves whether adopting some of these

approaches may improve their own teaching and learning cultures. If design fields have much in common, Schulman's advice seems particularly relevant in suggesting that design fields may have much to learn from one another about teaching, and from practitioners across design fields about the limitations of their current teaching techniques.

In his book "The Design of Design", Brooks (2010), whose own experience is in the field of Computer Science, laments that practices which are considered central to other types of engineering education programs are rare in Computer Science. He considers reliance on lectures and readings rather than critiqued practice a weakness in typical formal educational programs, and notes that the "best modern engineering education" includes critique of student work immediately in the Freshman year, concurrently providing science education (p. 245). He further points out that "strong engineering curricula often include 'co-op' or 'sandwich' programs, in which students intersperse on-the-job practice (and company training) between initial and final academic education" (p. 245).

### *2.1.2 Expertise in Design Fields*

Although most people have an idea what is meant by the term "expert", creating an actual definition for the term "expertise" is difficult to do (Kuchinke, 1997). The term "expertise" is often described in terms of behaviors or "behavior potential" of experts. Expertise cannot be developed solely through learning specific skills, knowledge, or heuristics. Nor can it be assumed that experts will always perform more effectively or efficiently than novices, as factors such as organizational restructuring can negatively impact an expert's ability to function. An individual with expertise "is typically seen as highly skilled and knowledgeable in some specific area, is presumably dedicated to keeping up-to-date through practice and continued learning, and has a high level of commitment to the area or domain of expertise" (p. 73). The degree to which

expertise is domain- and context-specific and the mechanisms through which expertise is developed are questions of discussion among researchers in this area (Kuchinke, 1997, p. 84)

Cross (2004) reviewed studies on expertise in design fields and compared them to literature on expertise in other areas. He found that experts in design fields differ from experts in other fields in a number of ways. Expert designers are "ill-behaved" problem solvers, and will tend to address even simple problems as if they were complex and do not have obvious solutions. They are focused on solutions rather than problems. Within familiar domains, they will re-frame problems pro-actively in ways which will help them efficiently find and structure appropriate solutions. Because such design situations are complex as well as ill-formed, designers are often required to make judgments based not only on the available information but also based on their own insights and previous experiences (Korkmaz, 2011). This type of "design judgment" is difficult to teach, but an essential part of design work.

Design experts differ from novices in additional appreciable ways (Cross, 2004). Clearly they have been exposed to a larger number of problems and solutions they can use as examples in their work, but their way of working with these examples also differs from the strategies used by novices. Experts can stand back from the specifics of an example to recognize underlying abstract principles which can be applied to future work. They are also able to access information in larger chunks and move quickly to more "generative" reasoning. They are much more aware than novices of the cognitive cost of strategies, and deviate accordingly from structured plans or processes. Unlike novices, who tend to use depth-first reasoning, experts use a mixture of depth-first and breadth-first reasoning, and switch rapidly between different aspects of a task. He also recognizes that a small group of "outstanding experts" exceed the level attained by others. These outstanding experts are able to work along "parallel lines of thought" to generate a range of

solutions, or focus on an appropriate narrow range of solutions. They frequently refer to "first principles" and tend to explore problems spaces from a particular perspective which will allow them to frame the problem appropriately to stimulate and pre-structure their design solutions. Finally, for these experts, conflict between their own design goals and clients' criteria inspire especially creative solutions.

Cross recognizes that there are different degrees of expertise, and that a period of "practice and sustained involvement" (at least 10 years involvement) and "dedicated application to a chosen field" is necessary before one reaches the level of an accepted expert (Cross, 2004, p. 428). However, many of the studies reviewed focus on relatively inexperienced "experts" (often comparing final year students to entry level students who are considered the "novices") because of the difficulty and cost involved in gaining access to highly-regarded experts working in the field. Therefore, not as much is known about the intermediate stages of development of experts.

Lawson (2004) describes five stages which designers must pass through in order to gain design expertise. The first stage is the acquisition of domain-specific schemata (complex sets of ideas which form common ground within an area of practice). After this, designers begin to develop a pool of precedent (previous design solutions which are used as points of departure in designing aspects of a solution to a new design problem). At this point, Lawson believes that a designer may be considered a competent professional. However, as they continue to gain expertise, designers go through three additional stages. They identify guiding principles, which allow them to structure and filter precedent materials and experiences. Those who become known for being able to use these guiding principles to exceed within a specific domain may be considered "experts". These experts will continue to develop the ability to recognize situations with little analysis and, finally, develop a set of design gambits or "tricks" which can be used to

solve many different problems within their domain of expertise. Lawson calls experts who have reached this level "masters".

Clearly, a significant amount of time is required to become what Cross and Lawson consider "experts". These experts can continue to develop and potential become what Cross calls "outstanding experts" and Lawson calls "masters". This would seem to indicate that those individuals who become experts and continue to develop beyond the minimum requirements for domain-specific expertise continue to learn throughout their careers and would seem to be an important trait in successful designers. The following sections discuss the nature and impact of lifelong learning and self-learning techniques which are so important in the continued development of professionals of all types, and designers in particular.

## 2.2 Continued Learning

As was discussed in the section on expertise, designers continue to learn beyond their initial university experiences. There are a number of models that are used to look at this type of learning.

### *2.2.1 Lifelong Learning*

Traditionally, education has been viewed as a "period of preparation and training" which takes place within primary, secondary, and university settings, and is "followed by a period of action"(Lengrand, 1989, p. 6). Within this view, the aim of education is to provide students with all attributes they will need to fill their future life roles. Schooling is therefore aimed at "cram[ming] the pupils' heads with all kinds of facts" which will allow them to build a satisfactory amount of "accumulated capital"(Lengrand, 1989, p. 6). However, if lifelong learning is recognized as a normal part of human development, our view of education may be seen as an important component of each phase in a person's life. Early education can focus on

the skills necessary to acquire knowledge and communicate with others, rather than the acquisition of large numbers of specific facts, while universal education of adults can be seen as a normal process which is just as important as the education of children (Cropley, 1989; Lengrand, 1989).

Based on this view, ongoing adult education can and should take multiple forms, depending on the topic and need. Different terms have been used over time, but the types of adult education generally fall into three categories, as described by Radcliffe and Colletta(1989). *Formal Education* is used to describe hierarchically structured, graded education systems and includes primary and secondary school, universities, and technical and professional training programs. *Informal Education* includes the daily experiences with family and the community that help individuals acquire attitudes, values, skills, and knowledge. *Non-formal Education* describes any organized education which takes place outside of the established system of formal education. Non-formal education typically focuses on specific skills, directed at a particular clientele and set of learning objectives. This paper will focus primarily on the formal and non-formal educational experiences of software designers, and ways that formal education did or did not support them in their role as lifelong participants in non-formal education as well as their role as software designers. Although Radcliffe and Collette distinguish between informal and non-formal education, for the purposes of this report I will refer to any educational opportunity and the use of any instructional or educational materials, including other human beings and references to one's own prior knowledge, as "non-formal education".

There is no single model for instruction within non-formal education. Instruction may be content centered (specific to a body of knowledge identified by specialists), problem-focused (helping students to learn general problem solving skills as well as generating information useful

in solving every-day problems), focused on conscientization (a process by which the disadvantaged can become aware of their own innate power to change society), or focused on developing learners' creative and planning capabilities (helping them to become more effective decision makers and change agents) (Srinivasan, 1989). Although educational opportunities include face to face or other types of prepared instruction, "experiential learning", in which adults learn by doing, is common. Experiential learning may be offered institutionally, within experience-based training, guided or cooperative educational opportunities in which a faculty member or field supervisor designs and measures learning goals, or by offering credit for life experiences. However, experiential learning may be entirely self-directed. This type of learning is discussed in the next section.

### *2.2.2 Self-directed Learning*

Tough (1989) describes adult learning as an iceberg, with self-instruction as the unseen base and more formal, instructor-led instruction as the visible top. Self-directed learning projects are typically aimed at acquiring skills and knowledge directly applicable to an anticipated task. According to a number of studies across multiple countries in the 1970s and 1980s, approximately 80% of the adult population were "continuously engaged in a series of learning projects, of which only 20% are occurring in formal classes" and a typical adult pursued approximately 5 distinct learning projects per year (Tough, 1989). More up to date statistics are hard to find; Livingstone (2001) indicates that most North Americans spend on average 10 hours or more on "informal" learning per week but that most studies do not distinguish between instruction and self-teaching. A 2004-2005 survey of adults in the United States found that 83% of adults in professional/managerial professions pursued some form of 'informal' instruction, while 93% were enrolled in employer-supported, work-related courses or training. Work-related

self-directed learning was not addressed(O'Donnell, 2006). Although self-instruction by definition does not involve pre-planned or instructor-led educational experiences, it clearly forms a crucial component of lifelong education.

These projects are most commonly motivated by an anticipated application of the knowledge or skill being explored. Self-directed learning may be chosen for a number of reasons, including desire to learn at one's own pace, desire to use a flexible learning style that matches one's own preferences and to structure one's own project, and the absence of restrictions such as the time and location a course is offered and transportation issues. Although self-instruction by definition does not involve pre-planned or instructor-led educational experiences, it clearly forms a crucial component of lifelong education, and should be considered in the development of formal education. Students must acquire metacognative skills in order to monitor their own progress and sustain motivation while learning. Furthermore, learners' expectations about their own learning and the attributions they make about their own failures and successes and the degree of control learners have over their own learning environment will impact future motivation for self-directed learning (Driscoll, 2000)

### *2.2.3 Continuing Professional Education*

Continuing Professional Education (CPE) allows professionals to "advance from a previously established level of accomplishment to extend and amplify knowledge, sensitiveness, or skill", and can take many different forms (Houle, 1980, p. 77). Taking lifelong learning into account can result in stronger and more useful preparation for professionals, and eliminate the need to "cover [all] the ground" in formal educational programs (Houle, 1980, p. 85). For pre-professional education, this may be achieved through internships and projects during which CPE is a resource. For professionals already immersed in a real-world setting, the need for education

typically arises from work tasks, and CPE is most valuable when it is seen as having a reciprocal relationship with work (Knox, 2006; Mott, 2000). Cervero (2001) notes that most practitioners understand that "the problems they face are… 'not in the book'" or in the type of research-based lectures typically offered by trainers (p. 25). Practitioners frequently use information acquired through CPE in different ways than the program designers intended, but this is not necessarily detrimental. Experts have a greater understanding of their own learning process than do novices (Daley, 2000). Cervero (2001) recommends that this advantage be used to fuel a change in both the content and educational design of continuing educational opportunities by integrating continuing education more fully into individual and collective professional practice. Knox (2000) similarly stresses the advantage of professionals' ability to choose appropriate educational opportunities which are relevant to tasks they are already immersed in. He recommends that pre-professional educational institutions take this into account by encouraging learners to become more self-directed and by responding to the needs being expressed by currently practicing professionals.

#### *2.2.4 "Growing" Designers*

Brooks (Brooks, 2010) discusses the need to deliberately "grow" designers on the job. Designers "need a combination of continuing formal education[1] interspersed with actual hands-on practice that is guided and critiqued by a master designer" (p. 247). Training courses led by a good teacher can provide a balanced overview of a subject, which can help a designer quickly "retool". He suggests that employers can also assist in employees' development by protecting designers' time by minimizing administrative and other bureaucratic distractions, and by

[1] When Brooks uses the term "continuing formal education", he is referring to intensive short courses, not university degree programs or courses. What Brooks refers to as formal education is referred to elsewhere in this paper as training courses, which fall into the category of "nonformal education" from a lifelong-learning perspective.

providing varied work experiences (potentially including working "sabbaticals" from the organization, for example to learn how users actually interact with the software by working in the role of a user for a short period of time). Professionals can "grow" themselves as a designer by constantly sketching designs, seeking knowledgeable criticism from colleagues, and by studying exemplars and precedents.

"Great designers, even the most iconoclastic, rarely start from scratch – they build on the rich inheritance of their predecessors" (Brooks, 2010, p. 205). Exemplars – examples of other's work, including both successes and failures – are "safe" models for designers to draw from in their own work. Knowing exemplars of the craft (including weaknesses as well as strengths) is very important for two reasons: to avoid risk by reusing valuable components of designs, and by providing examples of good style which a designer can draw on in creating original designs. Studying others' designs can also force attention to detail, and help makes one's own thinking more explicit.

## 2.3 Software Design

A review of the literature did not reveal a common title for what we will be terming "Software Designers". The Association for Computing Machinery (ACM) provides separate curriculum guidelines for each of the following: Computer Science, Computer Engineering, Information Systems, Information Technology, and Software Engineering (http://www.acm.org/education/curricula-recommendations). Denning (2001) listed 15 different "IT Specialties", including Computer Science, Computer Engineering, Database Engineering, Human Computer Interaction, and Software Engineering. He also provides a list of 15 "IT-Intensive Disciplines" including E-commerce, Information Science and Multimedia Design, as

well as Instructional Design, although ACM defines "IT" as a distinct field, not an umbrella term.

A history of the field explains to some extent the variety as well as the confusion in terminology. From the 1960s to the 1990s, Computer Science (CS) departments emerged and began differentiating themselves from departments which had originally been involved in the development of computing hardware (scientists, engineers, and mathematician) (Denning, 2001). A current definition of Computer Science is offered by the Liberal Arts Computer Science Consortium, which states that "computer science is the study of algorithms and data structures with respect to their (1) formal properties; (2) linguistic realizations; (3) hardware realizations; and (4) applications" (LACS, 2007, p. 2), with a particular emphasis on the formal properties of data structures and algorithms. They go on to recommend that there may be value in adding "consideration of social and ethical implications" and "the study of what is and what is not possible in the context of algorithmic problem solving".

In the 1990s, courses and then entire programs developed under the name "Software Engineering" (SE). According to the Software Engineering Body of Knowledge (SWEBOK) standards, SE is defined as "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering of software;" as well as the study of those approaches (Abran, Bourque, Dupuis, & Moore, 2001, pp. 1-1). SE was originally intended to focus on the use of rigorous techniques to develop reliable, dependable software which could accommodate the need to create larger and more complex software systems (Denning, 2001). Software Engineers suggested that SE programs be split from the traditional CS departments, they believed focused on "programming as mathematical activity", with the thought that Software engineering is as different from computer

science as chemical engineering is from chemistry (Denning, 2001, p. 10). However, in practice software engineering is often taught in computer science departments, and both areas may be moving closer to one another.

In 2008, the debate over what the field should be called and the unintended consequences of choosing a name continued to frustrate computing professionals. Attempts at avoiding the use of the term "Computer Science" have not been successful in communicating the full breadth of the field as it is understood by practitioners. The term "programmer" is interpreted by many as being very narrow, and focused on coding, although "insiders" consider it a broad term including the "design, development, testing, debugging, documentation, maintenance of software, analysis, and complexity of algorithms" (Denning, 2008, p. 19). An attempt to broaden the field by including Information Technology (IT) under the umbrella of Computer Science was disappointing, as it increased enrollments in IT programs but not in core computer science programs. Denning stresses that a member of the field of Computer Science has much broader responsibilities and interests than are understood by the general public – or perspective students. He illustrates this by pointing out that the typical computer scientist has many different voices, including: programmer, user, computational thinker, engineer, and scientist. Unfortunately, this realization does not assist in the attempt to find a common name for those who work in this general field.

An analysis of job advertisements for positions in the U.S.A. on Dice.com and ComputerWorld magazine conducted in 2004 revealed that the terms "Programmer", "Software Developer", and "Software Engineer" were used to describe very similar job descriptions, all relating to the role the author referred to as "software development" (Surakka, 2004). The author began with an assumption that these three terms would frequently be used synonymously, but

found, as he expected, that "low level programming skills" (which the author evaluated by counting the percentage of advertisements including the terms "Assembler", "C", "C++", and "embedded") were statistically significantly more prevalent in job advertisements for "Software Engineers", although certainly not exclusive to this job title. A quick and informal glance at Dice.com on June 1, 2011, shows a similar pattern; the first 5 results in a search for the term "software developer" include the job titles "Software Engineer IV (Web App Developer)", "Software Developer (Mid-Level)", "Software Developer III", and "Software Developer". Searching on the names of specific technologies (such as "Java" or ".NET") leads to similar results. The term "programmer" may be out of vogue in a Job Title – a search for "programmer" tends to bring up titles relating to "program management", although the seventh result in a search for "Java" brought up the Job Title "Java Developer/Java Programmer". It is noteworthy that despite the conclusion of both experts and practitioners that knowledge of a specific language is not as important as more general software design and development skills, employers tend to list specific programming languages not only in job descriptions, but also in job titles.

### *2.3.1 Software Design Education*

A number of professional organizations have created standards aimed at providing guidance for the development of Computer Science and Software Engineering programs. Although schools are not required to follow any of these standards, these professional organizations are quite influential in Software Design related fields. The following section discusses some of these standards, and then gives a quick overview of the types of discussions Software Design educators are having on the topic of Software Design education, as evidenced by conference presentations and published papers.

#### *2.3.1.1 IEEE and ACM: Software Engineering Body of Knowledge and Curricular Guidelines*

In 1998, the CS division of IEEE (Institute of Electrical and Electronics Engineers) and the ACM (Association for Computing Machinery), the two major professional organizations in the computing field, established a joint task force to produce a set of curricular guidelines for undergraduate programs (Atlee, et al., 2006). Software Engineering standards were based on the Software Engineering Body of Knowledge (SWEBOK) compiled by the same coalition. Although this coalition later split (IEEE wanted to pursue professional certification of software engineers based on the SWEBOK standards, while ACM felt that the field was not yet mature enough for this step), the ACM went on to create a flexible set of curricular requirements which universities could use to develop their own SE program or enhance CS or related programs with SE concentrations (Atlee, et al., 2006).

The SWEBOK standards list the following primary knowledge areas within software engineering: software requirements; software design; software construction; software testing; software maintenance; software configuration management; software engineering management; software engineering process; software engineering tools and methods; and software quality. The Software Engineering Education Knowledge (SEEK) based on SWEBOK was intended to provide "the essential and desirable knowledge and skills that any software engineering program should try to include in its curriculum" (Atlee, et al., 2006, p. 13). These areas were designed by a committee and are, where possible, based on educational research (Atlee, et al., 2006).

SEEK recommends 10 knowledge areas from software engineering as well as the related disciplines of mathematics, computer science, engineering, and economics. Each of the ten areas

consists of a number of knowledge areas, and the guidelines recommend a minimum number of lecture hours to devote to each area. The ten areas include:

- *Computer Essentials,* which includes computer science foundations and more advanced techniques (this covers a very wide range of computer science-related concepts and skills), as well as "construction tools" such as development environments, graphical user interface builders, and unit testing tools
- *Mathematical and Engineering Fundamentals,* which includes mathematical topics which form a foundation for computer science; engineering foundations for software (including methods and techniques related to analyzing and developing hardware, and systems development and engineering design practices, and measurements and metrics); and "engineering economics for software" (including software lifecycle considerations, various mechanisms for generating system objectives such as participatory design and prototyping, and methods for evaluating and ensuring cost-effective solutions)
- *Professional practice,* which includes concepts relating to group dynamics, psychology, communication skills related to reading and writing of code and technical documentation, team and group communication and presentation skills, and topics such as accreditation, codes of ethics, the nature and role of professional societies and software engineering standards, and employment contracts
- *Software modeling and analysis,* which includes principles and techniques for creating and analyzing various types of software models, and generating documentation for every level of software design and with various documentation techniques and specification languages

- *Software design,* which includes design issues relating specifically to the software development lifecycle, function- and object-oriented design strategies, architectural design, and human computer interface design, as well as design patterns and other techniques for lower-level design, and gaining familiarity with various types of design tools and design evaluation techniques
- *Software verification and validation,* including a variety of formal techniques for performing system checks and ensuring that a program meets the expectations of stakeholders
- *Software Evolution,* including the process and other activities involved in software evolution
- *Software process concepts, and the implementation of the software development process*
- *Software quality,* including basic concepts and the culture of 'software quality', and related standards and processes
- *Software management* at various levels, including project planning, personnel and organization planning, project management, and software configuration management (The Joint Task Force on Computing Curricula, 2004)

The full list of knowledge areas is very extensive. However, the steering committee intended that the topics be focused on practical knowledge, and indicated that students need sufficient exposure to all of these topics to become aware of available resources and their responsibilities as professionals. Although hands-on exercises are stressed, the committee's guiding principles state that it is not necessary to have hands-on exercises with all design patterns, standards, and industrially relevant area covered in the sub-areas. They intend the curriculum to be flexible enough to allow schools to adapt it to their university's additional

requirements, and to create more specialized programs. To aid in this goal, the SEEK provides a number of sample four year curricula which allow for various foci, and a number of curricular guidelines to help universities developing their own curricula based on the standards. Some central guidelines include:

- Curricula should stress that SE is both a computing discipline and an engineering discipline
- Concepts, principles, and issues should be taught as recurring themes throughout the curriculum
- Concepts which require academic maturity should be taught later in the curriculum
- Students should learn some application domains
- The curriculum should include significant real-world experiences, including case studies, practical assignments, course projects, and experience in an actual work setting
- Curriculum designers combine the knowledge areas in order to provide an efficient and synergistic curriculum

(Atlee, et al., 2006)

#### *2.3.1.2 IEEE and ACM: Computer Science Curriculum 2008*

A similar body of knowledge was created to guide the development of Computer Science curricula in 2001. In 2008, this set of guidelines was updated in order to keep standards current, but also to address concerns that the needs of industry and other interested parties had not been adequately met in the 2001 standards (ACM and IEEE Computer Society, 2008). The standards are intended to help produce learning objectives which will help produce graduates with the following characteristics, capabilities, and skills listed in Table 1.

**Table 1:** ***Computer Science Curriculum Standards***

### *Graduate Characteristics*

- System-level perspective
- Appreciation of the interplay between theory and practice
- Familiarity with common themes and principles
- Significant project experience
- Attention to rigorous thinking
- Adaptability

### *Capabilities and skills (paraphrased for brevity):*

- Cognitive capabilities and skills relating to Computer Science
  - Knowledge and understanding of essential facts, concepts, principles, and theories
  - Modeling and design of computer-based systems in a way that demonstrates comprehension of tradeoffs involved in design choices
  - Identification of requirements for a problem and plan an appropriate solution.
  - Understanding the elements of computational thinking, as applied broadly to everyday life.
  - Critical evaluation and testing.
  - Appropriate use of methods and tools.
  - Recognize and be guided by social, professional, legal, and ethical concerns regarding professional responsibility
- Practical capabilities and skills relating to computer science
  - Design and implementation.
  - Evaluation
  - Information management
  - Human-computer interaction
  - Risk assessment
  - Effective deployment of tools for construction and documentation of software, particularly in regard to solving practical problems
  - Software reuse
  - Operation of computing environment and software systems
- Transferable skills
  - Communication
  - Teamwork
  - Numeracy
  - Self-management of one's own learning and development and time management
  - Professional development
  - Software reuse and open source issues.

(ACM and IEEE Computer Society, 2008)

#### *2.3.1.3 The International Federation for Information Processing: Standards for Professional Practice*

In 1998, the International Federation for Information Processing developed a set of Standards for Professional Practice. These standards were created in order to address the needs of information technology professionals who worked in multiple countries. The goal was to create an overarching framework which could be adapted by organizations within each country. These standards were later presented at a joint conference with the International Conference on Software Engineering, during which 350 Software Engineers attended a forum regarding the standards' relevant to SE.

The areas addressed in these standards include ethics of professional practice; established body of knowledge; education and training; professional experience; best practice and proven methodologies; and maintenance of competence. Interestingly, in addition to the "education and training" area which is intended to cover undergraduate education, these standards also address the need for continued lifelong learning. Supervised experience following graduation and activities practitioners undertake throughout their professional lives are stressed in the "professional experience" and "maintenance of competence" areas. These guidelines later fed into the IEEE-CS/ACM Computing Curricula discussed in the previous sections.

#### *2.3.1.4 Liberal Arts Computer Science Consortium*

The Liberal Arts Computer Science Consortium (LACS) developed its own standards for Computer Science programs at liberal arts institutions. The LACS explain that traditionally computer science programs have focused primarily on the formal properties of algorithms and data structures, with a slightly lower emphasis on languages, machine hardware, and applications. They note that within the liberal arts setting, considering cross-disciplinary

perspectives on problem solving, the application of theoretical results, breadth of study, and communication skills are also stressed (LACS, 2007). The LACS guidelines suggest that undergraduate computer science programs in liberal arts schools focus on the following goals:

- To enable understanding the capabilities, limitations, and ramifications (technical, ethical, and social) of computing, the state of the art, and current research and development in computer science and related areas;
- To develop an ability to understand and analyze end user needs, master the techniques of creating and applying algorithms and data structures, and analyze their viability, correctness, and efficiency of utilizing analytical methods and appropriate theoretical results;
- To become effective at working individually and in teams, building on the work of others, and to be able to communicate technical information with both experts and non-experts;
- To prepare for adapting to changes in hardware and/or software technologies, and new and changing application areas through a firm grasp of fundamental principles and to develop an appreciation of the need for life-long learning;
- To appreciate both the demands and range of opportunities of the computing profession and provide for and encourage creative contribution to the art.

(LACS, 2007, p. 3)

Students who have completed a curriculum following the LACS guidelines should be able to:

- Understand multiple views of problem solving (e.g., 2 or 3 of imperative, object-oriented, functional);

- Have experience applying theoretical results to solving practical problems;
- Be able to apply critical thinking and problem solving skills across disciplines;
- Have experience with at least one large, team-based project or research project;
- Understand non-scientific perspectives and have sufficient background to be able to communicate effectively with people with those perspectives.
- Recognize the importance of social and ethical issues in computing.

(LACS, 2007, p. 4)

#### *2.3.1.5 ABET Computing Accreditation Commission*

The ABET accreditation guidelines must be followed by any program which wishes to have ABET accreditation. The guidelines for "Computing" programs break down into several sets, including "Computer Science and Similarly Named Computing Programs" (ABET Computing Accreditation Commission, 2010, p. 7), "Information Systems and Similarly Named Computing Programs" (p. 8), and "Information Technology and Similarly Named Programs" (p. 8). The guidelines for Computer Science programs appear to be the most applicable to this study. The general guidelines listed in Table 2 must be met by all three types of program. Table 3 includes the guidelines specific to Computer Science.

**Table 2:** ***ABET Accreditation for Computing Programs: General Guidelines***

| *General Guidelines* |
|---|
| 1. An ability to apply knowledge of computing and mathematics appropriate to the discipline |
| 2. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution |
| 3. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs |
| 4. An ability to function effectively on teams to accomplish a common goal |
| 5. An understanding of professional, ethical, legal, security and social issues and responsibilities |
| 6. An ability to communicate effectively with a range of audiences |
| 7. An ability to analyze the local and global impact of computing on individuals, |

organizations, and society
8. Recognition of the need for and an ability to engage in continuing professional development
9. An ability to use current techniques, skills, and tools necessary for computing practice.

(ABET Computing Accreditation Commission, 2010, p. 3)

**Table 3: ABET Accreditation for Computing Programs: Computer Science Specific Traits**

*Characteristics of Graduates*

1. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling of design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
2. An ability to apply design and development principles in the construction of software systems of varying complexity.

*Cognitive capabilities and skills relating to computer science*

3. Computer science: One and one-third years that must include:
   a. Coverage of the fundamentals of algorithms, data structures, software design, concepts of programming languages and computer organization and architecture.
   b. An exposure to a variety of programming languages and systems.
   c. Proficiency in at least one higher-level language.
4. One year of science and mathematics:
   a. Mathematics: At least one half year that must include discrete mathematics. The additional mathematics might consist of courses in areas such as calculus, linear algebra, numerical methods, probability, statistics, number theory, geometry, or symbolic logic.
   b. Science: A science component that develops an understanding of the scientific method and provides students with an opportunity to experience this mode of inquiry in courses for science or engineering majors that provide some exposure to laboratory work.

(ABET Computing Accreditation Commission, 2010, p. 5)

In addition, ABET accredited programs are required to monitor student performance, publish educational objectives which are consistent with the institution's overall mission and ensure that student outcomes relating to these objectives are documented, continuously review the program curricula, and hire faculty with appropriate expertise and educational backgrounds, as well as maintaining adequate facilities and instructional support services (ABET Computing

Accreditation Commission, 2010). The Computer Science related guidelines indicate that at least some full time faculty members must have a Ph.D. in Computer Science.

These recommendations have changed in the last few years. For example, prior to 2009, the guidelines explicitly required differential and integral calculus, probability and statistics, and a full year of laboratory science. As Popyack (2010) points out, this may afford accredited programs the opportunity to, for instance, work with mathematics and science departments to design courses specifically adapted to the needs of the Computing students. Perhaps even more excitingly (at least from Popyack's point of view), this also opens the doors to incorporating arts into Computer Science programs (possibly creating a Bachelor of Arts in Computer Science), or creating interdisciplinary degrees.

#### *2.3.1.6 Ongoing Discussions of Software Design Educators*

Education of Software Designers is an important topic to the leading professional organizations in the field, as evidenced by the work done by the IEEE/ACM joint task force. The IEEE has annual conferences focused on each of the following areas: "Software Education and Training", "Computer Science and Education", "Information Technology Based higher Education and Training", "Digital Game and Intelligent Toy Enhanced Learning", "Frontiers in Education" and "Development and Learning". In the last several years special symposia and workshops on topics such as "International Workshop on Technology for Education in Developing Countries" and "International Symposium on IT in Medicine & Education". In 2006, the second annual conference was held on "Engineering Education, Instructional Technology, Assessment & E-Learning" ("IEEE Conferences and Meetings," 2008). IEEE members are also provided access to many resources for continuing education. Similarly, the ACM has two special interest groups dedicated specifically to education of software designers:

SIG CS Education, which "provides a forum for educators to discuss the problems concerned with the development, implementation, and/or evaluation of computing programs, curricula, and courses, as well as syllabi, laboratories, and other elements of teaching and pedagogy" (http://www.sigcse.org/, June 1, 2011), and SIG IT Education, which has a very similar mission relating to the area of Information Technology, and which has created a model undergraduate curriculum as well as helping to create accreditation guidelines for IT programs. Each of these organizations has yearly conventions and a very active membership. Two major ACM journals focus on Computing education: "The ACM Transactions on Computing Education" and Inroads, a magazine "intended for professionals interested in advancing computing education in the world". Both ACM and IEEE also provide access to a variety of educational resources for their members' own continuing education.

#### *2.3.1.7 Open topics for Software Design Educators*

The debate over the most critical areas to be covered in software design-related programs continues. As part of his critique of the current state of computing instruction, Andriole (Andriole & Roberts, 2008) compared the results of a survey of practitioners (including upper management as well as software developers themselves) to the ACM Task Force standards. Although there were areas of overlap, including design, integration, information and application architecture, optimization, and metrics, there were a number of areas that practitioners felt were important which were not addressed by the task force standards, including knowledge and skills related to business strategy and applications, technology infrastructure and support, and organization and management. Andriole posits that the need for programmers is decreasing while the need for business and "enterprise" level skills is increasing. Therefore, he recommends that programs decrease their traditional focus on programming languages, algorithms, and

datastructures, and focus more on software engineering best practices. In his counterpoint response, Roberts argued that traditional programming skills were still relevant and necessary, and that the demand for these skills is indeed increasing. He warned that the United States risked "abandoning the playing field" by failing to produce students with these necessary "traditional" software engineering skills.

Dewar and Schonberg (2008) made a similar argument, expressing their concern about "worrisome trends in Computer Science education". Trends which concerned them the most were the lowering of mathematics requirements in CS programs and replacement of programming skills in several languages with "cookbook approaches using large libraries and special-purpose packages". They believe that these trends are leading to a lack of skills required by today's software industry, and the creation of "easily replaceable professionals."

This article sparked an ongoing debate, in which Dewar posited that with appropriate training in "the fundamentals", a computer science professional should be able to create bug-free software. These fundamentals include instruction in formal specification, requirements engineering, systematic testing, formal proofs of correctness, structural modeling, and other areas, and indicated that these could best be taught with more traditional programming languages such as C++ (Dewar & Astrachan, 2009). In his "counterpoint" argument, Astrachan argued that rather than dumbing down computer science curricula, decisions such as the use of the Java programming language were made for good pedagogical reasons, "working to ensure that our beginning courses were grounded in the essence of software and algorithms" (Dewar & Astrachan, 2009, p. 45). Other educators discuss the value of spending time and energy on projects that may help students to learn higher-order thinking skills (Hauer & Daniels, 2008) and give students experience dealing with clients, struggling with communication and people skills,

developing timelines, learning about new programming languages or technologies on their own, and dealing with complex and stressful problems similar to those they would encounter in the "real world"(Ghassan Alkadi, 2010). A related discussion examines the benefits and drawbacks of teaching within a context versus more generalizable, decontextualized instruction, and in what situations which approach should be used (Cooper & Cunningham, 2010; Gudzia, 2010).

Some educators are also concerned about the impact of the choice of in-course activities on student retention. Astrachan argued that programs must "encourage passion" by looking for "problems that motivate the study of computing, problems that require computing in their solution" in order to attract students back to the field (Dewar & Astrachan, 2009, p. 45). Kumar (2010) similarly pointed out that certain languages and programming development environments may be especially attractive to students in introductory courses, because they encourage playfulness through the ability to quickly design and implement interesting and fun applications.

Although this series of arguments may seem esoteric at times, their impacts on university programs can drastically change the focus of the education students receive, and according to the authors, directly impact the types of jobs graduates will be prepared for. Walker (2010) warns of the tendency of both faculty and students to recommend adding every conceivably useful skill or topic to their computing degree program, thereby preparing students for any job they might apply for. Yet, as he points out in his "Eight Principles of an Undergraduate Curriculum", not every topic can be adequately covered, much less be absorbed by students, in a 4-year curriculum (Walker, 2010). Determining *which* skills should receive attention is the challenge for educators and administrators.

Lethbridge (2000) addressed the types of knowledge most important to software professionals through a survey instrument which asked participants to respond to each of 75

topics identified from university curricula and SWEBOK standards. Respondents were requested to indicate for each topic how much they had learned at school, how much they currently knew, and how useful each topic had been to them. The results of this survey allowed the researcher to determine which topics covered extensively at the university level tended to be forgotten by professionals as well as areas in which professionals had to augment their learning on the job. Lethbridge provided a list of the 25 most important topics as identified by professionals, as well as a list of the 25 least important topics. "Importance" was calculated by combining participants' responses on two questions which addressed the usefulness of specific details of the material on the job (from 0="completely useless" to 5="essential") and the amount of influence the material had on their thinking (approach to problems and general intellectual maturity, with 0="no influence at all", and 5="profound influence on almost everything I do") (p. 45). For each of the 25 most important topics, the "amount learned in education" not surprisingly ranked lower than "amount known now", indicating that participants had learned more since graduating than was acquired at school. However, there seemed to be larger discrepancies in some areas than others. For example, among topics that ranked on average higher than 3.0 on a 5.0 likert type scale in importance, the following items received a mean of less than 2.0 in amount learned in education (where 1="became vaguely familiar" and 2="learned the basics". In contrast, 3="became functional (moderate working knowledge)", 4="learned a lot", and 5="learned in depth; became expert"): "software design and patterns", "software architecture", "requirements gathering/analysis", "O-O concepts", "HCI/user interface", "Ethics and professionalism", "Analytics and design methods", Giving presentations", "Project management", "Testing, verification, QA", "Technical writing:, "Databases", "Leadership" and "Configuration/release management" (p. 49). The last two items

received less than a 1.0, which indicated that these topics had not even been mentioned within the bachelor's program, to participants' recollection. While "leadership" may arguably be a skill that is not immediately necessary in an entry-level position, "configuration/release management" is likely part of a project of any significant size. Interestingly, the list of bottom 25 items includes some that professionals likely believe were covered more than necessary in their formal education. Not only were these items of relatively low importance, but in comparing "amount learned in education" to "amount known now", Lethbridge identified a number of areas where participants apparently forgot much of what was known. These included "linear algebra and matrices", "Physics", "Graph theory", "Control theory", "Differential/integral calculus", "Combinatorics", "Laplace/Fourlier transformations", "Chemistry:, and "Differential equations". You may note that these topics all relate to mathematics and physical sciences. It is possible that although specific concepts were forgotten, they were still useful in some way. Since Lethbridge has combined scores relating to direct importance on the job and influence on thinking in general, it is not possible to determine whether some of these items may have had a more indirect impact. However, except for "Linear Algebra" (which scored just barely above a 2.0), the items mentioned all had a mean score of under 2.0, indicating very little importance overall. These findings may indicate some areas where curricular designers could consider whether the balance of topics in degree programs is correctly adjusted for the needs of graduates.

Of course, looking at professionals' self-reporting of important topics is just one piece of the puzzle that faculty and students are concerned about. As part of his analysis of job advertisements, Surakka (2004) compared topics required in job advertisements to the degree to which related courses are required in ABET/CAC accredited programs (based on an earlier study by McCauley and Manaris), as well as comparing them to the importance given to each topic in

Lethbridge's (2000) survey of computing professionals. The most commonly requested skill-set in job advertisements, "Database Management Systems", was required by only 31% of accredited programs, although it also ranked among the highest on the items listed in Lethbridge's study (with a mean of 3.3 on a 6-point likert scale ranked 0-5, with items of greater importance ranking higher). Courses that were nearly always included in accredited programs were not necessarily a good match with the job advertisements, or with professionals' appraisals of the importance of these topics. For example, 96% of accredited programs required a course on Operating Systems, which was only "sometimes" mentioned by job advertisements (that is, this topic was mentioned in 2-19% of advertisements), although it also received a relatively high score of 3.3 on the scale provided by professionals. "Programming Languages", which was required in 87% of programs, was "hardly ever" required by job advertisements (that is, this was mentioned in less than 1% of advertisements). The next three topics, "Software Engineering" (required by 76% of programs), "Architecture" (69% of programs), and "Analysis of Algorithms" (67% of programs) were all included only "sometimes" in job advertisements. Of the three items that received the highest ranking by professionals (3.3), only one was included very frequently by programs (Operating Systems, included in 96% of programs). The others were Database Management Systems, which, as mentioned earlier, was required by only 31% of programs, and Human-computer Interaction, which was required in a mere 4% of programs. In contrast, Surakka's comparison of popular programming languages was a better match as the "first programming language" taught in academic programs in the 2001-2002 school year (which would have prepared graduates entering the workforce in 2004) was Java (in 49% of programs), C++ (in 40% of programs), and C (in 11% of programs), which matched well with employers' most frequently requested programing languages. The author cautions that reviewing job

advertisements cannot necessarily identify all skills which are *not* important on the job as some information may be missing. However, the implication is that skills that *are* included in job advertisements must be important on the job[2]. It is noteworthy that communication and team-skills were missing from Surakka's comparison table. A quick glance at Dice.com shows that these skills are requested in current advertisements. It is not clear whether Surraka was simply uninterested in these skills, or whether they were not commonly requested in 2004. Lethbridge (Lethbridge, 2000) includes topics such as psychology, philosophy, ethics and professionalism (which ranked in the top quartile in "importance"), technical writing (in the top quartile), and "people skills" including giving presentations to an audience" (top quartile), "leadership" (top quartile), and negotiation, indicating that these skills were valued by professionals in 2000.

When reviewing the Computer Science Curriculum Standards, the joint IEEE Computer Society and ACM society task force found that a major concern brought up by industry leaders was that graduates "[have] been indoctrinated in particular tools or processes that they then have to unlearn" (ACM and IEEE Computer Society, 2008, p. 11). The industry leaders recommended instead that graduates need to have an appreciation for why particular topics are important along with guidelines which will "help them think about the craft of modern software development" (p. 12). As one industry representative was quoted as saying in the report, "The thing we [as employers] can't afford to do… is teach candidates how to think critically, be effective problem solvers, and to have basic mastery of programming languages, data structures, algorithms, concurrency, networking, computer architecture, and discrete math/probability/statistics." Among the other specific issues found especially important by industry are: security issues,

[2] The specific skills as well as programing languages mentioned here reflect those that were important in 2004 and might not be directly relevant in 2011 – although other literature would suggestion that the "course topics" listed continue to be relevant both in education and in practice, if the relative importance of specific programing languages may have shifted.

quality issues (including testing, debugging, bug tracking, concerns about code readability and documentation, and the importance of code reviews), Software Engineering principles and techniques (including basic release management and source control principles and best practices for developing software in teams), code archeology (the necessity to be prepared to delve into large, poorly documented code bases and make sense of them), and back-of-the-envelope type performance tuning. The committee also addressed the need to draw increased attention to "teaching of basic programing", and a number or specific areas were modified to focus more on architecture and planning related concerns. The need for students to be exposed to different programming paradigms across the course of their program was also taken into account. Finally, the committee made changes to address the concerns of international competitiveness, legal, social, and cultural issues related to designing software meant to be used internationally.

### *2.3.2 Continuing Education of Software Designers*

Because technology changes so rapidly, the need for ongoing education beyond the period of formal education was recognized since the early days of computer science. In 1978, Fischer, Alvarez and Taylor surveyed practicing programmers to determine how they kept up to date, and made suggestions on the implications of this study to computer science education (Fisher, Alvarez, & Taylor, 1978). They found that the majority of programmers kept up to date by talking with colleagues, "shop standards", and books and manuals. They found that programmers tended to be more up-to-date than non-programmers, and suggested that managers ensure that they themselves were up-to-date and that they leverage programmer's inclinations to learn from one another by hiring new programmers with up-to-date skills, keep shop-standards and an in-house library of published books up to date.

Lethbridge's (2000) survey of in-practice professionals provides evidence of on-the-job learning by comparing self-reported levels of "current knowledge" (on a scale from 0="know nothing" to 5="know in depth/am expert (know almost everything)" to the amount learned in formal education (from "learned nothing at all" to "learned in depth; became expert (learned almost everything)") (p. 45). Although this can be seen as a troublesome gap between on-the-job needs and formal educational programs, it also indicates that professionals *do* continue their own education once on the job – not only on specific technical skills, but also in high-level, theoretical areas such as those mentioned earlier. The top four topics listed in "learned on the job" (calculated as current knowledge – learned in education) are "Testing, verification, and quality assurance", "maintenance, reengineering, and reverse-engineering", "project management", and "configuration and release management" (p. 46).

As was discussed earlier, Brooks (2010) recommends formal training, skilled mentorship, and broad use of "exemplars" or precedent materials for the purpose of on-the-job training. Exter and Turnage (2011) found that non-formal learning is considered a natural part of a computing professional's role, and that experienced professionals regularly engage in a number of activities ranging from reading books and online resources to learning from peers and mentors and attending structured training courses. Experimenting with and learning from samples of others' code or designs is considered a common and useful technique to learn from as well as to progress in the design process. In a related study, Turnage and Exter (in preparation) found that software design professionals use their own previous experiences as well as designs of others to further their own design thinking. This "precedent use" is similar to Brook's discussion of the use of "exemplars".

University programs need to provide a foundation for students to succeed in designing and developing increasingly complex systems, and faculty members and the field at large must continue to pursue ways to improve educational programs to fill this need. However, as was pointed out in Fisher et al's 1978 study, in a field which focuses on rapidly changing technologies, keeping up to date will always be important. Understanding how and what software designers learn after they have left university may help instructors and designers of university programs prepare students for the important role self-instruction will play in their future careers. As Walker (2010) points out, because the computing field changes so rapidly, it is not possible to anticipate future innovations and technologies that students may encounter on the job, much less teach them the specific skills. Therefore, "[students] must be able to learn on their own, and they should be able to place new ideas within a framework of solid principles. A program in computing should provide this foundation" (p. 21).

## 2.4 Instructional Design

### *2.4.1 Roles played by Instructional Designers*

#### *2.4.1.1 IBSTPI Standards*

In 1977, a Joint Taskforce was created by the Association for Educational Communications and Technology (ACET) and the National Society for Performance and Instruction (NSPI, which later became the International Society for Performance Improvement, ISPI). This Joint Taskforce developed a list of competencies for instructional design professionals. The Taskforce was later reorganized because of issues relating to conflicts of interest, and is now the International Board of Standards for Training, Performance and Instruction (IBSTPI). The 2000 version of the Instructional Design Competencies include 23 "domains" organized under four main areas; Professional Foundations, Planning and Analysis,

Design and Development, and Implementation and Management. Each domain includes a set of skills, some of which are "essential" while others are considered "advanced" and may not be covered in all programs. The 23 domains are listed in Table 4.

**Table 4:** ***Instructional Design Competencies: The Standards***

### Professional Foundations

1. Communicate effectively in visual, oral, and written form.
2. Apply current research and theory to the practice of instructional design. (NOTE: this entire domain is considered "advanced")
3. Update and improve one's knowledge, skills, and attitudes pertaining to instructional design and related fields.
4. Apply fundamental research skills to instructional design projects (NOTE: this entire domain is considered "advanced")
5. Identify and resolve ethical and legal implications of design in the workplace (NOTE: this entire domain is considered "advanced")

### *Planning and Analysis*

6. Conduct a needs assessment.
7. Design a curriculum or program.
8. Select and use a variety of techniques for determining instructional content.
9. Identify and describe target population characteristics.
10. Analyze the characteristics of the environment.
11. Analyze the characteristics of existing and emerging technologies and their use in an instructional environment.
12. Reflect upon the elements of a situation before finalizing design solutions and strategies.

### *Design and Development*

13. Select, modify, or create a design and development model appropriate for a given project (NOTE: this entire domain is considered "advanced")
14. Select and use a variety of techniques to define and sequence the instructional content and strategies.
15. Select or modify existing instructional materials.
16. Develop instructional materials.
17. Design instruction that reflects an understanding of the diversity of learners and groups of learners.
18. Evaluate and assess instruction and its impact.

### *Implementation and management*

19. Plan and manage instructional design projects. (NOTE: this entire domain is considered "advanced")
20. Promote collaboration, partnerships and relationships among the participants in a design project (NOTE: this entire domain is considered "advanced")

21. Apply business skills to managing instructional design (NOTE: this entire domain is considered "advanced")
22. Design instructional management systems (NOTE: this entire domain is considered "advanced")
23. Provide for the effective implementation of instructional products and programs.

(Richey, et al., 2001)

#### *2.4.1.2 ISTE and NCATE accreditation guidelines*

The International Society for Technology in Education produces standards related to secondary computer science education, technology facilitation, and technology leadership. None of these appears to relate directly to instructional design education, and the computer science education standards relate specifically to teaching computer science at the secondary level. The standards appear to be aimed at teachers, technology support personnel, and administrators.

As of March, 2011, AECT notified NCATE that it would no longer be maintaining standards for Educational Technology and Media Support Specialist-related programs. New programs will not be accepted by NCATE beginning in Fall, 2011 (http://www.ncate.org/Standards/ProgramStandardsandReportForms/tabid/676/Default.aspx). Although these standards will be briefly touched on in following sections because they have influenced programs up to 2011, they will not be discussed here in detail.

### *2.4.2 Open topics in Instructional Design Education*

Studies of how professional instructional designers spend their time indicate that they play many roles beyond those recognized in traditional Instructional Design models. In a survey of in-practice instructional designers, Cox and Osguthorpe (2003) found that, on average, instructional designers spend only about 23% of their time on "original design", with the next 22% spent on project management and administrative responsibilities, although the amount of time spent on various activities varied quite a bit depending on the specific job title held. A

review of literature by Kenny, Zhang, Schwier, and Campbell (2005) identified the following additional roles: communications, editing and proofreading, marketing, media development and graphic design, project management, research, supervision, training students/faculty development, team building/collaboration, and technology knowledge/programming. Similarly, a review of job advertisements paired with a survey of FSU program alumni found that job requirements for instructional design related positions frequently included the following skills: communications and collaboration; project management; business management; technology, e-learning and programming; adult learning theory; practice experience in specific ID skills (e.g. Needs assessment); online productivity software; measurement, research, evaluation and analysis; and other practical areas such as consulting skills, change management, and negotiating with subject matter experts and clients (Hanna, Yap, Fong, Fletcher, & Bancroft).

However, graduates may not be prepared for all of these roles. A survey of practitioners working in a range of environments indicated that those working in ID had a variety of backgrounds, including many with only bachelor's degrees and others who were at least initially self-trained in the field of instructional design, many of whom later returned for a degree in this area (Larson, 2005). When asked about their own degree experiences, participants indicated that programs attended varied significantly in specific courses and experiences offered. Most programs offered similar coverage of general ID competencies (as identified by professional organizations), ID models used in the respondents' own practice, learning theories, and "flexible design of learning theories, instructional strategies, and instructional models" (p. 28). However, programs differed on the degree to which competencies related to specific types of work environments and subject matter specific to their workplace was covered, with some programs offering "specific environment" programs, while other programs were more generalist. Alumni

of ID programs generally indicated they were not well prepared for the "cultural aspects" of the job, especially in terms of working with supervisors, workplace politics, use of resources, and the expected workload.

Participants in Larson's (2005) study were asked to identify an exemplary program. A related study (Larson & Lockee, 2007) looked at the most frequently mentioned program, and found that the skills emphasized were business competence (including the "ability to think, write, and communicate orally" and the "ability to make wise use of instructional design and technologies" (Larson & Lockee, 2007, p. 3)), communications, interpersonal relationships, analytic competence (including areas such as critical thinking and "problem definition and problem solving" (p. 3)), project management, business skills, and technological literacy competence (including knowledge of recent technology, evaluation of new and existing technology, and online teaching/designing and distance education skills). Based on this study, Larson and Lockee point out the value of contacting alumni, employers, and practitioners "to identify the job demands of a corporate environment and to develop educational practices that align with those demands" (p. 21). Interestingly, their findings from this case study and related literature point not to a list of specific skills but to the value of "preparation practices such as case studies, authentic project work, internships and assistantships, action learning principles, and situations designed to facilitate cognitive apprenticeships"(p. 20), which will allow students to engage in solving complex, ill-formed problems and to develop interpersonal communication and team-work skills as well as gaining an understanding of the cultural aspects of a specific type of work environment. Based on their review of earlier studies, Kenny et al (2005) similarly point out that a focus on the core skills associated with ID models is not sufficient to prepare students for work in this field, as these models are often not used in their entirety in practice, while many

other roles are played by in-practice instructional designers. They recommend instead looking further into what it "means" to be an instructional designer, including the ability to "make judgments about design situations that are complex, rich, and replete with tensions and contradictions" (p. 8).

Newer versions of the standards appear to attempt to address at least some of these areas. For example, the IBSTPI standards aim to "cover the whole design process and the different roles that instructional designers may assume", although "advanced" standards may not be met by all instructional designers, as some may focus on specific areas (Spector, 2006, p. 6). As discussed earlier, the standards cover four main areas: Professional Foundations (including communication and research skills), Planning and Analysis, Design and Development, and Implementation and Management (Richey, et al., 2001). There is a recognition that new areas of design may emerge, and that competencies should be generic enough to be customized to meet the needs of specific organizations (Spector, 2006). The AECT NCATE accreditation guidelines for Educational Technology and Media Support Specialist-related programs ("Standards for the accreditation of school media specailist and educational technology specialist programs," 2005) appear to be much more tied to the traditional Instructional Systems Design model, but do include a set of "management" related competencies, as well as specific competencies relating to print, audiovisual, computer-based, and "integrated" (hypermedia) technologies.

### *2.4.3 Software Design Roles for Instructional Designers*

In an exploration of key literature in the field of Instructional Design, "including official definitions, published professional competencies, and popular instructional design textbooks" (p. 33), Smith (2008) attempted to uncover the meaning of "design" within the field of instructional technology. The common themes she found addressed instructional technology's focus on

problem solving, theory, and process. Instructional technology's grounding in data appears to focus on humans and human systems. Smith explains that, according to the official sources, instructional design is "characterized by subdivision and partitions", allowing the design process to be divided among one or more specialists (p. 133). These activities include needs analysis, instructional analysis, learner analysis, writing performance objectives, developing assessment instruments and instructional strategies, developing or selecting instructional materials, and the evaluation of instruction. None of these activities appear to be focused specifically on the types of tasks performed by software designers. The area which may align the most with the activities of software designers is referred to in instructional technology literature as "development", which involves moving from specifications created in the "design" phase to specific items, including test versions, prototypes, or mass-produced versions of physical items to be used in instruction. Although these activities may conceivably overlap with the responsibilities of a software designer, nothing in this description indicates that instructional technologists will be specially prepared for a software design role.

The IBSTPI standards also do little to address the type of design and development central to the roles played by participants in this study. The competency that seems most relevant in the IBSTPI standards is: "Analyze the characteristics of existing and emerging technologies and their use in an instructional environment." The related performance components are: "specify the capabilities of existing and emerging technologies to enhance motivation, visualization, interaction, simulation, and individualization", "evaluate the capability of a given infrastructure to support selected technologies", and "assess the benefits of existing and emerging technologies" (Richey, et al., 2001, p. 70). The "design and development" competencies would appear to be the most relevant to the topic of this paper. These are written to be technology-

neutral, although computing professionals would likely find themselves involved in selecting or modifying existing materials, developing new materials, understanding end users ("learners" in the context of instructional design standards), and evaluating and assessing their end-product.

The IBSTPI standards acknowledge specialization among instructional designers. "The E-Learning Specialist" would probably be the specialty that most aligns with the purpose of this study (Richey, et al., 2001). As described in the standards, an E-Learning Specialist would need to be "familiar with a wide range of established and emerging technologies, their advantages and drawbacks, and their effect on learner motivation and the learning process" (p. 127), and must be able to assess new technologies to ensure they meet a project's needs. An E-Learning Specialist must also have "expertise in all facets of the design and development of technology-based learning" (p 125). However, these "facets" do not appear to extend to the skills common to software designers. Rather, they include "the use of color, interactivity, screen layout and motivating graphics" as well as the ability to "[reduce] technical content to clear and unambiguous format for various delivery formats". Probably of most interest for the purposes of this study, E-Learning specialists must be able to communicate between the design team and non-technical specialists and management. This would seem to imply that E-Learning specialists may be prepared to work closely with computing professionals, but are probably not expected (or prepared) to produce complex software themselves or to provide the type of design that a computing professional would.

The AECT's NCATE ("Standards for the accreditation of school media specailist and educational technology specialist programs," 2005) include under "Development" a sub-area specific to Computer-Based Technologies, which are defined as "electronically stored information in the form of digital data", including "computer-based instruction (CBI), computer-

assisted instruction (CAI), computer-managed instruction (CMI), telecommunications, electronic communications, and global resource/reference areas". Students in an NCATE-certified program sound be able to design and produce "audio/video instructional materials" and "digital information" with computer-based technologies, use digital cameras, video cameras, and scanners to produce instructional materials, and (for those preparing to become a school media specialist), "incorporate the use of Internet, online catalogs and electronic databases to meet the reference and learning needs of students and teachers". Another "Development" area, "Integrated Technologies", includes "hypermedia environments which allow for: (a) various levels of learner control, (b) high levels of interactivity, and (c) the creation of integrated audio, video, and graphic environments." Examples given are "hypermedia authoring and telecommunications tools such as electronic mail and the World Wide Web". Related competencies include using authoring tools to create hypermedia and multimedia materials; developing materials for distance education, combining electronic and non-electronic materials; using tools such as email and web-browsers; developing "effective Web pages with appropriate links using various technological tools (e.g. print technologies, imaging technologies, and video"); using writable CD-ROMs ; and using software to capture on-line materials for off-line presentations.

None of these standards would seem sufficient for preparing students for the types of roles participants in this study play. However, a review of instructional-design-related job advertisements found that programming skills were sometimes asked for. In addition to HTML, CSS, and related markup languages, requirements for "programming skills with Java, Java Script and Ruby" appeared in 31 of 258 ads analyzed (Hanna, et al., p. 15). However, Hanna et al also point out that 25 of the advertisements they reviewed "listed a degree in computer science as an alternative to a degree in instructional systems" (p. 15). So, perhaps one might expect Software

Designers of educational software to have a background in Computer Science or other Computing related educational programs.

## 2.5 Professional Interest in Educational Software

A quick online search reveals a huge amount of interest in educational software among a number of professional groups. For example, there are multiple groups dedicated to topics relating to the use of technology in education, such as the *Association for Educational Computing and Technology* (http://www.aect.org/default.asp ), *the International Society for Technology in Education* (http://www.iste.org/welcome.aspx ) and an online social network group, *Technology Integration in Education* (e.g. http://www.technologyintegrationineducation.com/ . Many groups focus specifically on e-Learning, for example, *eLearning Network* (http://www.elearningnetwork.org/). The Institute of Electrical and Electronics Engineer (IEEE)'s Computer Society includes a "Technical Committee on Learning Technology", which publishes a newsletter, conducts workshops and conferences, and provides a forum for discussion of related technical topics (http://www.ieeetclt.org). Of these groups, only the IEEE's TCLT mission appears to be primarily related to the design and development of educational software from a Computing perspective; the while other groups focus on the instructional design aspects of designing the software, or the use of that software in educational settings.

Recruitment for this study revealed that many software design professionals identify with this field, including those who came from a formal background in Software Design, Instructional Design, or other areas entirely. Chapter 4: Findings will provide details on the roles these professionals play and what they feel is unique about working on educational software projects, as well as exploring their formal educational backgrounds and on-the-job needs.

## 2.6 Conclusion

Expertise in design fields accumulates over time, beginning during the period of formal education but continuing once professionals graduate and begin their careers. Lifelong learning and self-learning strategies are important to professionals in all fields. They appear to play a special role for those in design-related fields. With rapidly changing technologies and evolving ideas on the best approaches to designing effective instruction, life-long learning strategies are especially important in the fields of software design and instructional design. However, a formal education is the foundation for professionals to build upon during their careers.

This study seeks in part to discover whether software designers working on educational software related projects have pursued formal education within the software design or instructional design fields, and, if so, to what extent they were prepared for their roles by that education. An investigation of standards and readings from Software Design and Instructional Technology related organizations did not reveal a focus on software design for educational software in either field. Therefore, it is difficult to predict what type of formal educational background professionals working as software designers on educational software projects will have.

The findings of this dissertation study will provide a better understanding of the range of formal educational backgrounds of those working in the area of educational software design. This may lead to recommendations for enhancing Computer Science, Software Engineering, and/or Instructional Technology programs to better meet the needs of those who will work in the industry. It remains to be seen whether this can best be done best emphasizing general skills which would foster good design regardless of the industry or purpose of the software in software design-related programs, or whether a special focus on instructional design principles would be

valuable. The Software Design standards reviewed would seem to indicate that current best practice leans towards a focus on a large set of general skills and knowledge, with an emphasis on hands-on experience. Similarly, it would be valuable to know whether those prepared by instructional design programs would benefit from learning more about software design principles or the skills commonly focused on in software design related programs. The Instructional Technology standards reviewed would seem to indicate that instructional designers may need skills which allow them to interact with, evaluate, and choose among a variety of technologies and need to be able to communicate with both technical and non-technical team members, but do not appear to suggest a strong focus on software design skills.

This study also hopes to address the types of non-formal education used by these particular software designers. The ongoing study of software designers working in a range of industries show that most software designers prefer self-study (using books, internet resources, and experimentation) in many situations, although employer, vendor, or third-party provided training and access to subject matter experts are also important resources. It will be interesting to learn whether the same methods are preferred by software designers working in instructional design organizations. It will also be illuminating to discover whether non-formal educational opportunities sought by these software designers are primarily focused on areas which would be valuable regardless of industry (such as programming languages, new technologies, and general communication skills), or whether information is sought which is particular to instructional design or instructional design related projects. This may in turn help inform related formal educational programs.

# 3 Chapter 3: Methods

As stated in Chapter 1, this study will investigate the central research question:

*What formal and non-formal educational experiences do software designers currently working in instructional/educational technology related fields report having experienced, and to what extent do they feel these experiences have prepared them for their current roles?*

This will be explored by addressing the following sub-questions:

1. *What are the primary role(s) played by the participants?*
2. *What formal education do the participants report having had? In what ways do they perceive these experiences have prepared them for their current role(s)?*
3. *Where are there gaps in the competencies (e.g. skills, knowledge, and attitudes) acquired through the formal educational experiences of these software designers? What topics are underemphasized by formal educational experiences?*
4. *What types of non-formal educational opportunities have these software designers sought or taken part in? How do these software designers seek and select these educational opportunities after they have joined the workforce? In what ways do they perceive these experiences have prepared them for their current role(s)?*
5. *What type of formal education do participants recommend for those planning to work in this field?*

Based a review of the literature, not much appears to have been written about the specific population being studied, although similar topics have been addressed with similar populations. Furthermore, the population does not seem to be well defined and it is not clear how to access a representative sample. Therefore, this study was exploratory in nature. A mixed methods approach was used to enable me to explore the topic in both depth (through rich description as

provided by analyzing data acquired through extended interviews) and breadth (through accessing a large and diverse group of survey participants and analyzing whether there are statistically significant trends indicating gaps in education or areas which are best learned through non-formal education). Comparing and integrating the results from both analyses provided a fuller and more reliable picture of the current situation than could be obtained by a single method alone.

This has been designed as a three-phase study. The use of Mixed Methods was chosen for three primary purposes (as described by Creswell and Clark (2007)):

1. *Triangulation Design:* Compare and contrast qualitative and quantitative results, to take advantage of the strengths of each technique.
2. *Exploratory Design: Instrument development and refinement:* The results of each phase inform the development of the instrument for the following phase. In particular, results of the analysis of Phase 1 interviews were to help shape the questions and especially the closed-ended response options in the Phase 2 survey instrument. Findings from Phase 1 and Phase 2 inform both the goals (what open issues need to be addressed?) and specific questions (what are the best ways to follow up on these issues?) in Phase 3.
3. *Explanatory Design:* Qualitative results are meant to explain quantitative results, and quantitative results are meant to build on qualitative results.

In addition to these primary purposes, Phase 1 was meant to help provide ideas for Phase 2 recruitment (by eliciting suggestions as to where potential participants could be reached), and Phase 2 was also to serve as a recruitment mechanism for Phase 3 (by offering survey participants the option to provide contact information for follow-up interviews).

Each of these purposes aligns with an existing mixed-method model, although none of the models explored met all three of these purposes.

According to Creswell and Clark (2007), the "Triangulation Design" is the most common Mixed method approach. The purpose of this design is to "directly compare and contrast quantitative statistical results with qualitative findings or to validate or expand quantitative results with qualitative data" (p. 62). The "convergence model" variant of this design is most similar to my original study design. "In this model, the researcher collects and analyzes quantitative and qualitative data separately on the same phenomenon and then the different results are converged (by comparing and contrasting the different results) during the interpretation" (Creswell & Clark, 2007, p. 64)

In the "Instrument Development" variant of an "Exploratory" mixed method model, a topic is "qualitatively explore[d]" with a small number of participants in order to "guide the development of items and scales for a quantitative survey instrument" (Creswell & Clark, 2007, p. 77). According to Creswell and Clark, this technique is always used in the development of a quantitative instrument, and researchers typically emphasize the quantitative aspect of the study.

An "Explanatory Sequential Design" is typically a two-phased mixed methods design which begins with collection and analysis of quantitative data and then moves to a second phase in which qualitative data is collected and analyzed. In the "Follow-up Explanations" variant of this model, quantitative findings are generally emphasized, and the qualitative findings are used to explain and expand upon specific quantitative results of interest. In the "Participant Selection" variant, the primary purpose of the quantitative phase is to "identify and purposefully select participants for a follow-up, in-depth, qualitative study" (Creswell & Clark, 2007, pp. 73-74)

As part of their seven-step process for selecting an appropriate Mixed Methods design, Teddlie and Tashakkori (2009) explain that after reviewing existing models to find the most appropriate research design, you may have to "combine existing designs, or create new designs, for your study", if you cannot find one that is a "perfect fit" (p. 163). When initially planning this study, I created the representation in Figure 1, a modified version of a triangulation design, to depict my plans for the study. This depiction was intended to show that the primary purpose of mixing methods was to triangulate the findings. The arrows which lead from the results of each phase to the data collection of the next indicate that the results of each phase were to influence the development of data collection instruments of subsequent phases.

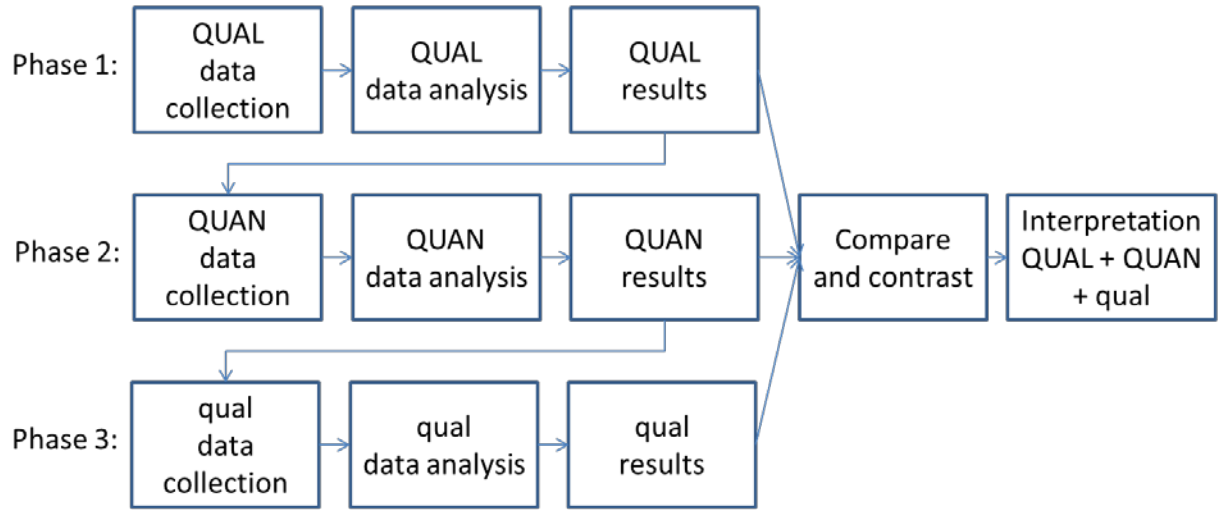



**Figure 1: Study Design (Modified Convergence Triangulation Design Model)**

However, in reality, my process was much less clean and more iterative. As Teddlie and Tashakkori (2009) explain in the seventh and final step to selecting an appropriate design,

> In some cases, you may have to develop a new MM [Mixed Method] design, using flexibility and creativity, because no one best design exists for your research project, either when it starts or as it evolves. Some MM studies change

> over the course of the research, resulting in designs with more strands than originally planned or with strands that change in relative importance. (p. 164)

As Teddlie and Tashakkori predicted, as my data collection and analysis progressed, I found that I needed to make modifications to my initial plan in order to optimize my use of the limited numbers of available participants to address my open-ended research questions. Figure 2 gives a more accurate representation of the actual process followed. Mixing was both sequential (with the results of each phase feeding into the next) and iterative (analysis of new data impacted my understanding of data previously analyzed, inspiring additional phases of analysis). Each phase expanded on the results of the previous phase(s). This design is probably most similar to a "fully integrated mixed method design". According to Teddlie and Tashakkori (2009, p. 151), "In these designs, mixing occurs in an interactive manner at all stages of the study. At each stage, one approach affects the formulation of the other and multiple types of implementation processes occur". However, data analysis of quantitative and qualitative was not truly combined in this study (that is, data were not converted from one type to another, although interpretation of qualitative results informed the selection of qualitative data to analyze, and patterns seen in the quantitative data in some cases confirmed or refined my interpretation of certain elements of the qualitative data).

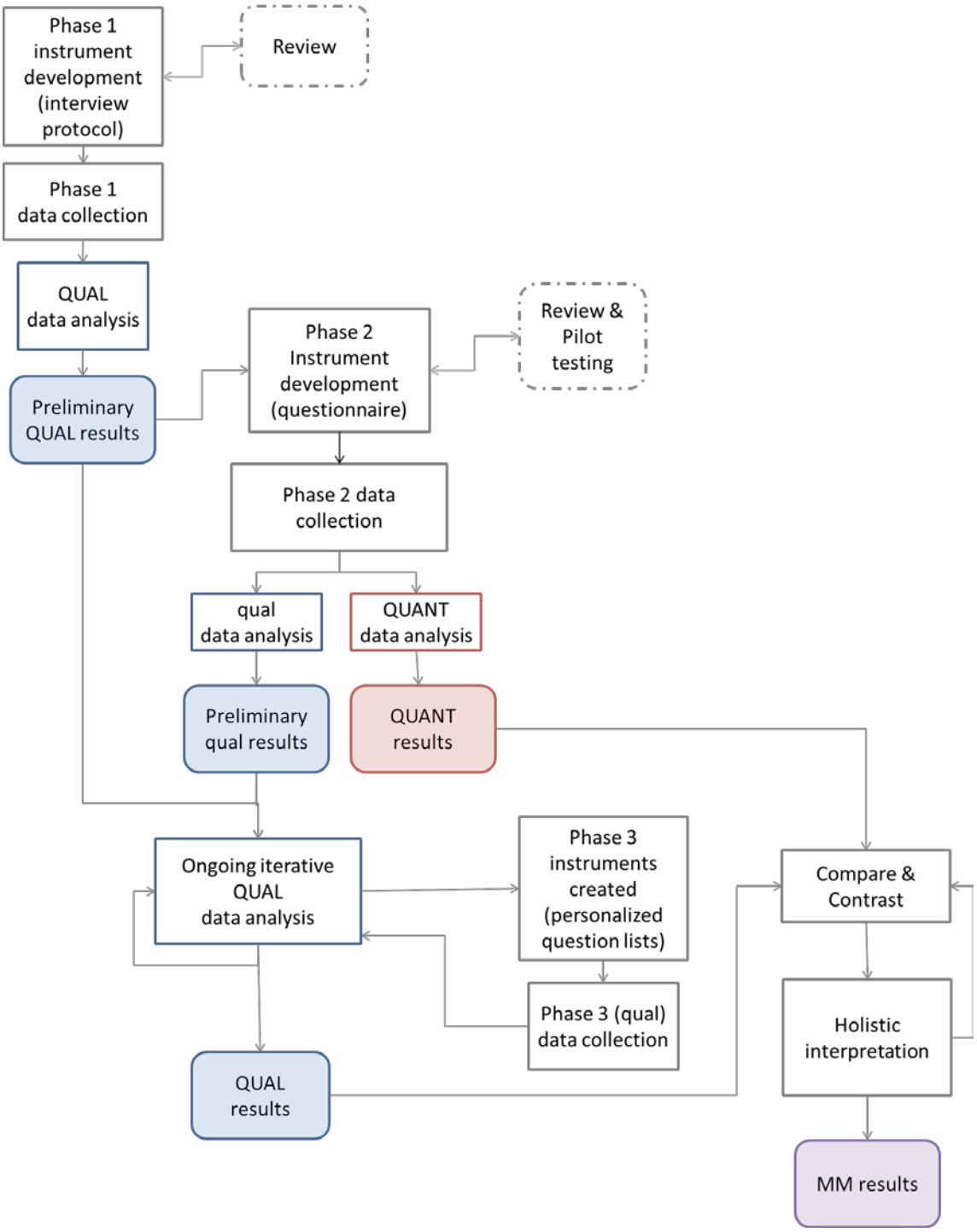



**Figure 2 Actual sequence of events**

In interpreting Figure 2, please note that data collection was sequential (that is, Phase 1 data collection was completed before Phase 2 data collection commenced, and Phase 2 data

collection was completed before Phase 3 data collection commenced.) However, data analysis was ongoing and iterative. This is described in greater detail in the Data Analysis section (3.5).

## 3.1 Terminology Used:

### *3.1.1 Software Designer*

Although "Software Developer" or "Computing Professional" may be more applicable generic titles (based on my reading of job advertisements and literature), I assume that the term "Software Designer" may better convey my meaning to an audience with a background in Instructional Design, as the term "development" conveys a slightly different meaning between the two fields, and the term "computing" may be understood more broadly than it is intended.

Therefore, for the purposes of this study, the term "Software Designers" will be used to describe a professional currently or recently involved in the design and/or development of software. This includes those involved with systems/software architecture, requirements gathering, specification writing, high-level design, low-level design, programming, user experience design, and/or quality assurance. The Software Designers may perform additional roles, such as Instructional Design, Project Management, or other supervisory or high-level planning related roles, but must spend at least some time on the software design roles mentioned here. Software being developed is primarily meant to serve the needs of a client and/or users (rather than software designed primarily to fulfill a course requirement or as a personal hobby).

### *3.1.2 Computing Education*

In alignment with much of the literature covered in the literature review, the term "Computing Education" will be used to refer to degree programs in Computer Science, Software Engineering, and similar or cross-over degrees.

### *3.1.3 Educational Software*

While I have not found a good definition in the literature for the term "educational software", for the purpose of this study, "educational software" will be understood to include any software that fosters teaching and learning, including learning management systems and lecture capture software as well as more traditional drill-and-practice applications, e-learning modules, and other types of software used by students to aid in learning. Software used for purely administrative purposes in an educational setting (e.g. billing or personnel management) will not be included.

### *3.1.4 Competencies: Skills, Knowledge, and Attitudes*

The central research question addressed by the study asks how formal and non-formal educational experiences prepare software designers for working in instructional or educational technology related projects. Several of the sub-questions implicitly address the *competencies* that are developed through these experiences. The IBSTPI defines "competency as "an integrated set of skills, knowledge, and attitudes that enables one to effectively perform the activities of a given occupation or function to the standards expected" (Richey, et al., 2001). For the purpose of this study, the term "competency" will be used to denote a skill, piece of knowledge, or an attitude held by a student or working professional.

### *3.1.5 Formal and Non-formal Education*

The definitions used for "formal" and "non-formal" education align with those discussed in the section of the literature review regarding "lifelong learning". The related terms to be used in this study include:

***Formal Education:*** Education acquired directly from a college/university program. This does not include individual courses taken post-graduation based on personal interest or employer needs, or employer or third party sponsored training.

***Non-Formal Education:*** Educational opportunities which take place outside of formal college/university programs, including, but not limited to, training provided by employers or third parties, mentorship by co-workers or other peers, and self-directed learning.

***Self-directed learning:*** Self-study, which takes place outside of a formal educational setting or employer or training courses provided by employers or third parties. This may involve the use of online resources, books, manuals, journals or other publications, or the use of techniques such as experimentation for the purposes of gaining a better understanding of a new concept, approach, technology, or programming language.

## 3.2 Participants

Participants selected in any phase of the study met the following criteria, as stated in the study proposal:

- The participant is a Software Designer. A participant will be included if the participant's primary role(s) focus on the design and/or development of software (including architecture, requirements gathering, high- or low-level software design, programming, and/or user experience design). Possible titles/roles may include software architect, software designer, software engineer, programmer, computer scientist, database designer, database manager, research & development (if related to software design/development), etc. NOTE: Participants who are retired or unemployed but have previously filled one or more of these roles were also considered eligible for participation.

- The participant's primary role(s) involves the development of educational software for use either within the same organization or for external users. The primary purpose of this software is to educate, instruct, or facilitate learning and teaching.
- The participant's software design experience is or was professional in nature. This may include full-time or part-time employment, but does not include time spent on work primarily intended to be a class project.

## 3.3 Participants

### *3.3.1 Phase 1: Interviews.*

Participants for the first phase of this study were identified via a snowball technique, beginning with recommendations provided by the researcher's own personal contacts. Screening questions were emailed to potential participants to determine whether they were or had recently been substantially involved in the design of software used in K-12 or higher education in a professional capacity. This included participants who played one or more of the following roles: systems/software architecture, requirements/specification writing, high-level and low-level design or programming (where programming tasks include some individual or collaborative design work), and user experience design.

Nine participants who met the inclusion criteria participated in interviews (See "Appendix F: List of Participants" for more details on their backgrounds). These included employees from universities, large companies, and small companies (including single-person companies) which created LMS extensions (e.g. Sakaii components) (3 participants), language education applications (2 participants), and three individuals who worked on a lecture capture system, e-learning application, and a web-based interactive multimedia application, respectively. All nine participants were male.

### *3.3.2 Phase 2: Survey.*

A number of different strategies were used to recruit participants, including posting invitations in listservs, linkedIn groups, and other discussion forums, as well as the use of a snow-ball technique starting with the researcher's own personal contacts. The online questionnaire was accessed by 215 participants, 70% of whom (151) were eligible to participate based on their answer to the initial filtering question ("Have you ever been involved in the design/development of software used for educational or instructional purposes in elementary, secondary, or higher education?"). Of these, many dropped out along the way, possibly due to the length of the questionnaire. Nintey-four (43%) completed at least some of the questionnaire beyond the first question, and 74 (34%) completed the entire questionnaire.

Of participants who completed the entire survey, 74% (53 of 74) currently worked in the United States, 5% (4) worked in the United Kingdom, and the rest were from various other locations around the world. Formal education levels varied between one non-high school-graduate to 44% (38 of 87 who reached the relevant question) with doctoral degrees. As mentioned in the limitations section, participation may have been skewed by the relatively high participation of members from academic listservs).

### *3.3.3 Phase 3: Follow-up Interviews*

The purpose of Phase 3 was two-fold: (1) To clarify any gaps or discrepancies between Phase 1 and Phase 2 findings; and (2) to interview participants who fit profiles identified in Phase 2 which were not represented during Phase 1. Therefore, Phase 3 participants are a subset of those who participated in Phase 2. Forty-four Phase 2 participants indicated their willingness to participate in a follow-up interview. Each of these participants was sent a personalized email (see the Procedures section for more details). Of these, one turned out to be a Phase 1 participant

and was not sent an email. Two others were not contacted because based on a mismatch between their demographic characteristics and the specific questions I was addressing at this stage. Participants were given the option to respond to questions via email (with follow-up conversation via email), or to schedule a real-time phone or Skype interview.

Of the 41 who were contacted via email, six initially responded. One wished to schedule a real-time interview but scheduling did not work out. Another responded, but responses indicated that his primary roles did not meet the criteria for inclusion in the study and his responses to these questions were not relevant to the roles discussed in this study. The other four participants responded to questions via email, and three of four responded to follow-up questions. Therefore, there were a total of four participants in Phase 3.

## 3.4 Procedures

### *3.4.1 Phase 1: Interview Procedures*

The semi-structured interview protocol was reviewed by members the research committee (as part of the dissertation proposal process) and by peers with experience in this area. As recommended for a naturalistic inquiry, I developed myself as a "human instrument" throughout the interview processes (Lincoln & Guba, 1985). This allowed me to use my own tacit knowledge and previous experiences to guide the adaptation of questions during the interview based on what the participant seemed to be saying (either by taking questions out of order or rewording or modifying them as appropriate to the participant's own train of thought and based on the participant's experiences. For example, if a participant had not taken any computing courses as part of university degree(s) he had attained, I would leave out the section on what was learned from computing-related courses and focus more heavily on what he felt he had learned or gained from "unrelated" courses that may have helped him in his future career).

The protocol was clarified and added to as interviews proceeded and early analysis was conducted. Therefore, different participants responded to slightly different questions.

The semi-structured interviews were conducted over the phone or via Skype, and were recorded, with participants' permission. Although participants were initially informed that the interview would take 30-40 minutes, if the interview was not completed within that time, the interviewer indicated that the requested time was up and asked whether participants were willing to continue. All participants elected to continue. At the end of the interview, participants were asked whether they would be willing to participate in member checking. Eight of nine Phase 1 participants indicated they would be willing to participate.

Interview data was transcribed. Every word was recorded as said if possible. Verbal ticks such as "um", "uh", and repeated words were left out of the transcription unless they indicate thought (e.g. if the speaker paused for more than a few moments after saying "um", the "um" was recorded). If individual words or phrases could not be understood after multiple passes, these were marked in brackets. The missing data did not generally impact my ability to follow the participants' meaning. After transcripts were completed, personally identifying information (such as employer names) were replaced by codes (e.g. <employer: university> for a participant who works at a university).

Transcriptions were completed on all interviews, with the following exceptions:

1. One was not recorded, but extensive notes were taken. The notes for this interview were sent to the participant, who made some corrections or additions directly to the notes. This version was used for analysis.
2. One section of one additional interview turned out to be missing. By the time this was discovered, the interview had taken place over a year earlier and it was unlikely that

the participant would recall his original words. Because of the researcher's style of note-taking (an attempt at verbatim transcription), notes were fairly complete. However, no direct quotes will be used from this section.

Audio recordings will be destroyed once the dissertation has been successfully defended, indicating that I have no additional need for them. Transcripts were initially created in MS Word (where participants were identified through initials only, which would allow the researcher to tie transcripts to emails and contact participants if necessary prior to completion of the analysis), then transferred to the QSR International's NVivo 9 qualitative data analysis software (hereafter referred to as "NVivo"). Personally identifying information (such as the name of employers) was omitted from the version stored in NVivo, and the MS Word documents will be erased after the dissertation has been defended and I no longer need to link these to specific participants (e.g. for the purposes of member checking or follow-up questions). Participants were assigned to codes which replaced their initials within the NVivo system.

### *3.4.2 Phase 2: Survey Administration*

A preliminary Phase 2 survey was provided as part of the dissertation proposal. This was modified based on findings from Phase 1. The adapted survey was reviewed several times by the committee chair, after which the following activities occurred:

1. A paper-based version was pilot-tested on a fellow graduate student with similar demographics as my target population. A think-aloud protocol was used, along with a follow-up interview. A number of weaknesses were revealed, including several questions which were not interpreted as I had intended. The survey took over an hour to complete, which was identified as a major issue.

2. The paper version was revised, and then reviewed by another member of my committee. Modifications were made to streamline the survey.
3. The revised survey questions were entered into the "Survey Monkey" system. It was tested by myself and a volunteer. We each entered realistic data several times in order to ensure that skip-logic and other functionality worked properly.
4. The web-based survey was pilot-tested on a fellow graduate student with similar demographics as my target population. Once again, a think-aloud protocol and follow-up interview was used. The survey continued to take longer than the target 15-30 minute timeframe.
5. I revised the survey once again, removing one question set, which may be used in a future study, and reorganizing other questions to make the survey experience flow better. The previous volunteer tested the survey again and discussed possible revisions with me.
6. The final draft version was piloted by three graduate students with demographics similar to the target population. One of the three had already participated in a pilot of the paper version – the other two had not previously seen the survey. The survey was taken on their own PCs and they recorded the time it took, then gave me feedback.
7. The final draft version was also reviewed by a peer who had worked with me on a related study, and the previous volunteer, after discussing the results of the pilot testing.
8. The survey was revised based on feedback from the reviewers and the pilot test results, then was reviewed a final time by my committee chair.

The final version of the survey can be viewed in Appendix B: Phase 2 Survey instrument.

The survey was administered via the SurveyMonkey online survey hosting system SurveyMonkey (http://www.surveymonkey.com). This site provides password-protected

security, as well as features allowing for a survey to be easily created and updated, and for results to be exported.

Two forms of invitation were sent out. One was directed to potential participants. The other was directed to people who might have contact with potential participants (such as instructional designers). On the final page of the survey a link was provided which could be forwarded to others who might be interested in participating. A unique link was generated by the SurveyMonkey system for each contact mechanism I attempted (e.g. each linkedIn group, list-serv, discussion forum, etc.) This did not allow me to identify individuals, but did allow me to review how effective each recruitment method was.

Participants were contacted in a variety of ways, including invitations sent to list-serves, organizational mailing lists, and to personal contacts who were requested to send the invitation on to those who match the target demographics. In order to do this, I became a member of a number of groups and listservs. In determining whether a group was appropriate, I first looked at the group description and the ongoing discussions (if available to me). I then reviewed the rules (if available) and existing posts. If the rules appeared to allow general topic posts by members and/or I already saw invitations for participation in other surveys or polls, I posted the appropriate invitation. If not, I wrote to the group owner or manager to ask whether I could post my invitation. Not all owners or managers responded within the timeframe during which I was collecting data. A few of them posted to the listserv for me. I also located a number of organizations which provided lists of educational software companies. I emailed or filled out contact forms, requesting that the invitation be forwarded to individuals who might be eligible or interested in participating. Finally, as mentioned earlier, a message on the last page of the survey

included an invitation that could be copied and forwarded to friends who might be interested in participating.

Participants who completed the survey could indicate they were willing to participate in follow-up interviews by providing an email address through which they could be contacted. As a thank-you for participating, participants could additionally provide an email address to be entered into a pool for a gift-certificate. Participants were given the option to provide their email address separately for each of these purposes, and some elected to volunteer for the interview but not sign up to enter the pool, and vice-versa. These email addresses were used only for these purposes, and not viewed while analyzing survey data.

One gift-certificate was sent to a participant. All email addresses provided for this purpose were entered into a spreadsheet. A volunteer was asked to select a number, and the email address on the row corresponding to that number was selected. The first participant contacted did not respond to my initial or follow-up email. Therefore, the procedure was repeated and a second participant was chosen. This participant responded, and received the gift certificate via email.

### *3.4.3 Phase 3: Interview Procedures*

An individualized email was sent to each potential Phase 3 participant. The email invited these participants either to respond directly to the interview questions (included in the body of the email), or, if preferred, to schedule a time for a phone or Skype call. The email option was given in order to allow participants to respond when it suited them, avoiding the scheduling issues which had reduced participation in Phase 1.

Each email message was tailored to the individual recipient and included the questions I wished to cover, based on that individual's demographic characteristics and on specific

responses to survey questions. Questions covered some or all of the following areas, depending on the individual's survey responses:

1. *General clarifications* (e.g. for participants who indicated they had a PhD but no other degrees, I asked whether they had any previous degrees. For participants who indicated their current title was "Assistant Professor" or "High-school Teacher" but who also design and develop software, I asked more about the role that software development plays in their job).
2. *Specific questions regarding Computing- or Instructional Design- related degrees that had been completed after 2005.* This was addressed for three main reasons: (1) Recent graduates were not a group included in Phase 1, (2) I wished to gain a better understanding of differences seen in statistical data between recent grads and others, and (3) I wanted to understand what is being done in current programs, so that recommendations would accurately address what is already being done well or could be improved based on other findings. These questions:
    a. Specifically asked about group and real-world projects done as part of the program
    b. For those with a Computing background only, asked about domain-specific experiences
    c. For those with an Instructional Design or related background only, asked whether programming or other technical matters were covered

    Table 5 and

    Table 6 show examples of questions written for a recent Computing graduate and a recent Instructional Design graduate, respectively.

Table 6 also includes an example of a clarifying question regarding the formal educational background reported by the survey participant.

3. *Specific areas covered well by programs.* The purpose was to attempt to gain insight into differences between groups seen in statistical comparisons.

   In order to keep the question list to a reasonable length, I narrowed the items in this section down as follows: After identifying items that had statistically different responses between groups (based on educational background or experience level), I omitted items that had obvious reasons (e.g. it is clear why participants with a formal education in a Computing related field had, on average, more coverage of programming languages than those without it). Other items were combined or omitted. For example, self-learning related items were combined into one item, "teach yourself things you need to know", and communication related questions were combined into one item addressing "communication skills". This resulted in the following list of items:

   1. Maintaining code over time
   2. Legal aspects of industry
   3. Business aspects of industry
   4. Interface design and user experience design principles
   5. Designing and Developing INSTRUCTIONAL software
   6. Teach yourself things you need to know
   7. Communication skills
   8. Work well in teams
   9. Working directly with users

10. Testing practices
11. Education theory courses
12. Designing and developing instructional software

Although statistically significant differences were not seen in the following items, they were also included, because they had emerged as crucial skills in both qualitative and quantitative analyses.

- Critical thinking
- Problem solving

For each participant, only the items that the individual marked as having been prepared "very well" by their formal education were included. Additional questions asked whether their educational experience was particularly good or particularly lacking in any other areas. Examples of questions are included in

Table 7.

4. *Ideal degree program.* For each person, I asked follow-up questions on their recommendations regarding an "Ideal bachelor's degree program." I reminded them of their answer, then included follow-up questions which addressed the following areas:
    a. Follow-up questions where something was unclear, or where I was interested in learning more about a specific response (e.g. "what do you mean by 'art'? why do you feel it is important?")
    b. A question asking them what roles they believe this program would prepare someone for. This allowed me to determine whether their suggestions were

intended to inform a general program for "Software designers of educational software" or were intended for a more narrow audience.

c. Questions exploring the degree to which they agreed with various program traits which emerged as themes in the data. In many cases, only a handful of participants specifically addressed these areas, but statistical data and my own impression from earlier interviews led me to believe that if they had been asked directly, the majority of individuals would agree that most of these traits should be included in such a program. I also was interested in learning how they believed that these traits could be incorporated into a program.

An example of an individualized Phase 3 interview email can be found in Appendix C: Phase 3 interview protocol: Sample of a personalized email.

**Table 5:** ***Questions asked to recent Computing graduate***

| |
|---|
| **1. Did any of your coursework the degree you received in 2007 involve engaging in real-world or realistic projects? If so, could you briefly describe what these projects entailed (e.g. duration, type of project, group vs individual project, whether it was for a real client or a hypothetical problem)? Was this helpful in preparing you for your current or previous professional role(s)?** |
| **2. During that degree, were any courses or activities focused on one or more specific domain (e.g. "Educational Software", "Games", etc.)? Was this valuable in preparing you for y our current or previous professional role(s)? If so, why?** |

**Table 6:** ***Questions asked to recent Instructional Design graduate who did not include any degrees prior to a PhD***

| |
|---|
| **1. In your survey response, you indicate that you received a doctoral degree in 2007. You also indicated that your majors included Educational Technology/Instructional Technology and Educational Leadership, with a minor or specialty in a physical science. You did not indicate any other degrees. Is this accurate?** |
| **2. Did any of your coursework the degree you received in 2007 involve engaging in real-world or realistic projects? If so, could you briefly describe what these projects entailed (e.g. duration, type of project, group vs individual project, whether it was for a real client or a hypothetical problem)? Was this helpful in preparing you for your current or** |

**previous professional role(s)?**
**3. Were any courses or activities focused on programming or related skills included in your Instructional Design-related degree? If so, what did this entail? Was this helpful in preparing you for your current or previous professional role(s)? If so, how?**

**Table 7:** ***Questions relating to areas well-covered by an individual's formal educational experiences***

1. **In your survey response, you indicated that degree program(s) you attended did a good job at covering the following areas. For each, could you please briefly indicate how this skill or topic was covered or fostered within the courses you took?**
    a. **Designing and developing instructional software**
    b. **Testing skills**
    c. **Ability to work well in teams**
    d. **Critical/analytical thinking**
    e. **Problem-solving skills**
2. **Were there any other areas relevant to your current professional position you feel the program(s) you attended excelled at?**
3. **Were there any experiences that you felt were really lacking in your own educational background?**

**Table 8:** ***Questions relating to an "Ideal bachelors program" (this is an example of questions written based on a specific response)***

**As you may or may not recall, the survey included an open-ended question regarding your suggestions for an "ideal bachelor's degree program to prepare someone for your current position". I have a few follow-up questions on this topic.**
**As a reminder, you indicated that the best type of degree would be an Instructional Systems Technology/Instructional Design degree. You indicated that it is not important for students to have a domain-specific background. You also indicated that a background in programming and other technical aspects of software design/development is important, but not as important as a background in education or instruction.**
**When asked for other useful program traits, you responded: "Educational Psychology & Learning Theory Instructional Design I and II Computer-Based Software Design, Human Computer Interface Design & Testing, Instructional Simulation Design, Game Design Human Performance Assessment/Measurement Consulting, Project Management Technical Writing Advanced Algebra, Advanced Geometry, Linear Equations, and Calculus (for some simulation programming)"**

1. **The question asked you to address the program in terms of preparation for "someone in your current position". What type of roles do you believe this type of program might prepare them for?**
2. **Do you feel that game design and the mathematics courses you recommend (which you indicate are useful for simulation programming) should be required**

**in all ID/IST programs? Or would these courses form a specialization area?**

**3. Similarly, do you feel that Human Performance topics should be covered in courses included in every ID degree?**

4. **Individuals who participated in the survey indicated that the following areas are important in developing an ideal program. To what extent do you believe that each of these should be incorporated into the program? Do you have any additional suggestions on good ways to incorporate these competencies into a degree program?**
   - **Foster creativity**
   - **Foster critical thinking skills**
   - **Develop artistic or visual design skills**
   - **Foster the ability and interest in continuous on-the-job learning**
   - **Gain experience with skills and tools used in real-world problems on the job**
   - **Give lots of practical experience**
   - **Provide a solid foundation in software engineering theory and practices**
   - **Provide a solid foundation in software development/programming theory and practices**
   - **Provide a solid foundation in instructional design theory and practices**
   - **Provide a solid foundation in user interface design theory and practices**
5. **Is there anything else you would like to add on this topic?**

If a participant responded via email, I wrote follow-up questions on areas that had been unclear or particularly interesting in the earlier message. In several cases multiple rounds of follow-up questions and responses occurred. The resulting discussion was similar in pattern to a transcription of a real-time semi-structured interview, although responses tended to be more concise and on-topic than in Phase 1 verbal interview transcripts.

## 3.5 Data Analysis

The data analysis was performed in stages after each phase, as described in the following sub-sections.

### *3.5.1 Qualitative Data Analysis*

#### *3.5.1.1 Overview of Qualitative Data Analysis Procedure*

Qualitative data included: Phase 1 interview transcripts, responses to open-ended items in the Phase 2 survey instrument, and responses to Phase 3 "interview" emails (which included

responses to both initial questions and follow-up questions, as addressed in a chain of emails with an individual participant).

Qualitative data analysis was conducted using the Constant Comparative Method for Naturalistic Inquiry (Lincoln & Guba, 1985) to find common themes within and across data collected from each of the three phases. Lincoln and Guba's technique includes the following steps (paraphrased, relevant material on pages 344-350):

1. Unitize the data, by breaking the text up into units that can stand on their own. Each data unit should be transcribed on an index card.
2. Categorize the data, as follows:
    a. Select an initial card from the pile. This will be the first card in the first category.
    b. Select the second card and determine whether it is a "feel-alike" for the first card. If so, add it to the initial category, by stacking it on the first card. If not, the second card begins the second category.
    c. Continue this process with all successive cards.
    d. Retain cards that appear to be irrelevant in a separate pile.
    e. Once categories have reached a critical size, write propositional statements to characterize the cards. Transform these into rules for inclusion for each category.
    f. Continue with steps c-e until all cards have been exhausted.
    g. Review the entire category set, as follows:
        i. Go through the "miscellaneous" pile, to determine whether cards should be moved into existing categories.

ii. Review categories to determine whether they overlap.

iii. Review categories to find potential relationships between them. Mark missing, incomplete, or unsatisfactory categories for follow-up in further data collection and data analysis.

h. Perform additional data collection relating to categories identified as missing, incomplete, or otherwise unsatisfactory in step 2.g.iii.

i. Determine when to stop based on exhaustion of sources , saturation of categories, emergence of regularities, and overextension

j. Review the entire category set once again to ensure nothing has been overlooked.

I have used this process on previous projects and it has worked well. However, for this study I decided to use the QSR NVivo data analysis software, after some initial exploratory coding done by hand using margin-notes. Using the NVivo software required me to make a number of modifications to this technique. The software allows a hierarchical model to be built and rearranged on-the-fly as data is reviewed and coded. Therefore, unitization and categorization occurred simultaneously, as I created a new "node" (or added to an existing node) once I identified a unit. Each node was given a unique name. As the number of nodes grew, I began combining them into categories and super-categories. Figure 3 shows how the node hierarchy is displayed and managed within the NVivo software (note: only one set of nodes is fully expanded here).

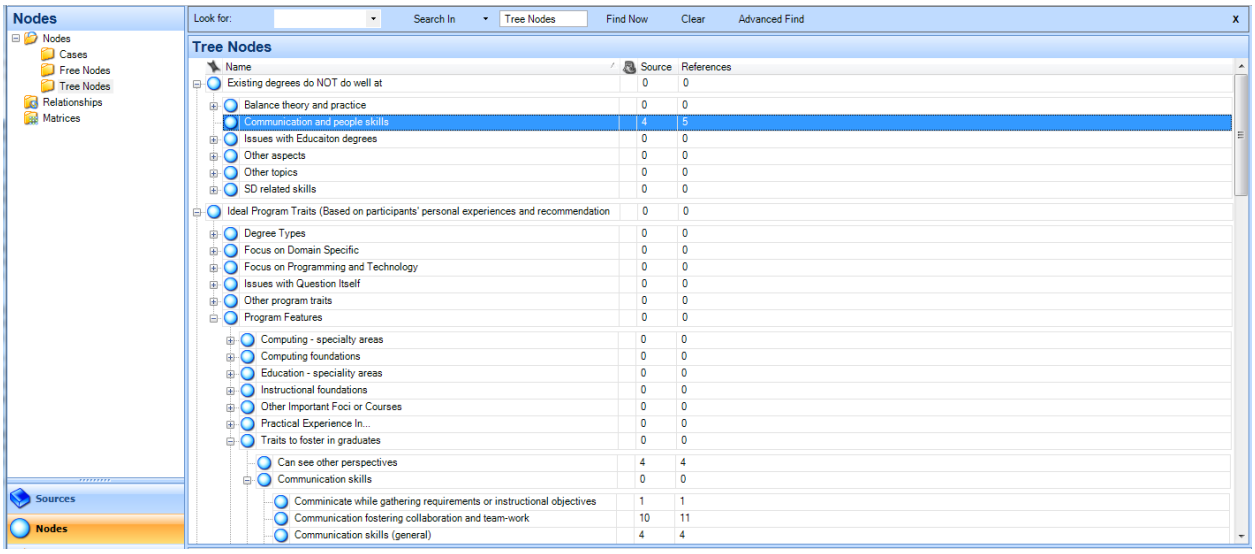



**Figure 3:** Node Hierarchy in NVivo software

As the analysis proceeded and additional data was added, new types of categories began to emerge and the definitions of some categories created earlier were modified. I re-visited all pieces of data multiple times as node definitions changed. External reviews of my coding hierarchy (discussed more in section 3.5.1.3) caused me to re-think some categories, after which I also did an additional pass through the data. Approximately a dozen full or partial passes were made through the data over time, in addition to some initial tentative coding I did by hand on the first few Phase 1 transcripts, and another by-hand pass done on the open-ended Phase 2 data prior to importing it into Nvivo.

### *3.5.1.2 Unit of Analysis*

The unit of analysis used was a text segment that stands alone as a single coherent idea. These segments varied from a single word or sentence to one or more paragraphs. For example, the following segments were both coded within the code hierarchy as: "Ideal Program Traits→Program Features→Traits to foster in graduates→Communication Skills→Communication fostering collaboration and team-work".

*Segment 1:* "collaboration" (Survey Participant #71). NOTE: this was in response to a question regarding traits of an "ideal program". The full response given by this participant to this open-ended question was: "collaboration, creativity, problem solving"

*Segment 2:*

> I don't regret one minute I spent as an English degree student, because those skills have helped me to communicate with my supervisors, with my peers, with my subordinates, with customers…I mean, I could not have taken, in my opinion, a more valuable skill into the workplace.
>
> Because I'm an effective communicator, I feel like I've been given more opportunities to participate in strategic circles, because a lot of the people who strategize don't necessarily know how to communicate, and they sort of invite you in because they know that you'll do a good job of it and then suddenly you're sitting at the table. I may be overstating this a bit, but I really do feel strongly that it's made a big difference in my career.
>
> (Interviewee 9).

Because of the richness of the data and because my research questions required me to look at the data from multiple perspectives, a block of text might be coded in multiple overlapping nodes. For example, the following transcript segment included a number of overlapping codes:

> So, the easy answer I have is that…is kind of to harp on what I think is failing in CS curriculum, which is that…<sighs> the hardest part of working in software development, and especially web development, is knowing how to work with

> other people and communicate with other people. There are lots of people who have great theory…whether that's even user exp designer theory, so, ok, I'm a user-centered designer, I studied in school, I kind of know what the processes are, or whether it's, oh, I'm a great coder, and I know how to code, and I can, you know, write a great algorithm, but I think it's much rarer to find people who know how to work with other people and work on a project and organize their time, and think about problems in…in creative ways, and in ways that break the problem down into manageable pieces. So, for me a big part of the education that people need to do great software design and development is really an education about how to work with other people, and how to work on projects. Great project management skills, great work skills, great communication skills. These are so critical. (Interviewee 3)

In this example, "kind of harp on…with other people" and "it's much rarer to find people who know how to work with other people" are both coded under the node "Existing programs do not do well at→communication and people skills". "There are lots of people….pieces" is coded as: "Existing programs do not do well at→Practical aspects of the job". A number of additional themes were identified that were added to program traits as part of the section on "Ideal Program Traits". For example, portions of this segment address communication skills ("knowing how to work with other people and communicate with other people"), project management related skills ("work on a project and organize their time" and "ways that break the problem down into manageable pieces"), creativity ("think about problems in…in creative ways"), and so on. In all, eight codes were applied to various portions of this

segment, with 2 or more overlapping codes in some areas. Figure 4 shows how this coding looks within the NVivo software.

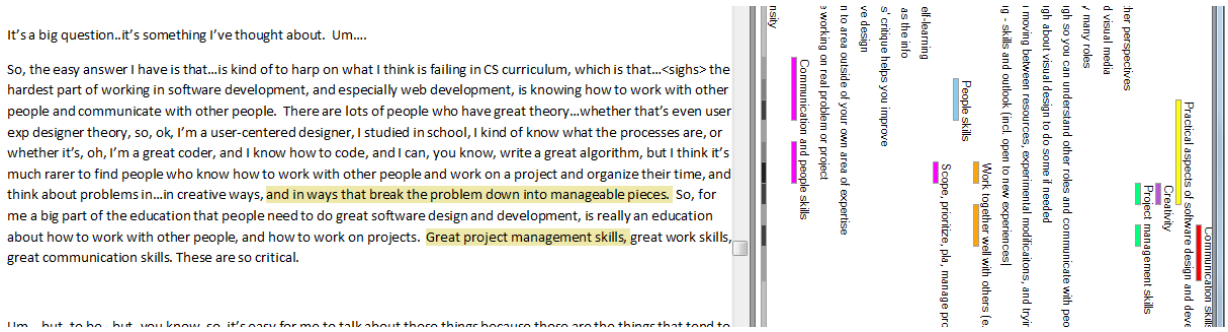



**Figure 4** Coding overlapping nodes in NVivo

### *3.5.1.3 Coding Data from each Phase*

After some initial rounds of coding on paper, Phase 1 interview transcripts were imported into the NVivo software and coded. Some categories that emerged from this data (such as demographic data regarding participant roles, types of degrees held, and types of software created) were used primarily to inform the development of the Phase 2 survey instrument and were later omitted from the overall coding hierarchy. The remaining categories were added to and modified as Phase 2 and Phase 3 data were added and subsequent rounds of analysis were done.

The open-ended responses to Phase 2 survey questions fell into two categories.

1. *Responses to "Other (please specify)" prompts for closed-ended questions.* These were analyzed individually (that is, only responses to an individual question were combined. This was done for the most part in MS Word documents, as the sophisticated NVivo tool was not needed in these cases).
2. *Responses to open-ended question relating to traits of an "ideal bachelors program for someone working in your role" and responses to related questions* (including those regarding the type of degree, whether the degree should be domain-specific, and

whether programming and other technical aspects of the work should be covered as part of the degree). The responses to the closed-ended related questions were coded as if they were text, along with responses to "other (please specify)" for these specific questions. This was done so that this set of data could later be compared and combined with Phase 1 data on the same topic (see below). An initial pass was made through all of these items on a paper printout to get a sense of the data. Then, data was entered into the NVivo qualitative analysis software. Each participant was treated as a case. For each participant who completed the "ideal program" questions, demographic data (including level of experience) was entered as "attributes" for the related case. Ideal program-related data was coded within the Nvivo system. Themes were allowed to emerge naturally through the analysis, and were combined and re-arranged in a coding hierarchy as coding progressed.

The Phase 1 and Phase 2 hierarchies were reviewed by my dissertation committee chair, and additional revisions were made. This primarily included splitting apart several sub-categories and merging others. Themes were re-named in order to clarify their intent.

Once these revisions were completed and an additional pass had been made through each set of data, the two hierarchies were merged into a single coding hierarchy for the entire data set. In the process of merging these two hierarchies, I discovered that there were a lot of overlaps between the major categories "Valued in own Formal Education" and "Recommendations for Ideal Degree Program". "Valued in own Formal Education" consisted of items that participants indicated were especially valuable or useful in their own formal educational experiences, while "Recommendations for Ideal Degree Program" included specific recommendations for courses, topics, and traits that should be included in a hypothetical "ideal" program. In some cases, it was

actually difficult to determine in which of these two areas a text should be coded, as participants regularly spoke about what they wish had been included in their own formal education, or referred to valuable aspects of their own formal education as examples of what should be done in an ideal program. Merging these together created a richer and better-organized single hierarchy. Similar choices were made at a smaller scale elsewhere within the data as it was re-reviewed at this stage.

The coding hierarchy was exported into a pdf file and sent to a peer with expertise in this area. This colleague has work experience in this area, has collaborated with me on related research projects, and is currently a faculty member in a Computer Science program. She also is very familiar with Computer Science-related standards, because she is currently involved in a curricular redesign. A summary of her feedback, which she has reviewed and approved, is included in Appendix D: Notes from external review of coding. Based on her recommendations, a few small changes were made to the coding hierarchy. She gave me feedback on a few areas I had specific concerns about (specifically, overlapping codes, and the merging of the "Valued in own Formal Education" and "Recommendations for Ideal Degree Program" categories). She also indicated that she felt that the coding structure "rings true" overall.

A small set of sub-categories was also sent to a peer with expertise in a specific area to review. Her recent research relates to the development of Design Judgment. I sent her a list of the nodes that I had categorized under "Skills and Knowledge Important on the Job→Design Judgment", and "Ideal Program Traits→Program Features→Traits to foster in graduates". Because this concept is difficult to grasp, her review was very helpful in identifying specific nodes that I felt belong in this area but did not truly meet the criteria.

All recommended changes were made.

Because only four cases were included in Phase 3, I elected to simply merge this data directly into the existing hierarchy. A few additional themes arose from this analysis, as well as from changes made after reviews by knowledgeable peers. Therefore, I did a final pass through the Phase 1 data to ensure that I captured any areas relating to these themes.

Finally, I created an outline of all qualitative findings to be used in the "Findings" section. In creating the outline, I revisited the contents coded under every node. This revealed additional areas where node names could be clarified or nodes should be merged or divided. These changes were all at the lowest level of nodes; no necessary changes were identified to the overall hierarchy at this point.

### *3.5.2 Quantitative Data Analysis*

A number of statistical procedures were used to analyze numerical data collected by the survey instrument.

Quantitative data was analyzed using descriptive statistics. Counts, means, and medians were used to describe characteristics of the sample or interesting sub-groups within the sample. Some calculations were provided by the SurveyMonkey system itself. Others calculations were performed within Microsoft Excel.

Differences between groups on binary items (such as roles played, which were entered as Yes/No for each role) were compared using a two-way contingency table analysis, which involved the application of a $\chi^2$ (Pearson Chi-square) test and use of a phi coefficient for estimating effect sizes (Green & Salkind, 2008). These statistics are used to "obtain an approximate test of the null hypothesis that two variables…*A* and *B* are statistically independent… [that is] the probability of *A* occurring is unaffected by the occurrence of *B*" (Kirk, 1999). In this case, these tests were used to determine whether membership in a group

(such as "those who had a background in Computing", "those who had a background in Instructional Design", "those who had a background in both", and "those who had a background in neither") correlated with participants' likelihood of playing specific roles, reasons for working in this field, impressions of skills needed for working in this field, and so on.

Non-parametric statistics were used to look for statistically significant differences responses on paired questions both for the sample as a whole, and between groups within the sample. These were primarily used to evaluate responses on unipolar 3-point Likert-type scales which were used to indicate the importance of various skills and knowledge on the job, and the degree to which these skills and knowledge were covered in formal education. These include the Mann-Whitney U test, Independent Sample Kruskal-Wallis, and Wilcoxon Signed Ranks. These tests were conducted using IBM ® SPSS 18 ®[3]. These statistics, which compare ranks rather than means, "are useful for problems that include one or more variables measured on a nominal or ordinal scale" (Green & Salkind, 2008, p. 349), and in cases where "the distributions of the test variable for the two populations do not have to be of any particular form (e.g., normal)" (p. 378), as is the case in this study. (In fact, I expected that the distribution would be skewed on some items. For example, as I anticipated, nearly all participants indicated that "critical thinking" is very important in a job in this field, with the result that a 3 on a 3-point scale was a near-universal score for this item across all groups).

## 3.6 Member Checking

Eight of nine Phase 1 participants and three of four Phase 2 participants indicated their willingness to participate in member checking. Each of these individuals was sent a draft of the dissertation manuscript, with a request to review chapters 4 and 5 (the Findings and Discussion

[3] Although data was ordinal, the SPSS software was not able to run non-parametric tests unless dependent variables were entered as "scales". It was verified that this was the correct procedure.

sections) and to indicate whether they felt that "my report is generally true to your own experiences and understanding of the area of educational software design and development." Up to this point, only two individuals have responded. Their responses are included in full in Appendix E: Member checking. As you can see, these individuals were satisfied with my interpretation of the data and gave only minor recommendations. All recommendations were incorporated into the final version.

# 4 Findings

The findings will be divided into four main areas, which cover the following topics:

1. *What it is like to work in "educational software design",* including how they choose to work in this area, what type of organization they work in and what type of software it develops, and what roles they play in their current position. This helps explain and contextualize the roles played by participants, addressing the first research question.
2. *Formal education,* including a general discussion of the types of degrees held by participants, and a comparison of participants from four major types of backgrounds.
3. *Skills and knowledge needed on the job.* Describes the types of skills and knowledge that are important to participants' current roles. Skills and knowledge unique to educational software design are highlighted in a final sub-section.
4. *Formal educational preparation for the job.* Summarizes the educational experiences of participants, and highlights gaps between what is important on-the-job and what was actually covered as part of formal educational programs.
5. *Recommendations for an ideal undergraduate program.* Discusses participants' recommendations for the "ideal" program to prepare students to work in this field.

Quotes provided in this section are derived from an analysis of both interview and open-ended survey data. Within the Findings section, you can identify phase 1 interviewee's comments because they are marked with an "I" followed by a number. For example, "I7" stands for the seventh Phase 1 interview participant. Survey responses are marked with an "S" followed by a number. For those that participated in a follow-up interview, comments typed into the survey itself are marked with an "S" (for example, "S73") while follow-up interview comments are marked with an "S" and followed by the word "interview" (for example, "S73 interview").

Appendix F: List of Participants includes a table which summarizes information about the nine phase 1 interviewees and 31 phase 2 survey respondents who were quoted directly within the body of the text as a reference to their characteristics for readers who want to place the quotes in context.

## 4.1 Working in "Educational Software Design"

In order to understand the roles played by software designers in this industry in context, I asked participants questions about why they choose to work in educational software design, as well as about their employer and roles they play on the job. These topics are discussed in the following sub-sections. Because the survey had a larger number of participants with a broader range of backgrounds, the majority of findings discussed here are from the survey, although interview findings as well as open-ended survey responses are used to illustrate interesting points.

### *4.1.1 Reasons for choosing to work in "Educational Software Design"*

Participants gave a range of reasons for choosing to work in this domain. Table 9 shows survey participates' responses to this question (closed-ended responses were based on themes discovered during analysis of Phase 1 interview transcripts). Participants could choose as many answers as they wished, and the majority had multiple reasons. As you can see, an ongoing interest in education is common among participants of all backgrounds. As one interviewee explained:

> I have always been interested in how computers can be used as tools for people to better understand their world and their environment. Whether as a communication medium or through groupware or through education. So, I've always had an interest in education. (I3)

Others saw a need for a particular type of software and began their own business, or began consulting for schools or other educational institutions. Participants felt that working in this area is very motivating because it makes them feel needed and valued.

> I'm very motivated to make a difference in education, and I'm not satisfied with making a difference in one classroom. I want to make a difference at a much higher level of impasse and one of the things that really attracts me about the position I'm in is there are 60,000 students taking the courses that I create… my team and that's very, very professionally satisfying to me. (I9)

Those who work in higher educational institutions in particular mentioned that they choose their current position because of the good working environment, which offers good hours, stability, and a chance to work in a "lifelong learning type environment" (I2).

**Table 9:** ***What caused you to begin working in educational software design/development?***

| | Percent | Count[a] |
|---|---|---|
| **Ongoing interest in education/educational support** | 68% | 56 |
| **Interesting design problems** | 52% | 43 |
| **Background in instructional design** | 30% | 25 |
| **Makes me feel needed/valued** | 29% | 24 |
| **Good working environment** | 21% | 17 |
| **Lots of experience in the educational software industry** | 16% | 13 |
| **Contacts in the industry** | 15% | 12 |
| **Just a job I found** | 13% | 11 |
| **Other (please specify)** | | 20 |

[a]n = 82. Participants were directed to "Choose all that apply"

Other reasons mentioned by individuals included enjoyment of the varied nature of the job and an attraction to the medium worked in, for those who began in education or other areas. For example, one interviewee who had an educational background in Classical Languages as well as Educational Technology explained:

> When PCs came along in the early 80s and offered the option of working on your own… not having to deal with these punch cards, but also being able to work on your own, at your own pace, work ahead if you needed to, seemed like a really attractive option to me. I guess I am more of an independent learner myself, which is why it did appeal to me." (I5)

Based on his experiences working on educational software as part of his graduate assistantship, he now runs his own software development company, focused on software supporting learning Latin, Greek, and related topics.

### *4.1.2 Organizations worked in.*

A set of questions addressed the types of organizations survey participants worked for and the roles they play on the job. Although this may not be proportionally representative of the field as a whole, I was pleased to see that companies of all types (Table 10) and sizes (Table 11) are represented, and that these companies develop educational software of many different types (Table 12). Participants are fairly evenly divided between educational software development companies, universities, and other types of organizations (either corporation that produce other types of products in addition to educational software, or educators developing software for use in their own classrooms or schools).

**Table 10:** ***Current Employment: Organization Type***

| | Percent | Count[a] |
|---|---|---|
| **Educational software development company which focuses on software used in higher education** | 8% | 7 |
| **Educational software development company which focuses on software used in k-12 school settings education** | 14% | 13 |
| **Educational software development company which creates software for use in multiple contexts** | 14% | 17[b] |
| **Department within a larger company which produces some educational or instructional software** | 11% | 8[c] |

| | | |
|---|---|---|
| **Department in a university which develops software primarily intended for use in a distance education program** | 4% | 4 |
| **Department in a university which develops other types of software for use at that university or across universities** | 12% | 11 |
| **Department, research center, or other group in a university which develops educational software for use in educational settings outside of the university** | 14% | 13 |
| **Other (please specify)[d]** | | |
| **University department (for own/student use?)** | 7% | 6 |
| **K-12 school** | 3% | 3 |
| **Works in multiple settings** | 3% | 3 |
| **Other** | 7% | 6 |

[a]n = 92, [b] Added one item from open-ended "other" field which fell into this category, [c] Added two items from open-ended "other" field which fell into this category, [d]Coded responses from open-ended "other" field

**Table 11:** ***Current Employment: Size of company/organization***

| | Percent | Count[a] |
|---|---|---|
| **Single-person business or independent contractor** | 13.0% | 12 |
| **Less than 5 employees** | 9.8% | 9 |
| **6 - 20 employees** | 10.9% | 10 |
| **21-99 employees** | 15.2% | 14 |
| **100 – 499 employees** | 13.0% | 12 |
| **500 or more employees** | 34.8% | 32 |
| **Don't Know** | 3.3% | 3 |

[a]N=92

**Table 12:** ***Current Employment: Type of Software Developed***

| | Percent | Count[a] |
|---|---|---|
| **Learning Management System (LMS) or Course Management System (CMS)** | 36% | 33 |
| **Components that reside in a LMS/CMS** | 37% | 34 |
| **Drill and practice application** | 26% | 24 |
| **Self-paced instruction (stand-alone application)** | 25% | 23 |
| **Self-paced instruction (web-based)** | 47% | 43 |

| | | |
|---|---|---|
| **Educational interactive simulation** | 37% | 34 |
| **Educational game** | 35% | 32 |
| **Software not used for educational/instructional purposes** | 25% | 23 |
| **Other (please specify)[b]** | | |
| **Tools for teachers (curriculum development and sharing, attendance, whole class instruction tools)** | 7% | 6 |
| **Assessment** | 3% | 3 |
| **Multimedia generation (podcasts, virtual images, "digital broadcasting"** | 3% | 3 |
| **Modeling tool** | 2% | 2 |
| **Software for specific platform (iPod/iPad/iPhone, interactive whiteboard)** | 2% | 2 |
| **Other (Augmentative reading support, inquiry environment, encyclopedic resource for vet training, blended learning solutions, etc)** | 5% | 5 |

[a]N=92; Multiple items could be chosen, [b] Coded open-ended responses to "other"

### *4.1.3 Current Employment: Formal title*

Survey participants were asked to enter their formal title. As you can see in Table 13, many hold management or executive management roles. This is likely in part because most participants are highly experienced and therefore far along in their careers. A review of the roles played by these individuals revealed that all of them continue to play technical roles, although they may be more likely to be involved in requirements gathering and higher level design tasks than day-to-day implementation. Faculty members who also do some software development are overrepresented in this survey (see section 5.6, Limitations).

A review of those who are in executive management or who own companies revealed that the majority of these work in very small companies. Of the two who indicated that they are "owners", one works in a single person business and the other in a business with less than five employees. Both play roles across the board including architecture, requirements, developer-level design, implementation, user experience design, and quality assurance. Of the 14 in "executive management", four work in a company with under 5 employees and four more work

in a company wtih under 20 employees. Six of them do not actually perform supervisory roles at all, and all play additional roles including requirements gathering (11 of 14), development level design (11 of 14), programming (6 of 14), user experience design (12 of 14) and instructional design (8 of 14).

**Table 13:** ***Formal Job Titles***

| | Percent | Count[a] |
|---|---|---|
| **Software Design (including Software Engineer, Designer/Developer, and Web Developer)** | 6% | 5 |
| **Lead Software Designer (with titles such as "Lead Developer" or "Senior Developer")** | 6% | 5 |
| **Project Management** | 2% | 2 |
| **Management** | 11% | 9 |
| **Executive Management (with roles such as "CIO" or "CEO")** | 17% | 14 |
| **Owner** | 2% | 2 |
| **ID/eLearning related** | 10% | 8 |
| **Faculty** | 27% | 22 |
| **k-12 teacher** | 5% | 4 |
| **Other** | 13% | 11 |

[a]N=82 who provide a formal job title in an open-ended field

### *4.1.4 Roles played.*

All but 11 participants played more than one role, and 66% (61 of 92 respondents) played more than 3 roles in their *current* position (see Figure 5). As you can see in Table 14, participants play a wide range of roles, from gathering user requirements and designing the high-level software architecture to low-level programming and database design. In addition, 47% do at least some instructional design work and 44% play a supervisory role. In the section "Four types of backgrounds", I will describe the variation between the types of roles played for participants with different educational backgrounds.

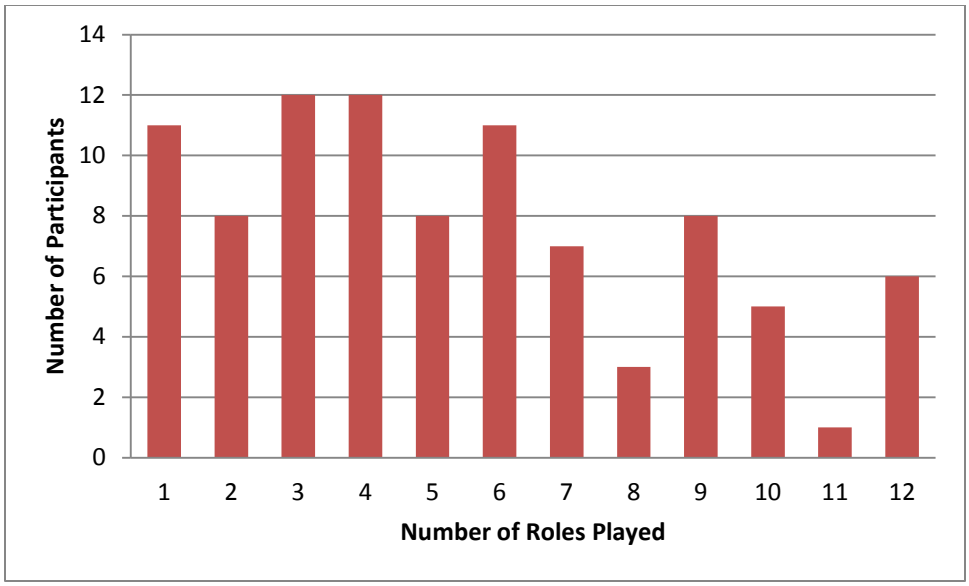



***Figure 5***Number of roles played by participants (N=86)

**Table 14:** ***Roles currently played***

| | Percent | Count[a] |
|---|---|---|
| **Software architecture** | 45% | 41 |
| **Business requirements gathering/generation** | 46% | 42 |
| **Technical requirements gathering/generation** | 47% | 43 |
| **High-level design** | 52% | 48 |
| **Low-level design** | 46% | 42 |
| **Programming** | 47% | 43 |
| **Database design** | 27% | 25 |
| **Web developer/Web designer** | 45% | 41 |
| **User Experience Design** | 53% | 49 |
| **Quality Assurance** | 27% | 25 |
| **Instructional Design** | 47% | 43 |
| **Supervisory** | 44% | 40 |
| **Other (please specify)*** | | |
| **Teacher/educator** | 9% | 8 |
| **Project management** | 2% | 2 |

| | | |
|---|---|---|
| **Other ("Creative Direction, Audio Design, Section 508 Compliance, Marketing, Sales"; "System support"; "teach, admin, student support, etc")** | 3% | 3 |

[a]N=92; Multiple items could be chosen.

## 4.2 Formal Educational Paths

Participants were asked to provide detailed information on degree programs they had attended. Unfortunately, this set of questions was apparently unclear to some participants, as a number indicated they had Doctoral or Masters degrees but listed no earlier degrees. Follow-up interviews with several of these individuals revealed that they had, in fact, taken multiple prior degrees. Therefore, the results reported in this section are not complete. I believe that participants generally *did* receive the degrees they indicate (as they were not inconsistent with other survey responses), but that they may also have received additional degrees.

The degrees held by participants varied quite a bit. Twelve percent (10 of 82) indicated they held at least one degree in both a computing-related and an education-related field. Fifty-four percent (44 of 82) had pursued at least one computing-related major (in areas such as Computer Science, Software Engineering, Information Systems, Human Computer Interaction, etc.). Of those remaining, 21% (8 of 38) had minored in, or had taken individual classes in, one of these areas. Twenty-nine percent (24 of 82) held at least one major in the areas of Instructional Design, Educational Technology, or another education-related field. Of those that remained, 41% (24 of 58) had taken a minor or some individual courses in this area. Only 16% (13 of 82) appear not to have taken any formal coursework in computing-, instructional design-, or education-related areas (This number is based on responses to questions about specific types of course taken as well as the formal degrees held. Therefore, this number should be trustworthy). Participants had also earned degrees in physical sciences (e.g. Physics, Biology,

Chemistry, Astronomy, etc.), social sciences (Psychology, Sociology, Anthropology, etc), art (e.g. Fine Arts, Graphic Art), Business, Philosophy, Law, and a variety of other areas.

Interviewees also had a range of educational backgrounds, from no degree to several with doctoral degrees. Most held multiple degrees, with an eclectic mix of majors. Across the 9 Phase 1 participants, three had one or more degrees in Computer Science or another computing field, three had one or more degrees in instructional design or educational technology, and three had at least one education degree or teaching certificate. Other degrees held included Psychology, the Classics (Latin and Greek), Physics, English, and Information Systems. Each interviewee who did not have a formal degree in computing or instructional design indicated that his own educational experience was extremely valuable in fostering critical thinking and other skills needed on the job. For example, one explained "A science degree ...it is important to think systematically, and avoid the subjective needs of the people making the demands on what software you need." (I1). Another similarly mentioned that his English degree helped him learn to think critically and communicate clearly, enabling him to solve the types of problems he encountered on the job.

### *4.2.1 Four Types of Backgrounds*

As was mentioned in the previous section, software designers in this area have a wide range of educational backgrounds. In order to explore the connection between their backgrounds and other factors, participants were asked the following questions:

1. Did you take any courses related to software design or development as part of your formal education?
2. Did you take any courses related to education or instructional design as part of your formal education?

Based on these questions, participants were categorized into four groups, as follows:

1. Those with some computing-related formal education, but no coursework in ID or education: These students had had some formal education in Computer Science, Software Engineering, or a related field. For the purposes of this report, any participant who answered "yes" to question 1 and "no" to question 2 was assigned to the group "computing only".
2. Those with some formal education in instructional design, educational technology, or other education-related field (since the numbers were relatively low, those with degrees in any education related field were combined with those with an instructional design or related degree), but no coursework in Computing: Any participant who answered "no" to question 1 and "yes" to question 2 was assigned to the group "ID or education only".
3. Those who had done at least some course-work in each of these areas : Anybody who answered "yes" to both questions 1 and 2 was assigned to the group "Both".
4. Those who had done coursework in neither of these areas: These participants were assigned to the group "Neither". Please note that all but one participant had at least one post-secondary degree and many had multiple degrees, so participants in the "Neither" group did have formal education in one or more other areas, as reported in the section "Formal Educational Paths."

In this section, I will discuss how these backgrounds relate to participants' experiences and beliefs relating to the roles they play in educational software design. In later sections, I will explore the degree to which each type of background appears to have prepared participants for the roles they play.

### *4.2.2 Experience in Software Design and Instructional Design*

Perhaps unsurprisingly, those with an educational background in a computing-related area have more experience in software design than do those without this background. As you can see in Table 15, 68% of those with an educational background in computing but no coursework in education, and a similar number of those with formal education in both areas, have more than 10 years of experience in computing, while only 30.8% of those with ID or education related coursework alone have this level of experience. Since all participants are currently software designers, all have at least some experience in this area.

Conversely, those with no coursework in education and instructional design have little or no experience in instructional design, as can be seen in

Table 16. Fifty-six percent of those with computing education only, and 25% of those with no background in either area, have no experience in instructional design at all. Still, it is noteworthy that many that have no formal education in instructional design (16% of those with computing backgrounds and a third with no degree in either field) are experienced instructional designers (with more than ten years of experience).

Perhaps most interestingly, those who have done at least some formal coursework in both areas are most likely to be highly experienced (with more than 20 years of experience), possibly indicating that people who remain in this industry have pursued formal coursework in both areas over time. Those with no formal background in either area tend to be the most eclectic, with no clear pattern to their experience levels.

**Table 15:** ***Experience levels in Software Designers as measured by years of experience (n=74) in computing, ID/education, both, and neither***

| | Computing only (N=25) | ID or Education Only (N=13) | Both (N=24) | Neither (N=12) |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| **Less than 1 year** | 4.0% | 15.4% | 0.0% | 16.7% |
| **1-4 years** | 12.0% | 30.8% | 4.2% | 0.0% |
| **5-10 years** | 16.0% | 23.1% | 29.2% | 16.7% |
| **11-15 years** | 24.0% | 15.4% | 8.3% | 33.3% |
| **16-20 years** | 8.0% | 7.7% | 16.7% | 16.7% |
| **21+ years** | 36.0% | 7.7% | 41.7% | 16.7% |

**Table 16:** ***Experience levels in Instructional Design as measured by years of experience (n=74) in computing, ID/education, both, and neither***

| | **Computing only (N=25)** | **ID or Education Only (N=13)** | **Both (N=24)** | **Neither (N=12)** |
|---|---|---|---|---|
| **None** | 56.0% | 0.0% | 0.0% | 25.0% |
| **Less than 1 year** | 8.0% | 7.7% | 12.5% | 0.0% |
| **1-4 years** | 16.0% | 23.1% | 12.5% | 16.7% |
| **5-10 years** | 4.0% | 23.1% | 20.8% | 25.0% |
| **11-15 years** | 8.0% | 23.1% | 16.7% | 0.0% |
| **16-20 years** | 0.0% | 15.4% | 12.5% | 8.3% |
| **21+ years** | 8.0% | 7.7% | 25.0% | 25.0% |

As Table 17 shows, participants with different backgrounds may be more or less likely to fill certain types of roles. Those with formal coursework in computing related areas are more likely to be directly involved in the implementation of software, as well as the more technical ends of designing the software. Both survey and interview participants indicated that they had played additional roles over the course of their careers. However, participants who either had education in computing fields or were self-trained seemed more confident in their ability to pick up new programming languages and technologies than those with a background in ID or

education only. During member checking, a participant with a background in computing remarked on this finding:

> The word 'seemed' is a rather weak word. I would say that most people who have computing degrees or self-trained were sure that they could pick up a new programming language. Nearly all programming languages are the same it is just syntax and APIs that make them different. (S56, member checking)

**Table 17:** ***Current Employment: Roles Played***

| | Computing only (N=27) | ID/Education only (N=13) | Both (N=24) | Neither (N=12) |
|---|---|---|---|---|
| **Product-Level Design** | | | | |
| **Software architecture** | 74.1% | 18.8% | 34.6% | 46.2% |
| **Business requirements gathering/generation** | 66.7% | 50.0% | 34.6% | 30.8% |
| **Technical requirements gathering/generation** | 74.1% | 43.8% | 38.5% | 23.1% |
| **Design of Data Models, Algorithms, etc.** | | | | |
| **High-level design** | 70.4% | 50.0% | 53.8% | 38.5% |
| **Low-level design** | 63.0% | 50.0% | 42.3% | 23.1% |
| **Implementation** | | | | |
| **Programming** | 70.4% | 31.3% | 38.5% | 38.5% |
| **Database design** | 44.4% | 12.5% | 26.9% | 23.1% |
| **Web developer/Web designer** | 51.9% | 43.8% | 53.8% | 30.8% |
| **Roles related to other areas of the overall product design** | | | | |
| **User Experience Design** | 59.3% | 56.3% | 53.8% | 53.8% |
| **Quality Assurance** | 22.2% | 18.8% | 38.5% | 30.8% |
| **Instructional Design** | 18.5% | 87.5% | 73.1% | 30.8% |
| **Supervisory** | 29.6% | 56.3% | 46.2% | 69.2% |

*Note*. Multiple items could be chosen. Participants were directed to choose all that applied.

Pairwise Pearson Chi-square tests were conducted to determine whether educational background had a significant impact on roles played on-the-job. That is, separate tests were run

for each possible pair of course types (Computing (Computing-related courses only) versus ID/Ed (Instructional Design or Education related courses only); Computing versus Both (at least one of each type of course); Computing versus Neither; ID/Ed versus Both; ID/Ed versus Neither; and Both versus Neither). The statistically significant results are displayed in Table 18.

Those with a formal background in Computing are much more likely to play technical roles, including Software Architecture and Technical Requirements generation as well as development roles including Programming and Database design. Unsurprisingly, those with a background in Instructional design are more likely to play an instructional design role than others. More interestingly, those who have no formal education in either area are the only group statistically significantly more likely than another group to have supervisory roles, although they are less likely to participate in technical requirements gathering, high-level design, and low-level design. It is interesting to note visually that in many areas areas "both" is identical to "ID/Education", and there are no significant differences between these two groups.

**Table 18:** ***Differences in likelihood of playing various roles, based on educational background***

| Roles and Groups compared | Chi-square | Effect size (Phi) |
|---|---|---|
| **Software Architecture** | | |
| **Computing more likely than ID/Ed** | 12.711*** | Large (-0.578) |
| **Computing more likely than both** | 9.010** | Medium (-0.429) |
| **Business Requirements** | | |
| **Computing more likely than both** | 4.608* | Medium (-0.307) |
| **Technical Requirements** | | |
| **Computing more likely than ID/Ed** | 4.027* | Medium (-0.326) |
| **Computing more likely than both** | 5.891* | Medium (-0.347) |
| **Computing more likely than neither** | 7.298** | Medium (-0.444) |
| **High-level design** | | |

| | | |
|---|---|---|
| **Computing more likely than neither** | 3.970* | Medium (-0.328) |
| **Low-level design** | | |
| **Computing more likely than neither** | 4.937* | Medium (-0.365) |
| **Programing** | | |
| **Computing more likely than ID/Ed** | 5.964* | Medium (-0.396) |
| **Computing more likely than both** | 4.601* | Medium (-0.306) |
| **Database design** | | |
| **Computing more likely than ID/Ed** | 5.218* | Medium (-0.371) |
| **Instructional Design** | | |
| **ID/ed more likely than Computing** | 18.087*** | High (0.690) |
| **Both more likely than Computing** | 12.790*** | High (0.511) |
| **ID/ed more likely than neither** | 9.420** | High (-9.614) |
| **Both more likely than neither** | 4.629* | Medium (-0.359) |
| **Supervisory** | | |
| **Neither more likely than Computing** | 5.029* | Medium (0.369) |

*Note:* Only comparisons that were statistically significant were shown.

ID/ed: At least one instructional design or education-related course taken as part of a formal degree program (no Computing courses). Computing : At least one computing-related course taken as part of a formal degree program (no ID/ed courses). Both: At least one of each type of course taken. Neither:No ID/ed or Computing courses taken in any degree program attended.
*$p$<0.05, ** $p$<0.01, ***$p$<0.001

An interesting area to highlight was the remarkable similarity in focus on user experience design (59.3% of those with a computing background, 56.3% of those with an instructional design or education background, and 53.8% of those with formal education in either both or neither of these areas indicated this is a role they currently play), an area interviewees stressed as very important, yet often difficult to "sell" to co-workers and clients, especially internal clients. As one explained:

> In the larger higher-ed [sic] community, there is [a] huge push towards usability. It's really a sea-change, where a lot of the software that we are working on had

> traditionally been developed and driven really from an engineering mind-set. Now there was a new emphasis on designing the software from the standpoint of the user, for taking the user need and making that the point of entry for a project, and working on clearly articulating that user need and designing around it. (I3)

He indicated that when he began his current role he considered himself an "evangelist" for user experience design, but he and his group had grown since then as the university organization for which he works began to understand the value of this area.

## 4.3 Skills and Knowledge needed on the Job

Interview participants identified a range of important skills, knowledge, and attitudes necessary to succeed in Software Design positions. The most commonly mentioned areas are summarized in the sub-sections below. Although the technical skills are discussed first, it is interesting to note that the largest numbers of comments had to do with communication skills, a focus on the needs of the user, and topics discussed in the "design judgment" section (particularly as relates to structuring software appropriately so that it can easily be maintained and adapted over time).

### *4.3.1 Playing different roles*

As described earlier, participants play many roles, and being "well-rounded" is considered an important trait. One interviewee indicated that he particularly enjoyed wearing many hats, but "it [is] harder these days to be a jack of all trades…one really has to focus, because there's so much learn, so much to know"(I3). This is why many teams have a balance of skills, often including user experience designers, graphic artists, and other types of media specialists in addition to content experts and developers. As they progress in their careers, participants have entered management roles. This can be a mixed blessing. "I love to do the

design so it's always a bit of a disappointment when I realize I won't have time" (s43 interview). However, playing the role of a mentor can be rewarding to more experienced professionals.

### *4.3.2 Technical Skills and Knowledge*

Along with the ability to program, software designers involved with development must keep up with new platforms and be able to ensure that new technologies will work with other business systems. They must know how to structure their code appropriately, so that it is both efficient and modular enough to be maintained over time (this is discussed further in the section on "design judgement"). Participants explained that different skills are needed for different types of programming (for example, "front-end programming" using Javascript, versus back-end programming or database design; scripted languages have their own advantages (such as the ability to easily view and learn from other's work)). Code reviews are an important part of the job, and are "part of a quality production environment." (s43 interview)

Those without a technical background may have new things to learn, even if they are not directly involved in software development. For example, as one participant explained, employees with a degree in an area related to education (other than instructional design) can have difficulty understanding the need to make software scalable.

> I mean, I would also say that a degree in education doesn't hurt, but it is not adequate in and of itself. And the main reason for that is that it is, it's so important that you can scale what you do. And I am constantly working with people who don't understand scalability issues. And I'm always having to teach them – re-teach them – how to approach instructional problems. Because… it has to work for 60,000 people in 50 states, who might be by themselves at home. (I9)

### *4.3.3 User experience design, visual design and usability related Skills and knowledge*

One participant explained that he has worked hard to "evangelize" on the importance of user experience design. In his current position on a team at a university, he now leads a team of user experience designers where previously no-one had paid explicit attention to this aspect of the work. As a user experience designer, he feels it is important to understand "the visual design aspect."

> I run into way too many user experience designers who actually do not understand visual design. It's not so much that you have to be able to do it, but that you have to understand visual media… that's part of your communication, that's part of the pallet that you use as a user experience designer. (I 3)

One interviewee explained that the idea of focusing on a user friendly design is not a new idea.

> User friendly software… was a common expression back in the 80s, I guess, and I don't know how long it lasted, but it was to really just concentrated on making the interface as personable as possible, easy for anybody off the street, not just a computer geek, could understand how the program was structured and how to find your way around, and find what you wanted, what you needed, and get some kind of useful feedback form the program. (I5)

He believes that this is still an important goal although some of the terminology has changed, but he is concerned that often decisions are being made for the purpose of novelty rather than for sound design reasons.

> We keep playing around with margin menus and task-bars and organizing search fields and where to put everything, and that's a constant process, I guess, but it's

> interesting to see how designers will land on one theory for a while, and then decide to change it 5 years later, sometimes just to be new and different and show that they're being new and different just to change it, whether it's good or not, just to shake things up for their users and not let things get stale. That seems to be perhaps more of a priority than analytically sound, simple, easy-to-read design that sometimes they just want to pick things up and make it look new and different just for the sake of that alone. (I5)

Other specific concerns come into play for certain types of users and contexts. In some cases, it is very important for the software tool to remain in the background, so that the focus is on the content. For example, in the case of lecture capture software, it is important to understand "that teaching, especially a lecture, is a performance. It's a stage performance. And that means that the tool has to kind of be in the background. It has to be easy to use, and that's pretty much it." (I2). Another lesson is the importance of keeping the user's attention, as it is "commonly accepted, I guess, that … the general public attention span seems to shorten on average all the time, and we need more activity, more motion, more…. Introduction of things new and different to keep us interested and attentive." (I5) Others discuss the importance of understanding specific visual media, and the of "being aware of the relationship between media and text, and how they reinforce each other" (I9).

#### *4.3.4 Management and Project Management related skills*

As discussed earlier, quite a few survey participants play supervisory roles. Even for those without a formal title of manager or project manager, managing staff, timelines, and other aspects of a complex project may be an aspect of working in this field, especially for more experienced professionals. Teams often include members from a variety of backgrounds and

roles. "Typically three to five teachers, at least two content reviewers, sometimes three, and then on the tech team, um, videographers and the flash programmers, typically we would have two to three flash programmers, three to five videographers, an artist, that's pretty much the team" (I6). If contractors are hired, budgets and timelines need to be negotiated.

One interviewee explained that skill that make a good project manager are similar to those that are needed to be a good instructional designer. "you have to be a good planner, problem solver, communicator, researcher, writer. What you have to add to it is the ability to lead and manage other people effectively" (S 43 interview). She went on to explain that some people, especially those with a computer science background in her experience, have problems with another important aspect of project management: managing deadlines and realistically budgeting time.

> Where many CS people fall down is in wanting to over-engineer everything. They want to build a state-of-the-art LCMS when a simple Moodle-based solution will be just fine – and where the budget offers no other choice. This is where instructional design and common sense help in reigning in the grand schemes of some folks. I also find that CS types greatly underestimate their time requirements – so it's a strange mix of over-specifying the solution and underestimating their time. (S 43 interview)

Those with a background in education may have other types of lessons to learn. "You tend to try to perpetuate things that make you successful in a classroom, and those aren't the types of behaviors that are important to doing in admin or in coordinating different projects" (I 6).

In his experience as a one-person company, another participant warned of an issue that is especially common for new enthusiastic designers.

> I have discovered over the years that if you attempt to produce the perfect program then you will never finish. Version 1.0 of an App needs to do something useful but does not need to do everything it could do. (survey 56 interview)

He went on to explain that once something is selling, he is likely to get feedback on what users *actually* want, which can then be added to a later version of the project. However, it is important to note that different business models may apply to different types of software. For example, one interviewee explained that web-based software is often sold on an advertising or subscription-based model, in contrast to the business model he is used to "which is more like selling books or any other educational materials" (I5).

#### *4.3.5 Communication and Team Skills*

Communication skills mentioned include verbal, written, and presentation skills. "I think …the one skill that has transferred throughout my positions [is] learning to talk with people" (I 8). The importance of effectively communicating a design, breaking up information in order to communicate it clearly, and an understanding of the relationship between media and text were all mentioned as important related skills.

Interpersonal skills are important because collaboration is such a key part of work in this area. They include the ability to have a back-and-forth discussion about a design or work plan, the ability to truly *listen* to others, and the ability to understand how others on the project think.

> And that ability to collaborate with other developers is SO important. I mean, for me, I will take a collaborative programmer that is not as good as a lone wolf, because I KNOW that in the end he will make fewer mistakes, because he collaborates. And, it reduces my risk, because if something goes wrong, two

> people know how to fix it, not one. And so you know, that is a really important aspect that doesn't seem to really be being emphasized, to be able to TRULY work in a team and write code with a team. (I 9)

Someone in this role needs to be able to adapt to working on a large team, working with remotely located team-members, and working on multi-disciplinary teams which may include people with expertise in a wide variety of media and content areas as well as software designers and instructional designers. It is important to realize that instructional designers have some areas of expertise while technical team members have other areas of expertise, and to understand how other team members work. However, sometimes personality issues play a bigger role in communication than the specific area of expertise.

> In terms of… for example, how easy is it to extract the information that you need, how easy it is to work with the person, how quickly can you accomplish your objective, that really depends a lot more on the person, and how easy that person is to work with. (I4)

#### *4.3.6 Design Judgment*

A fair number of participants discussed the ways that their own judgment developed over time. As one interviewee explained, "everything goes into the design and development process. There is no one approach and you build up judgment in design and development from every project that you are a part of" (I8). Judgment calls are important in technical roles. For example, participants mentioned the importance of having "good programing sense" and "good usability sense". They also play a role in efficiently learning new things needed on the job.

> Well, I guess you can call it professional judgment. After doing millions of data searches and millions and millions of searches of info [sic], you kind of learn to

> judge the quality of the info you get, and usually if it makes sense to me, it usually has some kind of merit to it. If it doesn't make sense to me, it is usually questionable material. I think I am at the point where, if the material is good, it will make some kind of sense to me. I should at least be able to connect what is the logic involved here/ the meaning to this. (I 8)

Participants mentioned other lessons that they have found are important in guiding their decision making in complex design situations, including knowing that there is more than one way to do something, understanding the importance of a good design to start with on following steps and final solution, and knowing when to use existing code and when to write new code. All of these concepts play a role in the crucial decisions relating to how to structure a program so that it will be both efficient and maintainable.

Experienced designers can tell a good design when they see one. "A lot of what I know is obviously from experience. I can look at a design and point out classes that are just a complete waste of time and should be made part of another class" (s56 interview). Another participant indicted that it is important to structure code in a way that will be maintainable over time:

> The way I approach software development, is to try to imagine where I want to be in five versions, and then start scaling it back , but I do that so I can make sure we don't , like I said, code ourselves into a corner. (I 2)

Although he is involved in the design and not development of the system and has not personally done much programming he indicated that he can tell a good programmers from a poor or novice programmers because experienced programmers know how to structure their code in a way that takes into account future needs:

> Because some day, your… code is going to be called upon by some new feature or some new back-end system. And it takes a little longer and you have to be a little more careful to write it that way. And you know, junior programmers don't get that and don't write it that way. And they are constantly rewriting our stuff. Where experienced programmers don't. (I2)

He went on to explain that this type of insight is more important in a developer than grounding in the domain area the software is being developed for, as an experienced programmer will be able to point out things that a content expert may not have thought of.

#### *4.3.7 Understand contexts and users*

It is important to understand who a client is and what the client wants.

> One of the most important thing about designing applications or content is knowing who your client and what their expectations are, so you can tailor everything you do for their needs and to use that to help create something that really works for them. (I 8)

However, the "client" for educational software is often not the end-user. Participants warned that it is very important to gain an understanding of the actual end-user, as a client or the "middle-men" may not know what the real end-user actually wants and needs. However, clients or users may not really know what they need.

> Coming to a really good and deep understanding of what your client wants and needs is a very important skills for software developers, and especially consultants like myself. And it's not always an easy process, because in many cases, the person that you are doing this work for only has a vague idea of what they really want to happen…all they know is that they are not happy and something is going wrong. And it falls to me and people like me to me and

> people like me to figure out what it is that they really want, and what they don't want, ok? Because it has an impact on the nature of the work that I do. (I7)

For example:

> So if they want a program to, um…oh, I don't know, help to grade papers, for example. Papers are submitted through email, and they want help grading it, and… they say "I want to grade these papers". They give me a list of criteria or rubrics to use. Then I'll start asking questions. Like, "what kind of grades do you assign? Do you use point grades or letter grades?" They'll tell me, "oh, its only points, I don't care about letter grades at all. So, that's a requirement that gets translated into this job that I'm building. And being able to… talk with clients and understand their true needs is a very important skill. (I7)

Unfortunately, this is not always easy to do.

> There's two obstacles … We've got too many people between us and the students, and I think the students don't know what they want either. This year it is like google, and the previous time something else. And by the time the system is out there it would probably be not what they want. So it's basically really tricky. (I1)

As another explained, in the area of higher education the people between himself and his actual users (students in higher education), may also have little insight into what works for those users.

> I think it's important that when people design systems, it's as near to what the end-user wants, rather than the people that are designing the system. There [are] a lot of people in the middle, from the faculties of whatever, who tend to be 20 [or more years] older than the target audience. (I1)

Finally, the software itself and the related training must be provided in a form that is appropriate for his users and their context. As another participant explained, it is important to "make sure that we provide resources that they can best utilize, and provide training to use those resources effectively." (I 6) This participant, who provides resources for k-12 schools across a specific region, regularly surveys and holds meetings with the administrators and library media teachers who are his clients, as well as getting feedback from colleagues from other regions.

#### *4.3.8 Need for Self-learning*

Self-learning is considered a very important aspect of the job in this field. "I think software developers are one of the most susceptible professions to obsolescence unless you really take control of your own learning." (I 9) As will be discussed further in the later section on self-learning, the way individuals go about learning may be very individualized, but typically is in response to a direct on-the-job need.

> So, In terms of my self-taught education, I would learn things as needed. So, if a project required a new language, I just decided that that's what I was going to learn. I didn't really set out on a particular trajectory and say, "Oh, ok, well, now I will really have to learn language X because language X was cool." It was always from a standpoint of, I need this for a project, so I'm going to learn it. (I 3)

However, the willingness to acquire skills one is not comfortable with is also very important.

> I don't have a huge belief in talent. I believe much more in skill and that skill is something you can acquire. What may be innate is the willingness to acquire those different skills, and I am not sure I know where that comes from. But that… for example, I myself know I am not a really strong visual designer, but I actually manage to do reasonably good visual design just from having studied it,

> from taking the time to take some color theory classes and take some typography classes, because I know that was my weakness. (I 3)

#### *4.3.9 Other things to be prepared for*

Individuals mentioned a number of other "life lessons", a few of which are mentioned here. As described earlier, one interviewee made sure that he always "starts small" on new software (specifically "apps" for iPhones, iPods, and iPads), as not every design will sell. Once it has sold and users start providing feedback, additional features can always be added. Another pointed out that unexpected changes to the design or the plan can occur in any project, "because once you start putting it together, you come up with other things you haven't through of. Almost invariably." (I 2) Finally, participants have experienced a variety of projects of different types, requiring different skill-sets, over time. One participant had worked in an especially wide range of positions:

> I've worked with educational publishers to do curriculum projects for middle, high school, and higher ed. Subject areas have included foreign language, developmental math, college biology, middle school history and social studies, economics, English grammar and more than I can list here. I've probably done several dozen educational products. I've also worked with companies to develop training materials for customers (before and after sale), and for employees. And, CEUs, which is training for professionals who need on-going training to maintain a license or accreditation. …I've done a lot of software and microprocessor kinds of training. Also, utility companies, oil industry, non-profits, state agencies, call centers – again, the gamut. I think if you have a long career in instructional design, this is pretty typical." (S43 interview)

### *4.3.10 Skills and Knowledge Especially Important for working on Educational Software Design*

When survey participants were asked what is unique about working in educational software design (as opposed to designing other types of software), participants across the four groups responded rather differently, as Table 19 demonstrates.

**Table 19:** ***Skills and Knowledge unique to developing educational software***

| | All Groups (N=89) | Computing only (N=27) | ID/ Education only (N=16) | Both (N=26) | Neither (N=13) |
|---|---|---|---|---|---|
| **Not unique – this job requires the same skills that I would need to gather requirements and develop software for any specialized group of users** | 23% | 44% | 6% | 7% | 31% |
| **Not Unique – I know this content well because I have been working with it for an extended period of time, but I would have learned about any domain after having worked in it for the same amount of time** | 17% | 30% | 6% | 4% | 31% |
| **Educational theory** | 52% | 22% | 69% | 85% | 39% |
| **Instructional theory** | 49% | 15% | 69% | 81% | 31% |
| **Instructional design experience** | 42% | 11% | 75% | 54% | 39% |
| **The ability to educate clients and co-workers on topics that are not well known or understood in this field** | 36% | 19% | 50% | 35% | 46% |
| **Other** | 24% | 22% | 25% | 27% | 15% |

*Note*. Multiple items could be chosen. Participants were directed to choose all that applied.

Those with formal experience in Instructional Design or education (including those with experience in both areas) were more likely to feel that educational and instructional design theory were important to their role, and more likely to have benefitted from instructional design experience (although, as mentioned earlier, they were also more likely to have had instructional design experience). As one survey participant explained in a follow-up interview, "I feel that when the project is an instructional project, the manager needs to have an instructional

understanding of what success will look like – not cool graphics or zippy performance, but instructional effectiveness." (S43, interview). An interviewee alluded to the difficulty of understanding how to make software engaging to students as well as targeting a student's specific learning needs "without providing more info about other things that a student doesn't necessarily need to know about or already knows." (I6)

Those with formal education in computing only were much less likely to feel that the domain-specific content is unique – rather, they feel that gathering user requirements is a part of any job in software design. The options presented for these closed-ended questions were based on the data collected during the interviews.

As one interview participant explained, an individuals' ability to communicate with users and subject matter experts is more important than a background in a specific domain:

> In terms of…how easy is it to extract the information that you need, how easy it is to work with the person, how quickly can you accomplish your objective, that really depends a lot more on the person, and how easy that person is to work with, and do they have experience with this type of thing, in terms of a software development project, that has much more weight than if it was in particular instructional design versus something in firefighting or whatever [sic]. (I4)

Others explained that they had developed experience over time, but this is something they would have done in any industry they would have worked in. One suggested that domain-specific knowledge is more important for some types of software than others.

> The specific [content] areas I don't need to know very much at all. …[The software we develop] is designed to… record a lecture as it's being given. And so…the level of expertise I that needed to really have is, what are they trying to

> do, what is important to capture, and what is important for students to have access to. Understanding that teaching, especially a lecture, is a performance. It's a stage performance. And that means that the tool has to kind of be in the background. It has to be easy to use. And…that's pretty much it. (I2)

He went on to explain that among programmers "the easiest to work with are the programmers that have the most experience with great user interfaces, understanding how people actually use software. Not specific to the education realm, in my case." (I2) Another explained that the personality of a subject matter expert one is working with makes a much bigger difference to his ability to determine requirements and build a good system than the specific domain one is working in.

> The personality component has way more to do with it than the subject. For example, how easy is it to extract the information that you need, how easy it is to work with the person, how quickly can you accomplish your objective, that really depends a lot more on the person, and how easy that person is to work with, and do they have experience with this type of thing, in terms of a software development project, that has much more weight than if it was in, you know, in particular instructional design vs something in firefighting or whatever. (I4)

About a quarter of survey participants described additional skills and knowledge that were unique requirements for a job in educational software design. These comments were remarkably similar across all four groups, and included teaching experience, understanding specific aspects of users and contexts unique to this area (such as student characteristics, classroom realities, time constraints when collaborating, knowledge of cognitive science and child development, and understanding measurement and assessment) and content-specific

knowledge (e.g. math education). Contact with subject matter experts in the field of education was offered as a valuable resource. Other survey participants described personal characteristics they felt were necessary to work in this field, such as "Passion for helping kids" and "[being an] expert learner with ability to see patterns in new knowledge domains, knowledge and skill elicitation from subject-matter experts." Another indicated this area is "unique because everyone in the process is a potential user."

Similar sentiments were offered by interviewees. One indicated an important factor in his role was the ability to "[understand] the needs of classroom teachers, and what resources they use in their classroom environment. Make sure that we provide resources that they can best utilize, and provide training to use those resources effectively." (I6)

Another interviewee mentioned several additional reasons that domain-specific knowledge is important. Because he is involved in designing an authoring system used by his team of instructional designers,

> …the authoring system has to afford best practices in language acquisition so I can't sort of bring someone in to sit in a room and say "yeah, that is a good idea" or "no, that's not a good idea" because there's so many other factors that have to be considered that are highly technical, you know, regarding reusability issues, and scalability…and so, if I don't have that in my head, I'm not going to make the right decisions, or at least not optimal decisions, about its structure. So that is one major reason I need to know it. (I9)

In addition, he felt that it is important to have, "face validity": letting clients and employees know that you understand the domain.

> When people meet me and they find out what my job is, and then figure out that I don't speak 5 languages, then I need to be able to portray to them enough expert knowledge that they will lend me some credibility in my position. (I9)

Interviewees also pointed out the need to educate clients on topics that might not come up in other domains. For example, one described the need to help k-12 schools understand that they had a "brand" and a character to portray on their web-sites in a previous position. They did not have much familiarity with this medium, "and yet they understood that they really needed a website. So, a big portion of my work was really educating them in how they wanted to use this medium" (I3). In his current position in a higher educational setting, he has found that he has had to serve as an evangelist for the importance of user experience design. Understanding common practices in the domain is also important for determining a business model. For example,

> One of the key decisions we had to make… how do we license [our software] and how do we price it. And one of the things I was very adamant about was that you had to price it such that an individual instructor could buy it. Because I was aware that, at least in the higher ed level, when a faculty makes a software request, it can be all kinds of rings of hell to jump through to get somebody to purchase it, and often if it's priced right, they'll just buy it themselves, without having to go through the bureaucracy. (I2)

Other important areas mentioned by interviewees included the importance of maintaining proficiency in developing for multiple platforms, which is sometimes more important in education than in other domains, synchronization of software packages with related textbooks, and the value of working closely with professional organizations in the specific domain.

## 4.4 Formal Educational Preparation for the Job

### *4.4.1 Computing Related Courses.*

When asked to describe the most important things they learned from computing-related courses, interviewees indicated that "software design fundamentals" (such as Object-oriented design) were important. Beyond these, learning to think in a systematic way and learning to think "outside of the box" were more important than other specific concepts. Learning how to interact with clients (e.g. from a systems analysis course) and how to work in small groups was especially valuable –for those who had these experiences.

Figure 6 lists types of courses that are typically identified as important components of computing-related majors. The charts to the right indicate survey participants' views on the degree to which each of these has been important across their professional careers. As you can see, participants generally agree that basic theoretical courses and foundational concepts such as Object Oriented Design are quite important, while other areas have been less important in their careers.

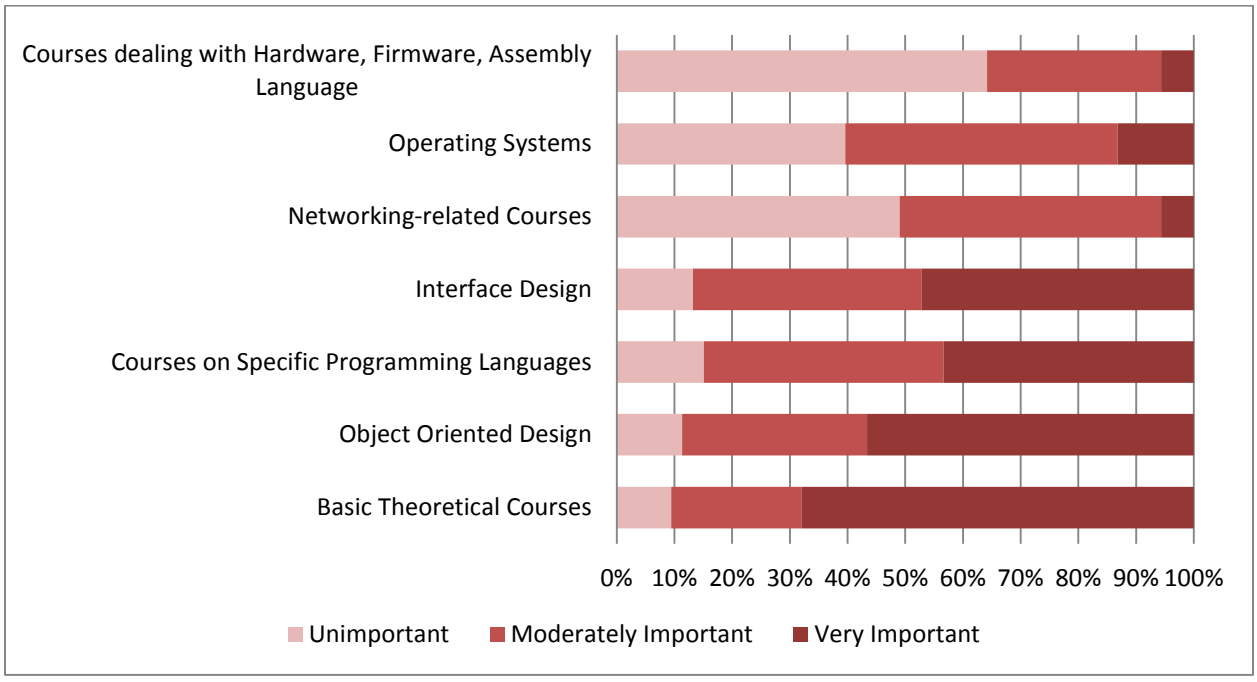


***Figure 6*** Importance of computing-related courses and concepts on the job (N=53 who responded to this question set)

Figure 7 lists a number of areas that are identified in literature and in the Phase 1 interviews as being important to software design practitioners in the field. Each bar shows the degree to which participants felt this topic was covered during their formal education. Few participants felt that these important areas were covered "a great deal" in their university courses.

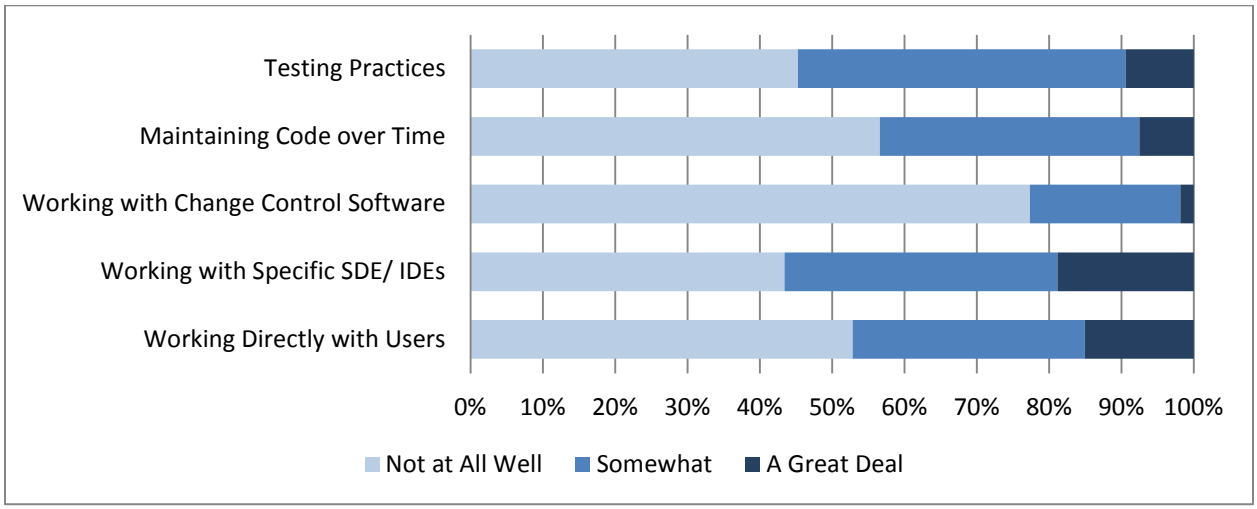



***Figure 7*** Degree to which topics were covered in computing courses (N=53 who responded to this question set)

For the most part, there were no statistically significant differences between participants who had taken Instructional Design or education related courses in addition to computing related classes and those in the other three groups on these items. However, those with at least one course in each area were more likely to indicate that they were prepared to "work directly with users" ($U = 451.0$, $p<0.05$, see Figure 8) and "testing practices" ($U = 454.5$, $p<0.05$, see Figure 9), based on an Independent Samples Mann-Whitney U test). It is somewhat surprising that "testing practices" were better covered in Instructional Design courses; although the researcher intended this question to cover software quality assurance related testing, it could be that

participants interpreted this question more broadly to consider types evaluation techniques learned in Instructional Design coursework.

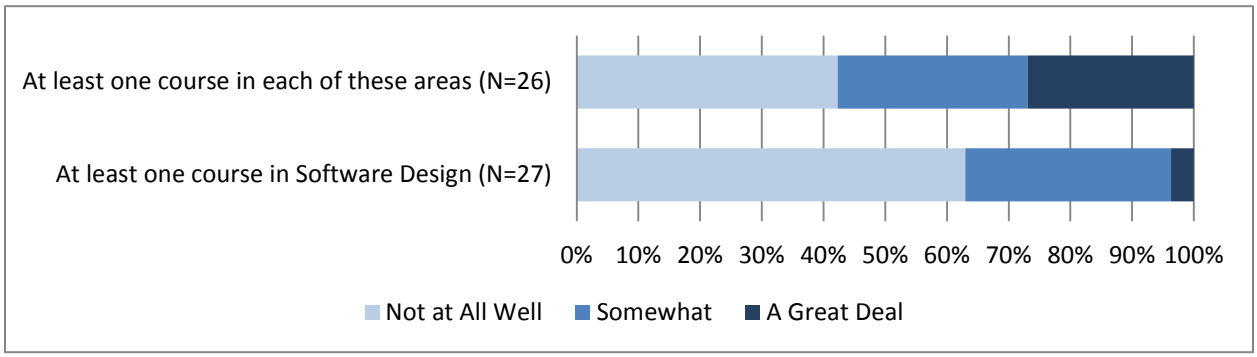



***Figure 8*** Degree to which coursework prepared you for "Working with Users".

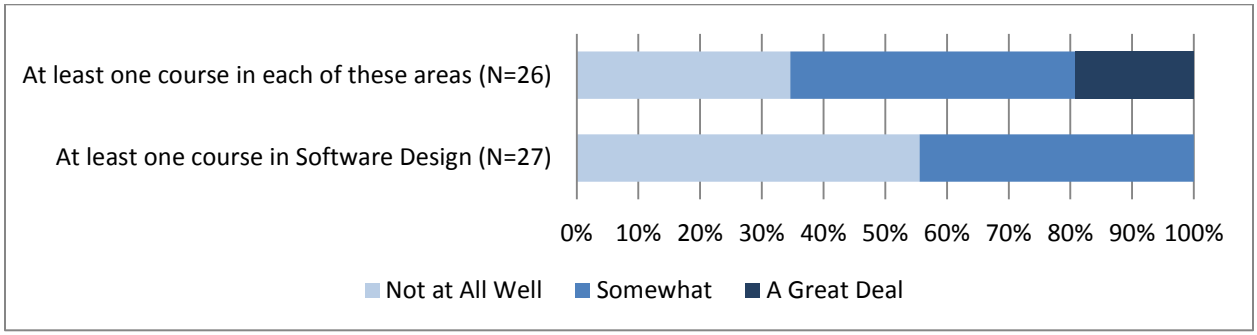



***Figure 9*** Degree to which coursework prepared you for "Testing Practices".

### *4.4.2 Instructional Design and Education Related Courses.*

One interview participant who had a degree in Instructional Design indicated that "The craft and science of Instructional Design was not something I was going to just pick up on my own" (I9). Others expressed varying levels of satisfaction with the degree to which their coursework prepared them for working on educational software. Useful aspects of particular programs included critical analysis of educational software, and valid and reliable assessment techniques.

Interviewees who had taken education courses (outside of the area of instructional design) indicated that these courses gave them a good foundation in learning theories (which

helps to create active and engaging resources for students). Software products developed by these participants grew out of knowing what they wanted for their own students. One participant indicated that his knowledge transferred to his current position in other ways: "Everything I learned in education applies to software design. There is a focus on the learner, a focus on the outcome, a focus on the process, a focus on individuality and customization – all important to both designers and educators." (I8)

Survey participants were asked how important their theoretical and practical coursework had been to their professional careers. As you can see from Figure 10, participants generally valued the education they received. However, Figure 11 shows that there are still some areas that participants felt they could have been better prepared for, especially developing and testing instructional software.

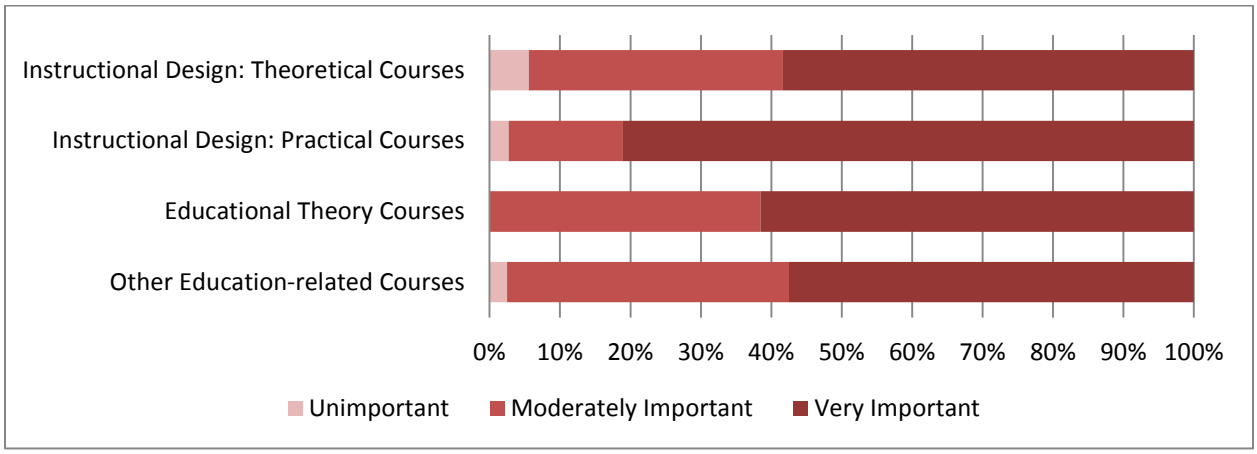



***Figure 10*** Importance of Instructional Design and Education Related courses on the job (N=42 who responded to this question set)

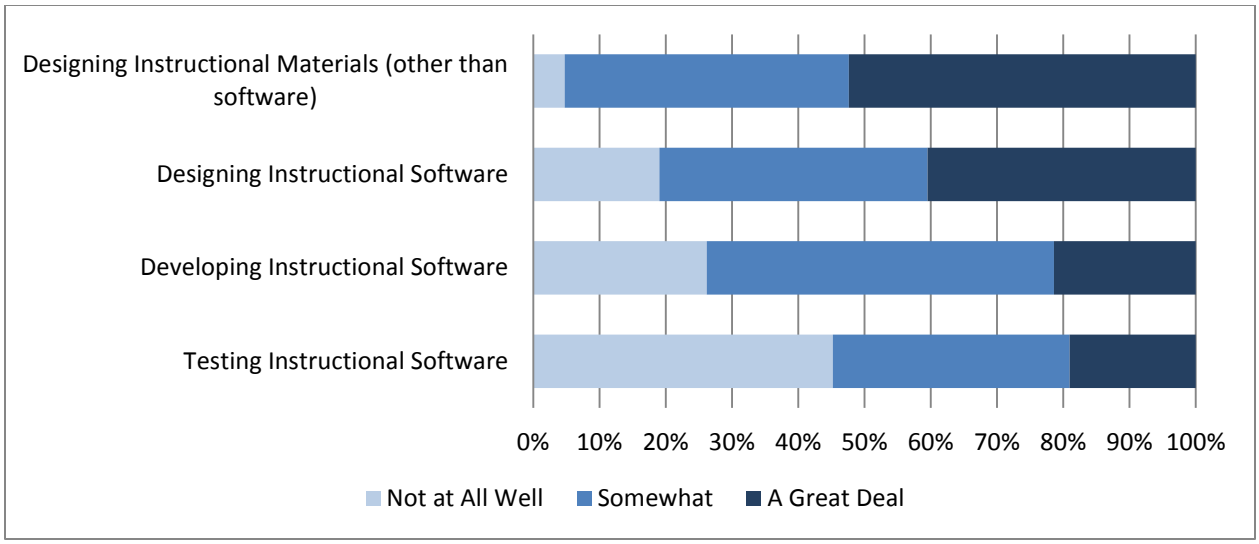



***Figure 11*** Degree to which topics were covered in Instructional Design related courses (N=42 who responded to this question set)

There was no statistically significant difference in responses to most of these items between those with some additional formal educational backgrounds in computing and those who did not have any formal background in computing. However, an Independent Sample Mann-Whitney U test showed that those with a computing background did feel more prepared to design ($U=296.0$, $p<0.05$) and develop ($U=293.0$, $p<0.05$) instructional software than those who had not taken any formal computing courses, likely because they had experience in both the software design and development aspects (from their computing coursework) and in the instructional and educational aspects (from their ID/education related coursework). See Figure 12 and Figure 13.

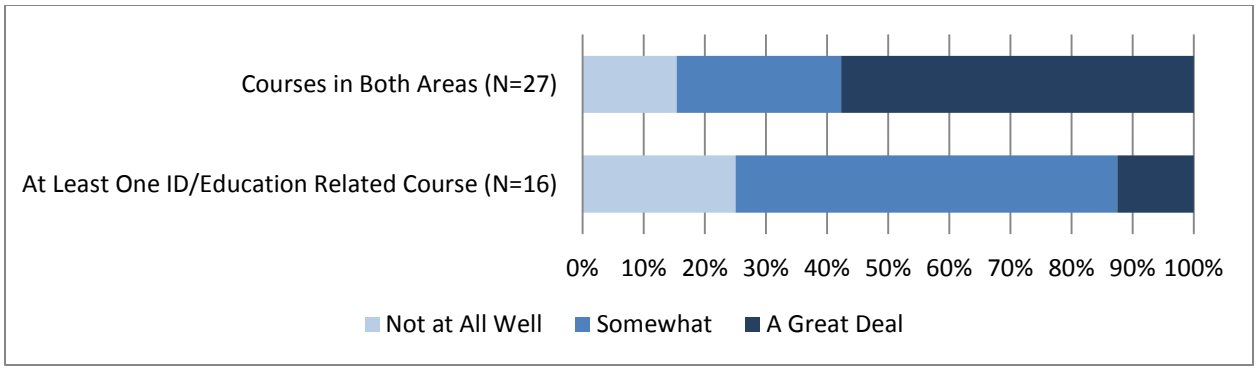



***Figure 12*** Degree to which formal education prepared you for "Designing Instructional Software".

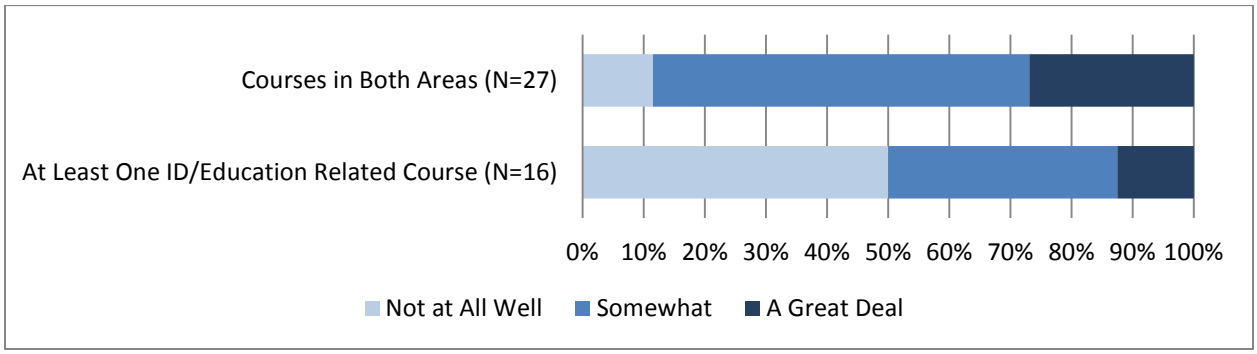



***Figure 13*** Degree to which formal education prepared you for "Developing Instructional Software".

Interestingly, participants who had had at least some coursework in both areas were less likely to indicate the importance of educational theory courses (U=130.0, $p<0.05$. NOTE: "N/A" responses were not included in the Mann-Whitney U test). This could be because some of the participants who indicated they had taken "at least one course" in both areas may not have as strong a background in educational theory as those who had taken ID/Educational but none in computing, and may have focused primarily on instructional design or related areas in their formal education, and who were therefore more likely to appreciate the benefit of an understanding of educational theory. Oddly, those with no background in software design were also more likely to indicate that "educational theory courses" were not applicable or not important in their current roles, while 100% of those with coursework in both areas indicated that this was at least "moderately important" (see Figure 10).

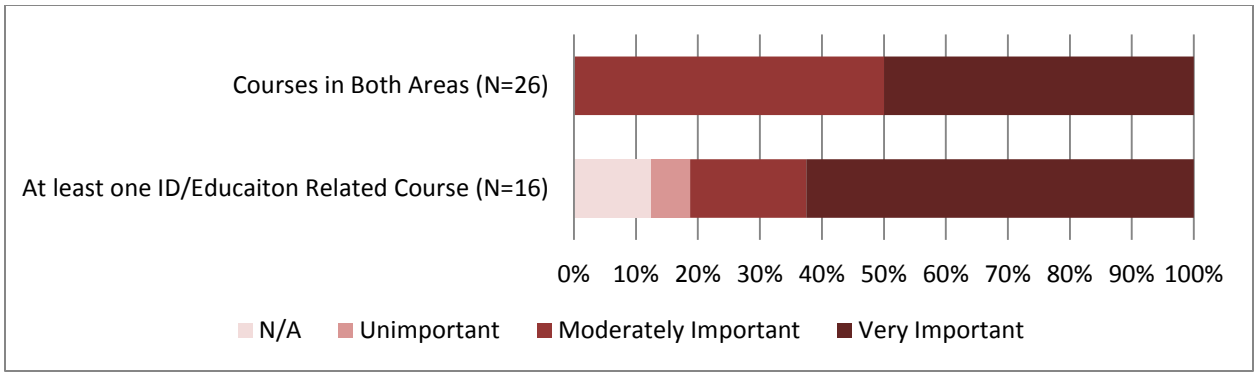



**Figure 14 Importance of "Educational Theory courses" across the span of your professional career.**

### *4.4.3 Preparation for the Job: "Unrelated" Courses and Experiences*

As was mentioned earlier, interviewees indicated that their non-computing and non-education-related coursework was very valuable to them. For example, one participant without a degree in either area explained the value of his English degree:

> I know it sounds cliché, but writing is at the center of almost every job…there are two things you learn in an English degree. One is that you hone your writing skills. The second is that you learn analysis. And that ability to do a thorough and clean and defensible analysis, and then to communicate it clearly, has benefitted me in every single aspect of my work. I don't regret one minute I spent on my English degree, because those skills have helped me to communicate with my supervisors, with my peers, with my subordinates, with customers…I could not have taken, in my opinion, a more valuable skill into the workplace. (I9)

Another participant similarly explained the benefits he gained from a formal grounding in the area of Psychology:

> The main [aspect of my Psychology degree] that was useful is the research methodologies [sic], going through the very deliberative process of coming up with a hypothesis and testing it, and…. Part of that is understanding and

> appreciating research methods, but also understanding and appreciating a methodological approach to a given problem. (I2)

Figure 15 shows survey participants' responses to being asked how important various types of "unrelated" courses and topics are to them on the job. The degree of variation may have to do with the variety of roles played and types of organizations participants are employed by. Unsurprisingly based on interview findings and the literature, "Communication" is the most highly rated item in this set.

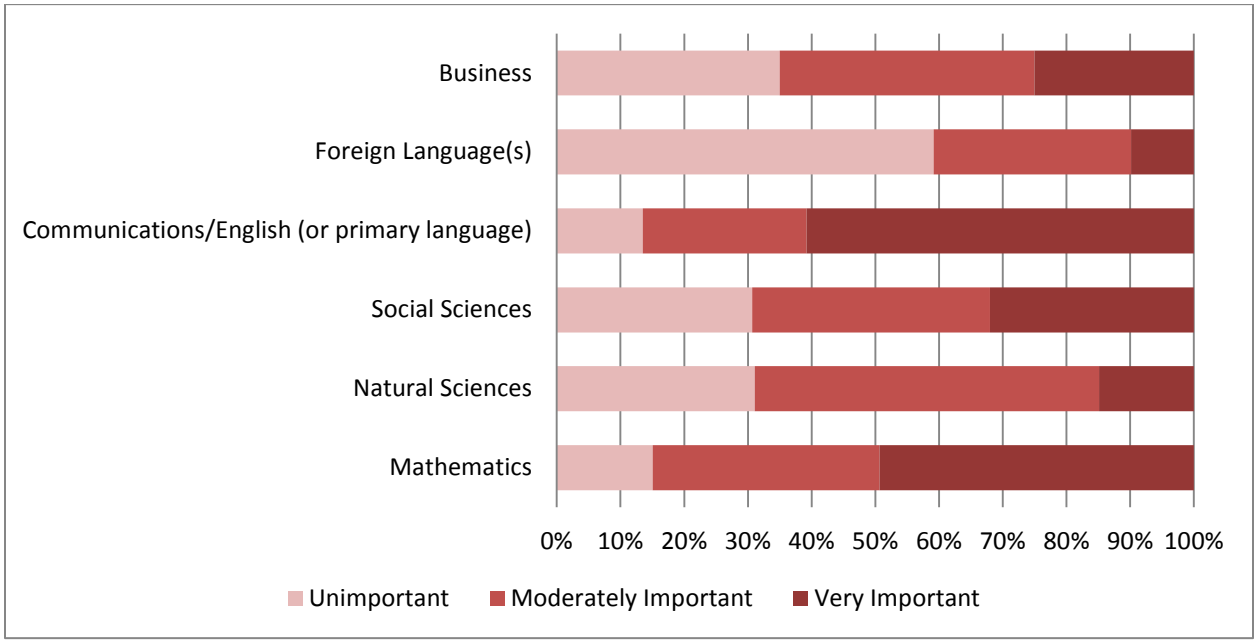



***Figure 15*** Importance of "unrelated" coursework on the job (N=78 who responded to this question set)

Figure 16 shows discrepancies between areas identified by interviewees and the literature as being important on the job, and the degree to which they were covered in formal educational programs.

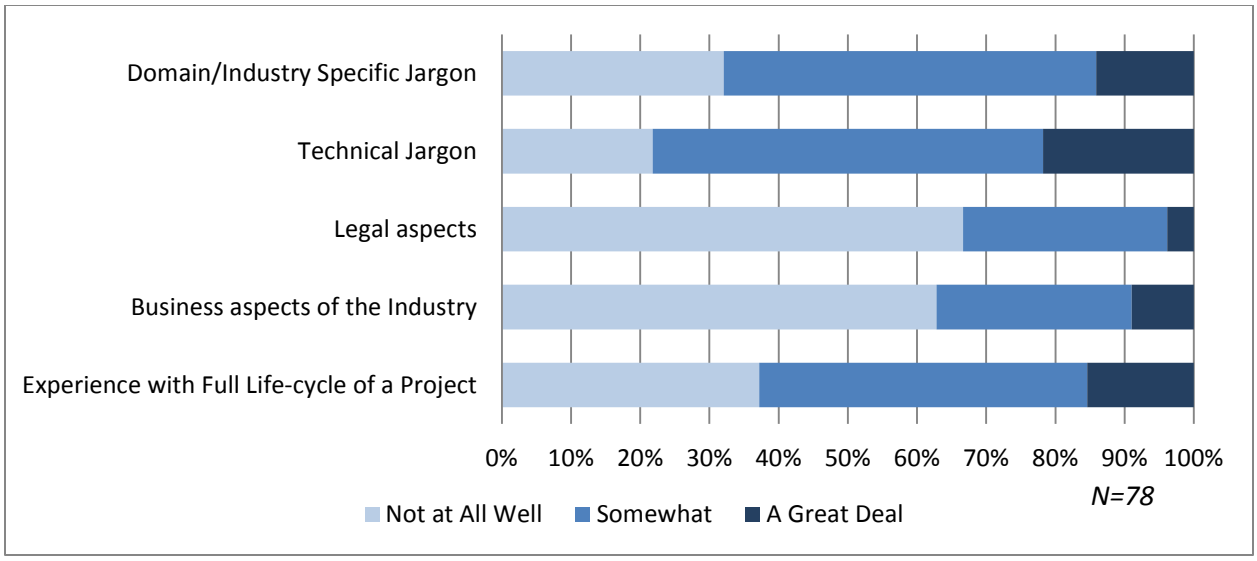



**Figure 16 Degree to which non-theoretical topics were covered in formal degree programs**

A Kruskal-Wallis test was conducted to evaluate differences between the four formal education groups. The test was significant for the degree to which participants were prepared for "Business aspects of the industry I work in" ($H(3) = 9.172$, $p < 0.05$). Follow-up pairwise comparison tests indicated that those with no background were less likely than other sources to feel prepared for the business aspects of the industry. Although the test did not show a statistical difference between those with a background in Computing only and those with a background in both areas, a quick look at Figure 17 shows clear pattern. Similarly, there were significant differences between the groups in the degree of preparedness for "legal aspects of the industry I work in" ($H(3) = 12.947$, $p< 0.01$). Follow-up pairwise comparison tests indicated that those with no background in either area were less likely to feel prepared for the business aspects of the industry, while those with at least some coursework in ID/education were most likely to feel prepared for the legal aspects. This is unsurprising, as those with a general Computing degree might not have had any background specific to the educational software industry. Unfortunately, participants were not asked about the degree of ***importance*** of either of these items to their professional careers, although an earlier item did indicate that "Business" was found to be "very

important" or "somewhat important" to about 2/3rds of participants, regardless of their formal educational background.

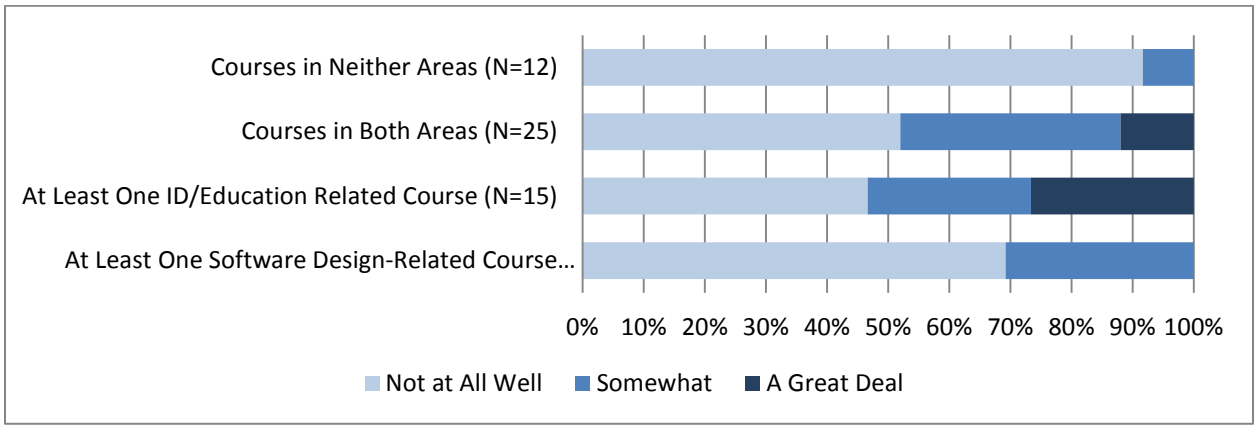



**Figure 17 Degree "Business Aspects of the Industry I Work in" was stressed as part of formal educational experiences.**

The Kruskal-Wallis test also showed significance for the degree to which participants were prepared by formal education for "Technical jargon used on the job" ($H(3) = 12.940$, $p < 0.01$) and "Domain/industry specific jargon used on the job" ($H(3) = 14.907$, $p < 0.01$). Pairwise comparisons indicated that those with a background in both areas were more likely to feel prepared for domain/industry specific jargon than those with a background in neither area or those with only a background in software design, while having a degree in both areas was preferential to having a degree in neither in understanding technical jargon (there were no statistically significant findings in either item to indicate whether ID/Education or Computing courses might be more helpful in learning either of these types of jargon).

### *4.4.4 Gaps between Formal Education and Needs on the Job*

#### *4.4.4.1 Gaps identified by interviewees*

Interviewees identified a number of gaps between skills and knowledge acquired as part of their formal education, and what they needed on the job.

Practical skills were high on the list of areas that interviewees felt unprepared for. As one explained, "more practically oriented courses might have… expedited my own development, as a designer. Otherwise, it seems I did have to learn…just about all of the practical stuff on my own, or by the seat of my pants, as we used to say." (I5) Another indicated that these skills were touched on at only a very high level (emphasis added):

> They talked at a very high level about … software development methodologies. They didn't go into any real detail… That is actually a huge part of the job, and I mean in a real, practical, low-level manner. **They talk about it in general terms, at a high level, but then, you can't apply that to everyday life. It's not useful**. (I4)

Participants also felt unprepared by their degrees for the "people skills" they needed on the job.

> The working with people part [is] critical. And complex…[my] formal education, for example, never taught me how to work… with a production team in an effective way, or to work with a staff in an effective way, or kind of nitty-gritty aspects [sic]...(I6)

Business and financial skills were not generally covered by participants' degree programs. As one participant who had taken courses and degrees in a number of fields recounted (emphasis added):

> It never happened, it didn't happen in my English degree, didn't happen in my journalism minor, didn't happen in my masters , it didn't happen in my PhD [in Instructional Psychology & Technology]. **And I can see why, from the point of view of the academy, that they don't consider it a core competency, but I also**

> **feel like any program that plans to create a leader should prove them with a sound understanding of business finance and accounting.** Because, whether their role is 'chief learning officer' or something else, they are going to be encountering that every day – or maybe not every day, but at least regularly. (I9)

#### *4.4.4.2 Gaps identified through analysis of survey data*

The next series of charts shows a comparison between the importance participants give to various topics, and the degree to which they believe they were prepared by their formal educational experiences in each of these areas. These items were based largely on findings from the interview, along with findings from an earlier related study on the non-formal learning experiences of software designers. In each case, a 2-sided Related-Samples Wilcoxon Signed Ranks test was performed to determine whether participants' feelings about the importance of an item and the degree of coverage in degree programs attended were aligned. Statistically significant differences are noted with asterisks.

Participants placed high value on the ability to work well in teams, perform multiple roles, and communicate with specialists in other areas. More than 40% also indicated that strong skills in a particular role or specialty area were "very important", with nearly all remaining participants indicating this was at least "moderately important". Yet, only a minority of participants indicated that these areas were well-covered during their formal education. Other than the ability to work well in teams, there was no significant difference in the degree to which participants felt prepared based on the type of formal education they had received.

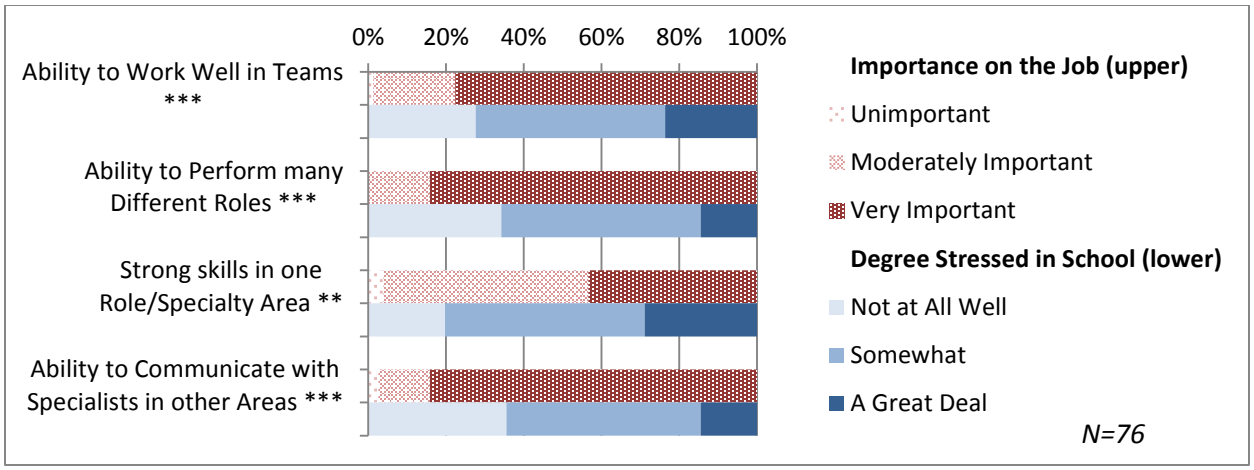



**Figure 18 Survey participants' responses to roles and communication**

*** statistically significant, p<0.05, ** statistically significant, p<0.01, *** statistically significant, p<0.001**

A Kruskal-Wallis test was conducted to evaluate differences between the four formal education groups. The test was significant for the degree to which they were prepared by formal ***education*** to work well in teams. Follow-up pair-wise comparisons indicated that those with a background in either ID/Education or Computing were more likely to feel prepared than those who did not have a background in either ID/Education or Computing[4], which you can also see in Figure 19. This is not surprising, as team-work is a typical component of coursework in both of these fields.

[4] When correcting for Type I error wtih Holm's sequential Bonferroni approach, only the pairwise comparison of "Neither" to "Computing courses only" and "Neither" to "Both" show statistically significant results. However, without this correction, a comparison of "Neither" to "ID/Education related courses only" also indicates a statistically significant difference. Figure 19 shows that the results for those with either of these degree types are similar, with those who had neither degree indicating a much lower level of preparedness, supporting the non-adjusted findings.

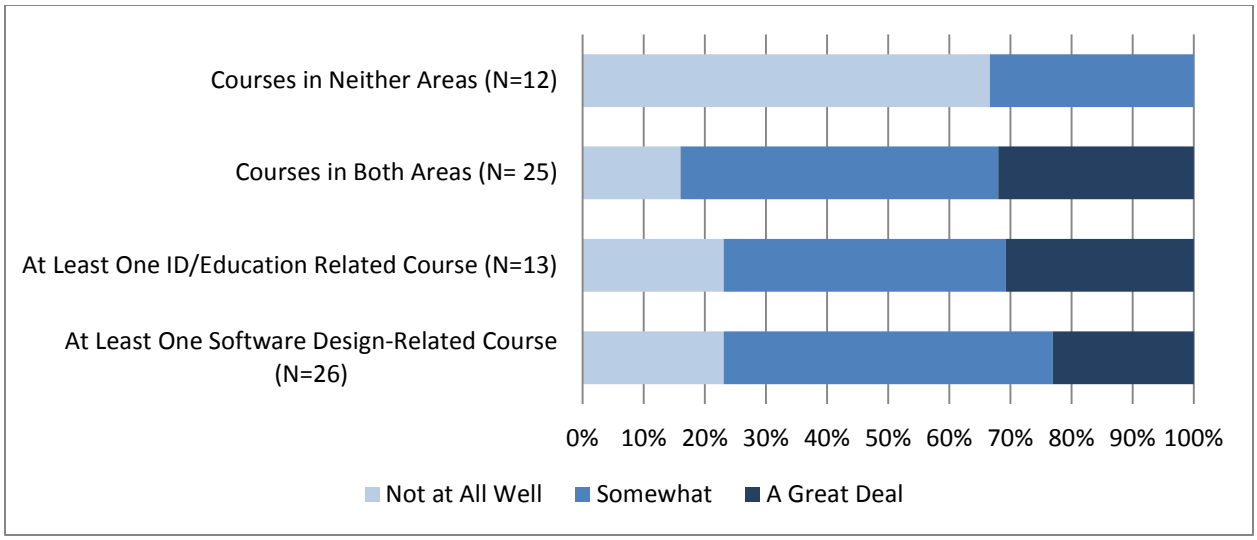



**Figure 19 Degree to which "Ability to work in teams" was stressed as part of your formal education.**

Interview participants indicated that critical thinking and the ability to solve problems creatively were the most important aspects of their job. Regardless of what degree path they followed, courses that stressed critical thinking were seen as the most valuable. Teaching one's self was a natural part of the job. Therefore, it is not surprising that each of these areas was considered "very important" by survey participants. However, the degree to which these important skills were stressed in degree programs varied quite a bit across all participants as a whole, as can be seen in Figure 20. There was no statistically significant difference in participants across *types* of courses taken based on a Kruskal-Wallis test comparing the four groups (Computing courses only, ID/Education courses only, both types of courses taken, or neither of these types of courses taken).

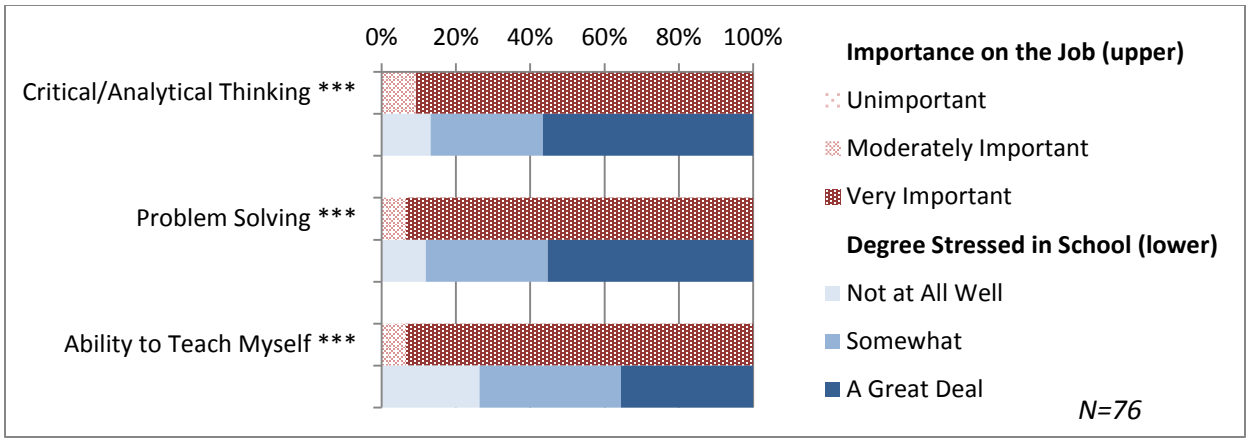



**Figure 20 Survey participants' responses to important types of thinking and attitudes about learning**

*** statistically significant, p<0.05, ** statistically significant, p<0.01, *** statistically significant, p<0.001**

As shown in Figure 21, knowledge of specific programing languages and technologies was seen as less important than other aspects of their education by survey participants. Responses to later questions indicated that this is because participants felt these areas could be learned on one's own. Interface design and user experience design principles were rated as more important overall, with few participants indicating they were "unimportant". A Kruskal-Wallis test revealed that those who had taken Computing courses were significantly more likely to feel that specific programming languages ($H(3) = 22.665$, $p<0.001$) were well stressed in school[5]. Those who had taken some courses in both areas were most likely to indicate that web languages and technologies ($H(3) = 15.108$, $p< 0.01$) and technologies were well stressed in school.

Knowledge of Interface Design principles and User Experience Design principles were found to be highly important by survey participants, as can be seen in Figure 21. A Kruskal-Wallis test comparing participants with different formal educational backgrounds indicated that there were significant differences in the degree to which participants felt prepared by their formal

[5] When correcting for Type I error wtih Holm's sequential Bonferroni approach, only the pairwise comparison of "Neither" to "Computing courses only" and "ID/Education courses only" to "Computing courses only" show statistically significant results. However, without this correction, a comparison of "ID/Education related courses only" to "Both" also indicates a statistically significant difference.

education for each of these items (H(3) = 16.232, $p$=0.001 for "Interface Design principles" and H(3) = 14.507, $p$<0.01 for "User Experience Design principles"). Pairwise comparisons between groups found that those who had taken at least some courses in both Computing and ID/Education were significantly more likely to feel that these topics had been stressed than other groups[6]. Results for these two questions were nearly identical, leading the author to question whether participants believe there is a difference between "Interface Design principles" and "User Experience Design principles".

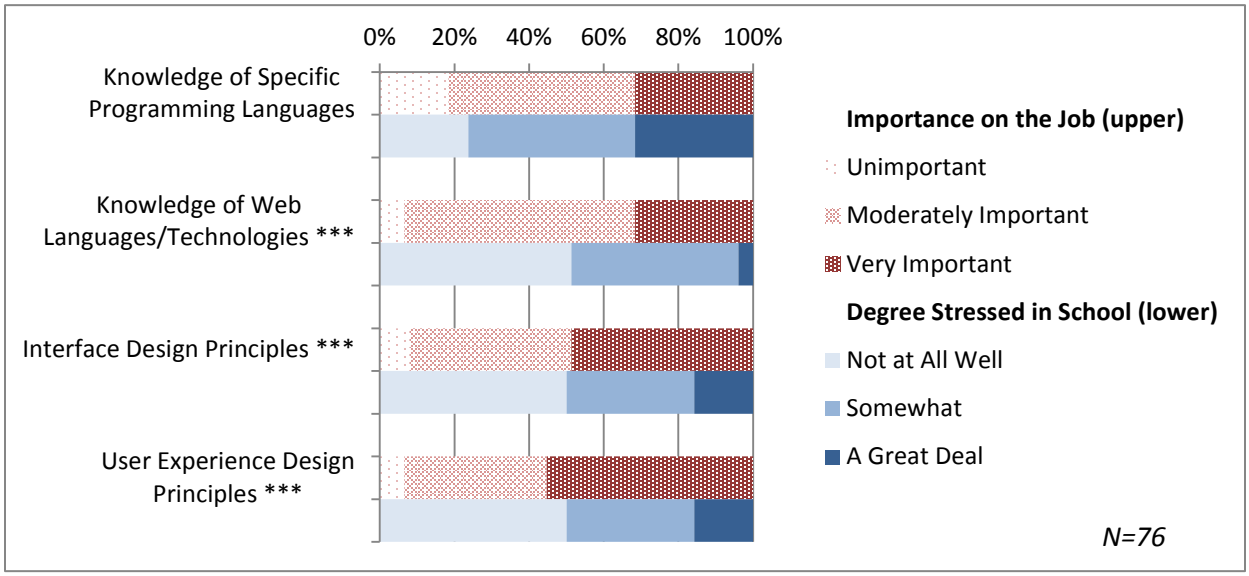



**Figure 21 Survey participants' responses to specific technical skills and knowledge**

*** statistically significant, p<0.05, ** statistically significant, p<0.01, *** statistically significant, p<0.001**

Figure 22, Figure 23, and Figure 24 show various types of resources and strategies used to learn on the job. As was discussed in the previous section on interviewee's self-learning strategies, each of these areas is seen as an important component of a software design

[6] When correcting for Type I error wtih Holm's sequential Bonferroni approach, only the pairwise comparison of "Neither" to "Computing courses only" and "Neither" to "Both" show statistically significant results. However, without this correction, a comparison of "ID/Education related courses only" to "Both" also indicates a statistically significant difference for "Interface Design principles" ($p$=0.042). As there is no statistically significant difference between "Computing courses only" and "ID/Education courses only", one cannot conclude that either of these types of courses is better in preparing students for this area.

professionals' work life. Therefore, it is not surprising that very few participants found these areas "unimportant". However, based on participants' recollections, these areas do not appear to be stressed by many degree programs.

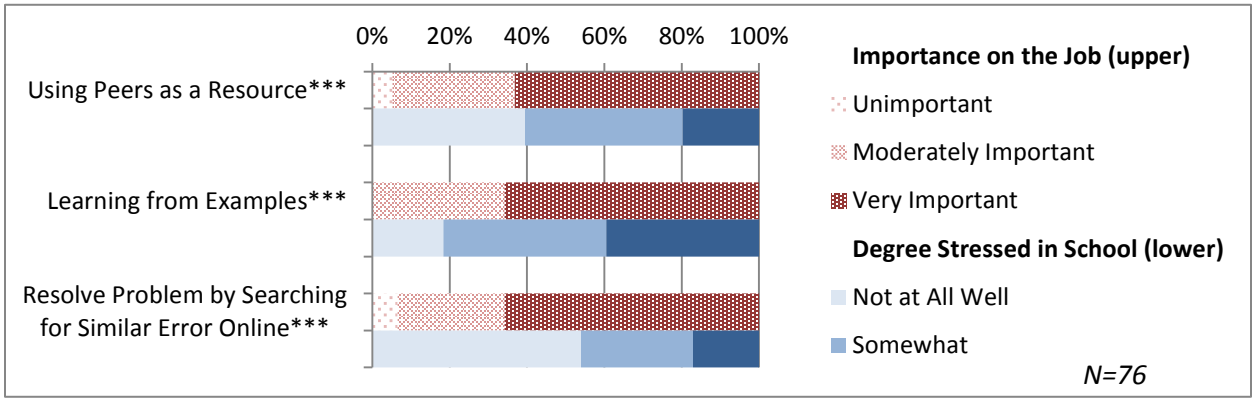



**Figure 22 Survey participants' responses to using resources to learn on the job**

*** statistically significant, p<0.05, ** statistically significant, p<0.01, *** statistically significant, p<0.001**

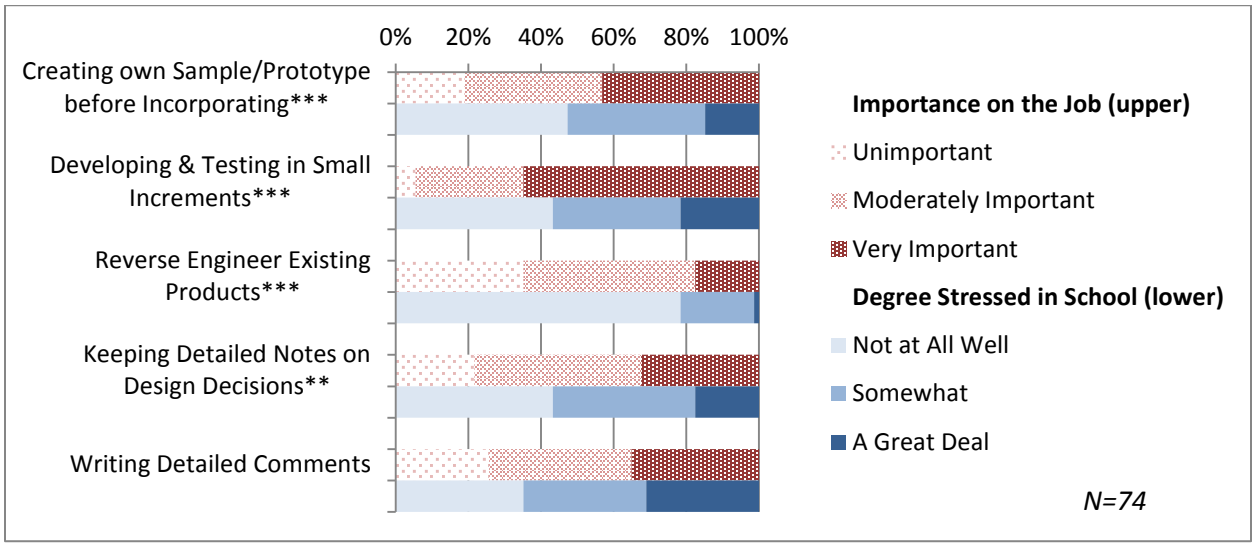



**Figure 23 Survey participants' responses to learning from experimenting and good coding practices**

*** statistically significant, p<0.05, ** statistically significant, p<0.01, *** statistically significant, p<0.001**

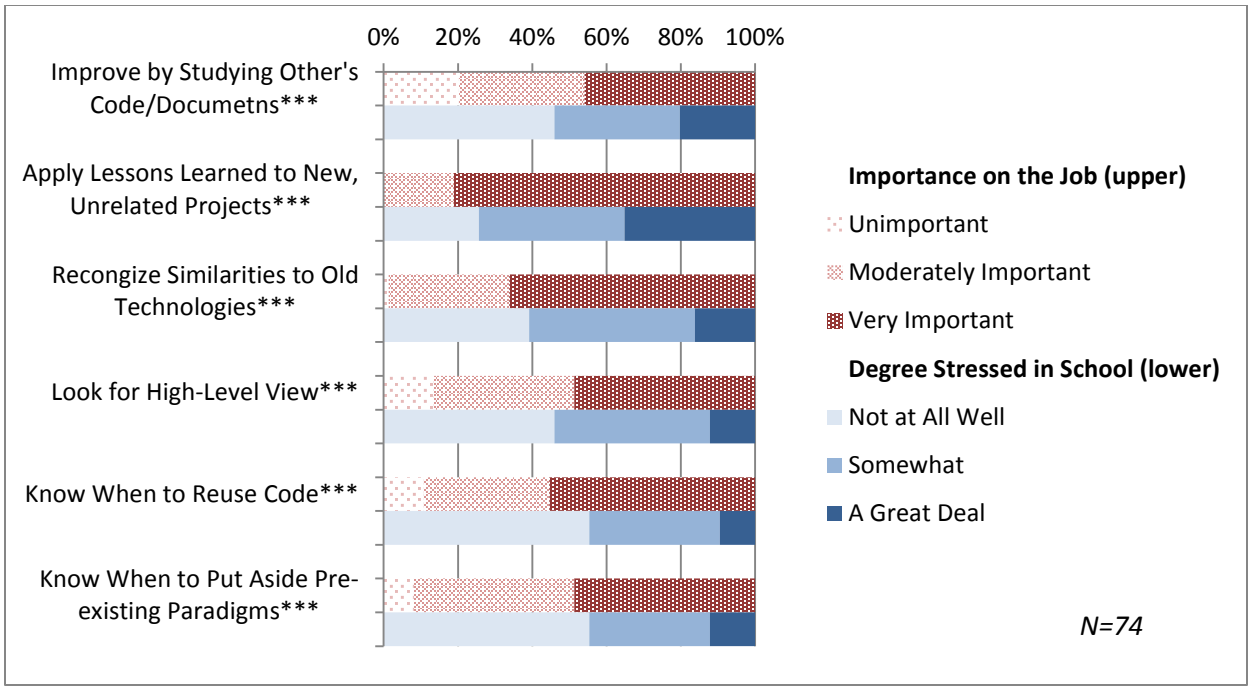



**Figure 24 Survey participants' responses to precedent use at the design and programming-level**

*** statistically significant, p<0.05, ** statistically significant, p<0.01, *** statistically significant, p<0.001**

## 4.5 Types of Non-formal Educational Experiences

As was discussed in earlier sections, self-learning is a crucial skill in this field, and one that participants often enjoy. This section discusses the types of non-formal sources available and how they are used. Data presented in this section is based on interviews.

### *4.5.1 Sources and Materials Used*

#### *4.5.1.1 Traditional print materials*

Because of the preference for learning on one's own time, published materials are an especially useful source. Traditional print materials, including books, "trade books", and journal articles are not used as frequently as they were prior to the ubiquity of materials available on the internet. However, books may be used when a depth is required. The internet may be used when looking for print materials, as it can be helpful that some online sources are cross-referenced with library search catalogs. Unfortunately, books themselves are not as easily searchable as

online materials and technical books rapidly become out-dated. Journal subscriptions are often used to keep an eye on new trends, but participants indicated that their number of journal subscriptions has also gone down as there are useful sources available for free online.

#### *4.5.1.2 Online sources*

Online sources are not only free and readily available, but also easily searched for just-in-time information or help. These resources include web-sites, discussion forums or online discussion community groups, RSS feeds, and real-time video tutorials. These sources have a number of advantages, including the ability to find multiple approaches to the same problem, and the availability of other people's responses to what you read. This is helpful, since one of the complaints about internet sources is that not all contain quality information. "I love sites that have comments, because I find that comments often give you the best feedback on the quality of something" (I3). However, Internet searches tend to be less successful when tackling an ill-formed question; "if you don't know how to frame the question, it's very hard to go to the internet" (I 3).

#### *4.5.1.3 Other People*

There are many situations in which the best way to learn is to get access to other people. They are especially helpful when an issue is complex, or when an answer is needed in a hurry. They can help you to find good sources of information, or identify poor sources. They may also help you to identify an alternative solution to an issue you are tackling. In addition to colleagues and end-users, becoming a member of a professional organization or an informal community (such as an online forum) is a good way to connect with others.

Subject-matter-experts are good sources of information in their own particular area of expertise. These may include co-workers or personal contacts, a client, or even an expert

consultant hired for the purpose of helping or training the team on specific topics. A team may also purposely develop their own content experts. As one interviewee explained, his team's goal is "always to try to build internal capacity" by developing expertise in certain areas (I6). Subject-matter-experts are approached for a number of reasons. They may have expertise in a particular role or field, for example, one interviewee with a technical background explained that he would approach the instructional designers on his team in to get a better understanding of how to present related topics in a way that "flows" well for learners. "I'm not an expert of that particular area…it's going beyond strictly usability. Because we are in a specific field, so then I would seek out the expertise of someone else." (I4) A colleague may also be approached on non-technical matters.

> If it's truly non-technical knowledge, mostly I network with people that I've worked with before. Because usually what I try to figure out is, you know, how to navigate the corporate infrastructure. And so I'll go to people that I've worked with in the past who I know have been in similar situations and I'll say, "hey I ran up against this situation, what suggestions do you have?" (I 9)

When designing instructional software, designers often need to find subject matter experts on a particular topic area, for the purpose of "knowledge transfer" of jargon and related concepts. These may be content area specialists, or may be users. One participant discussed working on a training system for a special mail-sorting device.

> I was capturing their knowledge, and encoding it in such a way that it would facilitate other people who had less skill and less knowledge to finding the same sorts of problems. So, you know, it would start up with questions like, "How is it failing, is the mail getting jammed up? If it's getting jammed up here, are the

> letters being crumpled, or are they just stacking up on one another? Are they flying all over the floor? Is ink smearing? Is it failing to read the addresses?" So it ask a series of questions, and at the terminal node of the system it would start to make recommendations, do this kind of test, replace this module, clean this part of the machine, try it again, those sorts of things. Because the people who repair these machines – some of had 10, 20 years of experience with the machines, but the younger people did not have their level of skill and knowledge, and yet the machines are still in use…that's why. So, I was capturing expertise from an older person." (I 7)

Working with experienced colleagues and other contacts has other types of advantages. People in different roles learn from one another over time. "I think that as we work together…there's something that happens as you work with a programmer and the designer, and, you know… all of you start to learn from each-other." (I2) Learning can occur simply by observing and interacting with others.

> I spend a lot of time actually listening to them and learning from [programmers] and understanding what they do… earlier in my career I really would sit down with them and have them explain to me what they do and why they do it and, you know whenever something would go majorly wrong because of the code I'd have them walk me through what went wrong, why it went wrong and what should have been done instead, and as a result I've learned a lot about how to write code well. Even though I cannot do it myself, I can often tell by the way a developer is talking about what they are doing, whether they're doing it right or not. Not the

> specific thing that they're writing, but whether they're structuring and approaching the problem in a good way. (I 9)

In one case, a consultant with expertise in a specific area was brought in to help on a project and train the rest of the team to use a new technology.

> We had a consultant come in for a year and we started from defining the requirements from the very beginning of the project to the end. And that seemed to work for us, for the team. Because you learn how to think, and you learn how to build, and you get to see what the RW problems are. (I 1)

Co-worker's critique can help software designers to improve. This may include formal code-reviews, or may be more informal. As one explained, he appreciated working in a larger organization earlier in his career, as there were "lots of smart people that come and kind of beat me about the head and shoulders when I wrote code and say, "oh, you know, there's really a better way to do this"(I 3). Content experts can be called in to review work to ensure that it meets standards, and colleagues from another institution with a similar mission can review the work to determine whether it meets learner's needs (at least in the case of government or non-profit work). Users themselves can be a good source of feedback. "We would download our product and get…feedback from the people in the field and users. For example we developed an English-only project, got feedback from people that that was great, but can we do it in Spanish." (I 6).

Human beings are especially helpful in getting past areas where you have an incomplete understanding.

> And a lot of times I'll [look things up on the internet] first, and become dangerous with that knowledge, and then go to one of my peers or subordinates and sit down

> with them and say, "now it seems to be this way, that's what I understand, can you fill in the holes?" (I 9)

Working together on a problem can be especially helpful.

> Sometimes not knowing how to frame the question, those are the exact problems that working with another person is perfect for, because together you can muddle through it, together you can find patterns that you hadn't seen before. [sic] (I3)

#### *4.5.1.4 Training courses*

Non-university training courses are another way to pick up new skills, or at least to help you to get to a certain point which you can build upon on your own. A good trainer can help provide a solid foundation for further learning. One interviewee who prefers short, intensive learning experiences takes week-long courses on programming and web-design at the "Big Nerd Ranch" (I2). Others indicated that he preferred to take courses for areas outside of their own expertise, such as "color theory and type" (I3) or usability testing (I9). However, other participants were not likely to take training courses, and some actively avoided them. Courses may not be appropriate for solving specific problems at hand, and may not teach what one really wants to know. Useful courses may not be available locally.

#### *4.5.1.5 Conferences*

Conferences can be "inspirational" and give professionals a chance to meet and talk with others who have similar issues and challenges. However, they may be less useful as a learning tool. As one interviewee explained,

> I find that the signal to noise ratio at a conference for me hasn't advanced my career has gone down a lot because most of what people are saying I either know, or I know that I don't need to know. It's very hard to find stuff that I don't know that I need to know with respect to the conference." (I9).

However, he felt that attending conferences outside of his own area of expertise was especially helpful.

> An exception would be when I actually step completely outside of the field, so I went to a conference … about children's media, because we were doing some pre-k stuff for the first time… and I felt every minute I was there I was learning. That was because I was outside of my field, not inside of it. (I9)

Expense is another concern in attending conferences.

#### *4.5.1.6 Examples of other's work*

As will be explained in section 4.5.2, examples of other's work can be particularly important resources in self-learning. These can include entire designs, or can be at the code level. A software designer may spend time generally looking for good examples, or a sample could be sought out when trying to understand how to solve a particular type of problem or learn how to use a particular type of technology.

One participant mentioned that when he began working in this field in the 1980s, there was not much support available in his formal educational program for the development of the type of software he was working on. Nor were there a lot of good models of educational software.

> So that was kind of up to us, and it was up to me as I was doing my own software, and based on the models that was out there so far… there was one big set of programs in our field that had been developed at the University of Deloware and actually another set from the University of Illinois at Urbana Champagne, that both had relied on previous work with the PLATO system, which was a very big mainframe-based system from the 70s, that was specifically oriented towards the

> educational area. The PLATO was an acronym for something like 'personalized learning and training operation'. So they, I presume they had done a lot more…uh… real… critical design work in those programs, they were done with large grants and things, federal and university grants, and …they had lots of development time involved, and so they, I guess, sort of served as the models in our field. (I5)

At a lower level, software designers learn by studying. "[I] usually start by looking at programs that do a good job, and start deconstructing them…so, getting under the hood." (I3) Developers generally do not simply look at code samples; they learn by installing, running, and modifying them.

> So, I usually start with something that is doing kind of what I want it to do, but not quite, and then I start to modify it, and tweak it, and through that I have to figure out how the code works and how it's constructed. (I 3)

In addition to learning new skills, samples may be helpful to "challenge yourself by seeing how other organizations do the same thing" (I 1). Some participants regularly spend time "looking around" to see what others have done. One interviewee mentioned that web sites from professionals in other design fields are particularly good sources of inspiration.

> I look at design sites, visual design, information design, user experience design. I try to cast my net fairly wide, and look for inspiration. I especially like looking at design sites, visual design sites even though I don't do a lot of visual design….(I3)

He explained that he does not directly incorporate what he sees into his own designs.

> I think it's more that there's stuff that gets wedged back in some portion of my brain that then I use creatively and gets expressed in the work that I do… it does occasionally express itself in my work, when I'm having to do User Experience design or I'm working with user experience designers, because I've been looking around a lot, I can bring new ideas to the table, so I can say "ah, I saw a widget that did this and it worked in this way, and it was really cool, and we might try that here"… but [generally] I just feel that it kind of like it kind of is the fuel that I use in my own creative work. It's kind of like the raw material… I don't always feel like it expresses itself directly, but I always feel like I'm better able to do my work when I'm well-fed so to speak. I know that sounds all vague and a little touchy feely, but that's definitely how I feel about it. (I 3)

#### *4.5.1.7 Other sources*

Other sources mentioned by individuals included learning from teaching, searchable publications, and going to museums for inspiration. A few participants mentioned the need for some way to help (others on the team) to learn "soft skills", but they were not sure what mechanism to recommend.

#### *4.5.1.8 Choosing and mixing sources*

When choosing a source, participants often go to the easiest or most convenient source first. Some sources are particularly obvious for a given topic.

> If I know that there's somebody around know knows something really well, and that's the first thing that thing that comes into my head, that's probably where I'd start. On the other hand, the flip side, is that if I know where a resource is online, that is what I'll reach for. So, it's really more what pops into my head.(I3)

Participants frequently use a mixture of sources to learn a new skill or concept.

> Generally if…I actually kind of use an amalgam for most things. So I will start… for example, if I wanted to learn a programing language, most likely I will have been already reading articles about it, I would have done a couple of practice exercises that I found online, or whatever, and that would be kind of forming the foundation of forming my interest, and do I really want to do this. Then I would look for, hey, is there a week long course I could take to get form absolute beginner to moderately self-sufficient beginner. And then, it's getting to that plateau. Then you continue with getting different articles and trying different things to get what you actually accomplish some of it. (I 2)

### *4.5.2 Self-learning strategies*

Learning new skills, programming languages and approaches is a natural part of working on a software design project. This often means learning something new on a just-in-time basis to resolve an immediate need.

> I also find I learn the most when I'm programming, when I am working on my own problem, when I'm working on something that's right in front of me and very tangible, and understand the workings of the problem. I have… noticed that the further away a piece of code is from the thing that I am trying to do at the moment, the less I can learn from it. (I3)

As mentioned earlier, it is typical to learn from samples of other's work, often through a process of experimentation. This may involve moving between one or more examples and making small changes, then trying out the technique on one's own code.

> I also do a fair amount of experimental and exploratory work of my own. So, sometimes I'll be called upon to say, well we need to do, I don't know, something

> like security. So I will go out and do an internet search and turn up 2 or 3 different approaches to doing some kind of security thing. And then I'll... If the source is available to me, I will download it and will actually try writing some software purely on my own, just for the sake of understanding how that works. So it's an experimental approach to understanding technology. (I7)

This type of experimentation can involve a fair amount of trial and error.

> And then practice, I mean, especially with design… "practice" is a weird word for it though. It's more like trying and failing, many times. Um… it's not like you practice skating and you get better. The way I look at it anyway, is there are ten-thousand failures and you rule out all except for the one you want. (I2)

After explaining this process, one interviewee reflected:

> it's interesting, "self-taught" makes it… sounds like it is all comes out of my head, but it's more like I know that I am standing on the shoulders of giants and I know where to find the next giant. (I3)

One participant purposefully created notes on his own projects, problems, and solutions, which he could refer to in similar situations in the future.

> So, I keep day to day notes on exactly what I am doing with that software and the project, what succeeds, what fails, what is not working, what the data looks like, where it's broken, how it's broken, error messages, XML notations, just everything, everything, on a day to day basis. So that I can go back and any point and "how do I get from here to there". Also explains sort of blind alleys I went down and why I abandoned them, and it's all in there, so if I forget I can look it up. Not everybody does that. (I7)

In addition to updating technical knowledge and improving design skills, these types of self-learning approaches help software design professionals to gain experience in their specific domain over time.

## 4.6 Recommendations for an Ideal Undergraduate Program

Interviewees and survey participants were both asked to describe what they felt would be an "ideal bachelor's degree program to prepare someone for your current position." Survey participants responded to several closed-ended questions regarding the degree type, the importance of domain-specific content, and the importance of technical content. An open-ended question prompted them to explain what other traits such a degree program should have. Forty-four of the 74 participants who reached that point of the survey provided at least some response to this question, though some participants responded only briefly, while others provided fairly detailed responses.

Themes discussed in this section were derived from the survey questions as well as interview responses on related questions. Traits that interviewees indicated were particularly valuable in their own formal education were also included in the analysis.

### *4.6.1 Degree Type*

Interviewees and survey participants had a number of recommendations for an ideal undergraduate program. Recommendations varied depending on the participants' own educational background and experiences. For example, an interviewee with a background in Computer Science said:

> I think computer science degree is the most useful. A CS degree teaches the fundamentals of computers and software, and there are many basic concepts that

> you need to know to be a good programmer or a specialist in this field… no matter what you do. (I7)

In contrast, others indicated that ideally they would like to include some time in each field, but were concerned about how realistic that might be.

> I don't think there is another one single thing; you'd probably need a good mixture. You'd probably, you really do need an educational background, but then you need to complement that with technical skills. So I think, either one would work, if you had formal education in one, and informal on the other, that would probably work. If you had formal education in both, that would probably be best (I4)

Others found their own backgrounds to be most useful. "My masters HCI degree did a lot of it. I would probably retool it and add a year specifically for education software, I would add time in the ed. school." (I2) "A science degree… For debugging the software and maintaining a stable system, you need to be able to solve problems, using the scientific method." (I6). One pointed out that the type of degree should depend on the personality of the student.

> Someone like me, computer science was really a good choice. Worked out really well for me. But someone with a different personality or experiential bent, I wouldn't recommend it for everyone, because they are not technically minded like I am, and it is kind of a waste of time for you. (I8)

Others noted that any path could work, or came up with interesting alternative paths. "Whatever they are most passionate about, that is what they should do. I am a big believer in Joseph Campbell – follow whatever you are passionate about and makes you excited" (I6). "A mix of creativity with art and design combined with user experience design" (S54). "Either

instructional design or sainthood, with a maths minor" (S17, whose role includes development and instructional design, and who is also a K-12 teacher.)

Closed-ended survey responses followed a similar pattern. As can be seen in Table 20, those with a background in computing-related areas suggest similar degrees. Interestingly, those with a background in instructional design or education or a background in both areas are more likely to recommend a hybrid degree or double-major, which was also the most common response overall.

**Table 20** ***Suggested degree type for an ideal degree program.***

| | All | Computing only (N=25) | ID/Ed only (N=13) | Both (N=24) | Neither (N=12) |
|---|---|---|---|---|---|
| **Computer Science** | 12.2% | 20.0% | 0.0% | 8.3% | 16.7% |
| **Software Engineering** | 17.6% | 44.0% | 7.7% | 0.0% | 8.3% |
| **Information Systems** | 0% | 0.0% | 0.0% | 0.0% | 0.0% |
| **Human Computer Interaction Design** | 5.4% | 8.0% | 0.0% | 0.0% | 16.7% |
| **Instructional Systems Technology/Instructional Design** | 20.3% | 4.0% | 15.4% | 37.5% | 25.0% |
| **Other Education-related** | 0% | 0.0% | 0.0% | 0.0% | 0.0% |
| **Hybrid program or double major including CS (or similar) and Education (or similar)** | 25.7% | 8.0% | 46.2% | 41.7% | 8.3% |
| **Doesn't matter/any path would work** | 2.7% | 0.0% | 7.7% | 0.0% | 8.3% |
| **A degree has little value for working in this field.** | 0% | 0.0% | 0.0% | 0.0% | 0.0% |
| **Other (please specify)** | 16.2% | 16.0% | 23.1% | 12.5% | 16.7% |

*Note:* The most common responses for each group highlighted in light red.

When asked whether it is important for students to have a domain-specific background (e.g. a background in education when working on educational software) (see Table 21) or whether it is important to have technical skills (see Table 22), responses also varied depending on participants' own background, with those with a background in instructional design being much less likely than other groups to indicate that domain-specific knowledge is not important and those with a background in computing indicating that technical knowledge is necessary.

Some interviewees indicated that although domain-specific knowledge is useful, having experience in other domains could be useful as well.

> I find myself looking at each course as a spoke in a wheel where there is always crossover to other domains. For example, as I pursued my master's degree, I found that courses I was taking could be used outside of the internet security realm and applied to an educational realm in terms of student education and safety. I do look at educational software and gaming as an instructional medium that needs some new foci so that transference of knowledge is obtainable and not just the entertainment value. (S73 interview)

**Table 21** ***Importance of "domain-specific specialization or focus within this type of degree program (such as "education" or even a more specific area such as 'language education' or 'science education')" in an ideal degree program.***

| | All (N=74) | Computing only (N=25) | ID/Ed only (N=13) | Both (N=24) | Neither (N=12) |
|---|---|---|---|---|---|
| **Yes, it is important to focus on the domain one plans to work in** | 27.0% | 32.0% | 30.8% | 25.0% | 16.7% |
| **Yes, students need to work within one domain to get practice in a realistic project, but the specific domain is not important** | 29.7% | 16.0% | 38.5% | 37.5% | 33.3% |
| **No, not important** | 35.1% | 40.0% | 23.1% | 29.2% | 50.0% |

| | | | | | |
|---|---|---|---|---|---|
| **Other (please specify)** | 8.1% | 12.0% | 7.7% | 8.3% | 0.0% |

*Note:* Most common responses for each group are highlighted in light red. Closed-ended options were based on common interview responses.

**Table 22** ***Importance of "formal training in programming languages and other technical aspects of software design/development" in an ideal degree program.***

| | All (N=74) | Computing-only (N=25) | ID/Ed only (N=13) | Both (N=24) | Neither (N=12) |
|---|---|---|---|---|---|
| **Yes, it is important** | 44.6% | 68.0% | 23.1% | 45.8% | 16.7% |
| **Yes, important, but not as important as a solid background in education or instruction** | 12.2% | 4.0% | 15.4% | 25.0% | 0.0% |
| **Yes, it is important, but not as important as the ability to think critically** | 28.4% | 16.0% | 38.5% | 25.0% | 50.0% |
| **No, not important – people can learn that on their own** | 10.8% | 4.0% | 23.1% | 0.0% | 33.3% |
| **Other (please specify)** | 4.1% | 8.0% | 0.0% | 4.2% | 0.0% |

*Note:* Most common responses for each group are highlighted in light red. Closed-ended options were based on interview responses.

However, across all groups a fair number of participants indicated that it was not necessary to have domain-specific preparation, or that having domain-specific examples and projects were important, but only in order to provide a realistic context to learn in – the specific domain one "practiced" with (e.g. "education" for those with a computing background) was not important. For example, one interviewee said:

> I think it's useful to be exposed to the idea of immersing yourself into an industry or topic-specific area, as a way of understanding how to go about doing such a thing, not that what they learn specifically will be of value, but the amount of processing and the communications and learning skills that are developed in the process of trying out one or two of these areas is useful (I7).

From this point of view, the ability to develop requirements based on user (e.g. student) needs, work in a multi-disciplinary team, and think creatively is more important than knowledge in a specific content area. Participants also explained that if they needed additional expertise, they would contact a Subject Matter Expert, such as the team's Instructional Designer or another colleague or client with expertise in the relevant area. It is initially surprising that despite differences with other groups, over 50% of those with an instructional design background indicated that domain-specific knowledge is either unimportant or only important as an area for practice during a degree program. However, the question literally read "Is it important to have a domain-specific specialization or focus within this type of degree program (such as 'education' or even a more specific area such as 'language education' or 'science education')?" For those with an instructional design background, the "domain-specific knowledge" might have alluded to a more focused area such as "science education for 5th graders". For interview participants with a background in Computer Science or a related field, however, "education" or "instruction" was itself a specific domain which could be learned over time as one works with clients and end-users to create requirements for a project.

Similarly, a number of participants indicated that although a background in Computing is helpful, it is not as important as critical thinking or other traits such as "people skills" and flexibility. For example, one survey participant said:

> If someone studies computer science or software engineering or IT or education, those would all be very helpful tools to bring, but they are not necessary to being a good QA engineer. The ideal program would mix the practical experience of engineering/software/technology with people skills, puzzle-solving, teamwork, and flexibility. (S62)

However, as is evident in Table 22, the vast majority of participants (over 85%) indicated that having formal training in programming and other technical topics is important, though a fair proportion of these indicated that this was not as important as a program that fostered critical thinking (28.4%) or topics in education or instruction (12.2%). Interestingly, among those who had at least some formal education in both areas, the most common response was that a technical background was important, while the majority of the same group indicated that domain-specific knowledge was either unimportant, or important only as an area in which to practice.

### *4.6.2 Traits to foster in graduates*

Many participants focused on the types of graduates that should be produced by the degree program. Traits that a program should foster in graduates are discussed in the following sub-sections.

#### *4.6.2.1 Communication and Team skills*

The highest number of recommendations related to the development of communication and interpersonal skills. Communication skills included the ability to make a presentation, as well as general and technical writing. These skills play an important role in fostering collaboration and team-work, as well as the ability to communicate with those outside of the organization.

One interviewee reflected:

> I don't regret one minute I spent as an English degree student, because those skills have helped me to communicate with my supervisors, with my peers, with my subordinates, with customers…I mean, I could not have taken, in my opinion, a more valuable skill into the workplace.

> Because I'm an effective communicator, I feel like I've been given more opportunities to participate in strategic circles, because a lot of the people who strategize don't necessarily know how to communicate, and they sort of invite you in because they know that you'll do a good job of it and then suddenly you're sitting at the table. I may be overstating this a bit, but I really do feel strongly that it's made a big difference in my career. (I9)

Good communication skills are especially important when communicating a difficult message.

> They need excellent people skills in order to spend a career telling other people that their code has problems or why the product can't ship yet. (S62)

One survey response stressed the importance of listening skills: "Ability to communicate ideas and LISTEN to others" (S64). An interviewee also stressed that getting the information one needs is a skill that can and should be taught. "Learning to ask the right questions to the right individuals is something that does not necessarily come easy to me however there were courses that provided a framework of what kinds of questions to ask." (S73 interview)

#### *4.6.2.2 Design Judgment*

As discussed earlier, good design judgment is crucial in these types of design situations. A number of suggestions for an ideal program related to developing judgement skills. One interviewee specifically recommended "practical training that helps them to develop design judgment and how to make decisions when faced with different types of problem" (I 8). Others mentioned things students need to learn to be able to make good design judgments.

Some recommendations could be applied to any software design project. For example, a software designer should avoiding "designing yourself into a corner" (I2). It is important to

know when to say no to a strict adherence to theory when it is not appropriate, and know when it is appropriate to reuse code and when it is not (S56).

Other recommendations are more specific to the domain of educational software design. For example, it is important to know what educational applications are possible and what is suited to the type of technology being used (S74 interview) and to "analyz[e] critically the use of technology at all as an instructional tool, and what all the biases are that are involved in that" (I5). Content must always be considered along with other design aspects; "Content is king. Good content trumps flashy design." (I6).

The task of acquiring domain-specific knowledge is itself a unique skill which is developed over time. One interviewee called this learning "subject-matter patterns." She explained:

> I have found that all disciplines have critical structures or patterns to their knowledge. These structures give learners something with which to organize the knowledge to be assimilated during their course and help to give a common frame of reference to relate to practitioners in the discipline. An example of this might be for an apprentice mechanic to know the components of a car motor: what they are, what they do, and how they interact. In a medical realm, this would be anatomy, physiology and pathophysiology. Each of the major areas have sub-areas with their own patterns – sort of like an outline of written material, but often very specific to the discipline involved. (s74 interview)

#### *4.6.2.3 Creativity*

Fostering creativity was another commonly recurring theme. As one survey participant put it, "people just need to be creative and go from there. All else will follow" (S67). An interview participant explained:

> They also need the creativity side too…a lot of the people in my CS program were really good technically, but [weren't creative]. Really important – if you are not creative in development you lose a part of your competitive edge. When you thinking about it most people in CS programs can create a complex list with data-structures in memory, [but] so what – not everyone knows how to take those data structures and use them to transform the look and feel of a user application.

When asked whether this skill is specific to education or generally useful for software developers, he responded:

> I think creativity is a multi-disciplinary thing. There is no one discipline that can claim creativity. It crosses borders. (I8)

#### *4.6.2.4 Seeing other Perspectives*

The ability to see other perspectives and understand how others think is important in working with others on a team, as well as in designing something that works for users. Recommendations for learning social sciences such as anthropology and sociology (discussed in section 4.6.4.6) often relate to this need.

> If your education doesn't help you to really understand that not everyone learns the way that you do, then you have missed something critical. This is much more important to educational software than to general UI design. (S9)

#### *4.6.2.5 Critical Thinking*

The term "critical thinking" without further explanation was frequently included in short survey responses. This is unsurprising, considering that survey participants nearly universally indicated this is a highly important skill on the job. Recommendations for fostering critical thinking include labs and projects (including those in unrelated domains) and "student presentations and critique of existing software" (S74 interview).

#### *4.6.2.6 Strategic, methodical thought process for problem solving*

Another very frequent theme related to the types of thinking that go into effective problem solving. Terms used by various participants to describe this type of thinking included "systematic thinking", "strategic thinking", "scientific thinking", "logic thinking", and "structured thinking". One survey participant summarized it as the "potential to understand the problem, critically analyze it, and solve it" (S11). As discussed in section 4.6.2.5, critical thinking skills were often considered even more important than specific technical skills. "They need to come out of the program knowing the basic skills of the trade, but they need to be problem-solvers at heart" (s54).

Participants' personal experiences indicate that this type of thinking can be developed through courses in a variety of areas, including English, Psychology, and Physics, as well as in courses related directly to software design. "I think the most important thing is the way you think. [Computer engineering] courses show you to think in a certain structured way, to decompose problems into smaller bits." (I1) Once a problem is analyzed, it can be effectively communicated. One participant explained that his English degree was especially helpful in teaching him to do this type of analysis. "[the] ability to do a thorough and clean and defensible analysis, and then to communicate it clearly, has benefitted me in every single aspect of my

work" (I9). Another interviewee with a background in psychology explained how learning – and frequently using – the scientific method in school prepared him to use this type of thinking in the future.

> The main thing with the psychology degree that was useful is… the research methodologies, going through the very deliberative process of coming up with a hypothesis and testing it, and….you know. Part of that is understanding and appreciating research methods, but also understanding and appreciating just a methodological approach to a given problem.
>
> It's a discipline, it's like exercise. When you start exercising it is very hard and uncomfortable and your body rebels against you, but if you keep doing it, eventually it becomes natural, the body becomes accustomed to it, it is no longer rebelling as much…. It's going through the steps… And in my undergrad, we would go through those steps not just in the thesis but also in papers and different things, and if you go through those steps, you have kind of a routine. You know what I mean? (I2)

He explained how this is useful in software design:

> Designing software is a very broad thing. You know, I'm going to design an application…that means a hundred-thousand different things have to happen. And it's helpful to be able to break out what those things are, so that they can actually be accomplished. So, for example, again, defining what the software is supposed to do. Defining who the software is supposed to work for. These are individual categories of tasks that need to happen. (I2)

One interview participant reflected that his understanding of what he had learned evolved over time.

> If you define preparation as being about what you learn and the kind of structures for thinking about/addressing your problem that you develop along the way then yeah [his formal education] did [prepare him]. You learn these ways of thinking sometimes from disagreeing with the approaches that you were taught or that the professors used. You know, you learn from the hidden curriculum of the educational environment where you are challenge not just by your professor but by your peers and those trying to understand what you do and how you make a difference. (I 8)

In addition to analyzing a problem, strategic thinking helps professionals to come up with appropriate designs. "I look for people who are strategic thinkers...people who can see and visualize a future world where the solution is already in place then figure out how to create the solution." (S54). Students must also be prepared to use their problem-solving skills while developing and testing a design "the ability to tackle and solve puzzles (i.e. things that are going wrong with the software)" (s62).

#### *4.6.2.7 Technical Literacy for areas outside of one's own expertise*

Non-specialists in an area need to be sufficiently technically literate to be able to communicate with people in other roles and appreciate what they do, to know what is possible, and to inspire creativity in the overall design. For some roles, it is "Not necessary to know how to program, but need to "get" technology, understand the underlying principles behind what the programmers are doing" (s15). Technical literacy extends to areas beyond programing. "I do seek people who are technically literate in the latest art, illustration, design and animation

technology," explained one survey participant, although he went on to explain that it is even more important that potential employees can see the big picture and have a passion for bringing good solutions to life (s54).

Some amount of formal background in the type of medium one will work on is also valuable. For example, one survey participant who recommended an instructional design degree also recommended inclusion of 3D modeling courses. When asked to explain this in a follow-up interview, she responded, "I would recommend [inclusion of 3D modeling] in a degree program as it provides an understanding of how hard it is to create these things." (S73 interview) She recommended courses related to game design for similar reasons. "An understanding of the concepts of gaming, including the question why which is what keeps kids playing the game, is vital to the success of a program. Adapting concepts into a game format and then being able to transfer the knowledge to real life is a tough road" (s73).

#### *4.6.2.8 Project Management Skills*

Project management related skills to be developed in graduates include the ability to break a problem down into manageable pieces and the ability to balance budget and time constraints. One interviewee recommended that students be given planning exercises to practice these skills, after which students would be required to design and develop the project.

> Perhaps assign a "planning" exercise for a student to outline a project of his/her choice – first assuming unlimited resources, then applying budget and time constraints so that the client knows the optimum, recommended and minimum possibilities for their project. (S74 interview)

#### *4.6.2.9 Self-learning skills and outlook*

Participants recommended encouraging a positive attitude towards acquiring knowledge and new ways of thinking. Graduates should enjoy staying on the cutting edge (S62) and "quickly pick up new technologies as they emerge" (s56), as well as having the "potential to get acquaint to new environments quickly" (s11). They should be open to new experiences. One participant warned that this is "a necessity – the field changes rapidly. If change is outside your comfort zone, consider a different career" (s43 interview). In addition to learning new things, "being able to let go of the past and be able to move forward is a really important kind of orientation to have, because things change so quickly" (I1).

This outlook is not restricted to technical matters. "They must have a desire to discover the new, because we are constantly looking for new ways to help kids learn" (S54). This may include new technologies, or new instructional approaches. "Instructional models change regularly to adapt to new ways in which users access and use information in academia." (S69).

### *4.6.3 Passion for this work*

In addition to skills and knowledge, it is important for new graduates to bring a passion for working in this area. One survey participant recommended the following traits in an ideal graduate: "an interest in changing education methods an interest in programming an interested in designing educational games and simulations [sic]" (S63). Another recommended "love for making something perfect (or at least better)" and an "interest in technology and learning new things"(s62).

#### *4.6.3.1 Other aspects*

Other traits that should be developed in a graduate include:

- Able to approach information as a learner

- Able to play many roles
- Attention to Detail
- Balance budget and deadline constraints
- Be a good teacher
- Can evaluate claims and evidence
- Can see big picture
- Can work across domains
- Entrepreneurial
- Focus on learner's needs
- Focus on the outcome
- Focus on the process
- How to think out of the box
- Know how to scale design up to be used by lots of users
- Leadership qualities
- Object-oriented thinking
- Pushes boundaries
- Technical aptitude
- Understand EDUCATIONAL aspects needed in software

### *4.6.4 Program Curriculum*

The following sections include recommendations made about courses and experiences that should be included in the ideal Bachelor's program.

#### *4.6.4.1 Computing foundations*

These were the most frequently mentioned specific course areas. Fourteen individuals referred generally to foundational courses by terms like "programming", "computer science basics", or "solid grounding in basic computer science principles" (s12). The most frequently mentioned topic was "Database design". Other specific topics mentioned include[7]:

- Algorithms
- Computer organization
- Concepts of programming languages
- Datastructures
- Efficient programming - does not assume unlimited resources
- File handling

[7] An external reviewer agreed that the topics mentioned in this category are generally considered to be foundational , while the topics included in "Computing specialties" are areas of specialization.

- Networking
- Object oriented design and object oriented programing
- Operating systems
- Processes (unclear)
- SE foundations
- Systems analysis and design

Three individuals mentioned the names of specific programming languages, but these appeared to be examples rather than recommended topics. Others mentioned the importance of not focusing too heavily on any specific programming language "Computing should provide… strong foundational skills not just a language. Then any language can be utilized" (s23). Another explained, "There is a lot that is unstable and will change, but the way you think won't. So I think it's more important to get the basics in rather than any specific technology" (I1). Others suggested the importance of having exposure to both depth and breadth of experiences.

#### *4.6.4.2 Computing specialty areas*

The following specialty areas were each mentioned individuals or a small number of participants.

- 3D modeling
- AI
- Game design
- Hardware design and experience
- How to use computation for RW problem solving
- IT strategies and analysis
- Security
- Technology (unclear)
- Web-specific knowledge and skills

Some participants probably mentioned these particular topics because they play an important role in their specific position. But one survey participant succinctly explained his reason for including "in-depth courses in various application areas (graphics, database, AI, etc)" along with a number of "computing foundations" in his survey response: "Philosophy: the degree program

should provide a broad introduction to many aspects of the field, along with in-depth study in one or more advanced areas." (s37)

#### *4.6.4.3 Instructional foundations*

As was the case with computing foundations, many individuals (9) generically recommended "ID intro courses" or "ID theory." A fair number also recommended courses relating to educational psychology or learning theory (8). Other related topics mentioned by individuals or a small number of participants included:

- Assessment and measurement
- Create educational objectives
- Critical analysis of instructional media
- Design for different environments
- Evolution of instructional technology
- Formative and Summative evaluation of software
- How to get requirements from students (specifically)
- Instructional media and strategies
- Learn to identify subject-matter patterns
- Learning goals analysis
- Learning Object reuse
- Provide a rationale for practical decisions
- What learned in ed transfers to SD

However, fewer responses related to instructional and educational skills than to technical skills. One possible explanation may be explained by the response of one survey participant: "To build programs, students don't really need a prescribed curriculum. I believe if they want to become specialists, then completing coursework in Computing in Education / Ed Tech, etc. would be beneficial" (s67).

#### *4.6.4.4 Education specialty areas*

A few participants mentioned other specific specialty areas within the realm of education, including:

- Educational Administration

- Adult education
- Simulation design
- Domain-specific teaching skills ("Maybe a year on skills and processes for teaching maths" (s17))

#### *4.6.4.5 UI*

Interface Design or Human-computer interface design were also frequently mentioned skills. Other related skills mentioned included usability, user testing, user profiling, and visual design. One interviewee who had taken a degree in this area indicated it was "absolutely" useful to him. He explained that this background included "a lot of prototyping and going through the process of user testing and user profiling, um…imagining, creating scenarios, and imagining users... people who might actually use the product and defining what those are specifically…" (I2). One survey participant had perhaps had poor experiences with classes in this area, and warned that "Classes on user interface design [to be taught] by someone who knows what they're talking about--failing that, just have students take some industrial design courses." (s40)

#### *4.6.4.6 Other important foci or courses*

Other specific foci or courses recommended include:

- Art and Visual Design
- Business and Finances
- Design theory (specifically mentioned)
- English (general)
- Experiential Analysis and Planning
- Foreign language
- Game development
- Human Performance
- Intellectual Property Law
- Knowledge Elicitation (not sure what this means)
- Marketing, Market research
- Math
- Philosophy and logic
- Project Management

- Social Sciences
- Understanding of Research

Recommendations for social sciences often referred to the importance of these types of courses in developing "people skills". Interviewees who mentioned this area typically indicated that a variety of different courses may fill this role. Specific social science areas recommended included anthropology, psychology, sociology, and generally "how people think".

#### *4.6.4.7 Practical experiences*

One of the most heavily discussed areas were practical and realistic experiences, which were generally described as being extremely important to developing a students' ability to succeed in the field.

##### 4.6.4.7.1 To be learned from practical experiences

Several discussed the importance of having a good balance between theory and practice. "Program should encompass a mix of theory and practical application" (s51). Others called for less theory and more practice. One participant discussed how theory she learned in a degree program she is currently enrolled in already has impacted her real-world job practice. Others explained that practical experiences are important in providing a context for applying theoretical principles, and helping them see how the various things learned all relate to one another.

Participants indicated that many important skills can be best learned through practical experience. These include how to work as a team, including practice in the necessary communication skills, group dynamics, and how to work well together. Specific lessons include learning to identify others' strengths and learning to collaborate electronically. Other practical experiences include gaining experience with working with users and clients, the difficulties that occur in real-world settings, and the ability to scope, prioritize, plan, and manage a real project

from end-to-end. A few discussed topics relating specifically to instructional software, including instructional design practice and the need to balance educational and technical requirements.

Many programming skills and testing practices could be best learned and practiced through realistic projects and other real-world experiences. This includes the ability to write maintainable software, and to maintain it over time – a topic that participants found is rarely covered in past or current programs. Other specific skills technical include bottom-up development, requirements gathering, and documentation. Participants indicated a lack of experience with specific practical tools and methods in their own educational experiences, and recommended incorporation of topics including build process, continuous integration, practice with IDEs or SDEs, release management, and version control.

Higher-level lessons include knowing what is possible, learning to multitask, and understanding what you will actually do with the degree once you graduate. As one interviewee explained, "think too many times students enter into a program not really knowing what they are going to be exposed to and what they could do with this training. (I know I was one of those students and somehow lucked out with the right choices. Others are not so lucky!)" (s73 interview). It is important for students to experience the complexity and difficulty of realistic experiences.

> An ideal program would bring existing large-scale real-world applications and infrastructures into the curriculum. Current traditional college degrees have a tendency to teach students how to ride a tricycle. This doesn't teach them how to work on a busted jalopy rigged with a car bomb. (S66)

#### 4.6.4.7.2 Projects

Projects were a frequently mentioned type of practical experience. Projects can vary in size and complexity. "Instruction and early labs may begin with simulated problems and clients but should eventually include both individual and group real-world projects with real clients" (S74 interview). The fundamentals do not necessarily have to be mastered before beginning the use of projects. As one interviewee explains,

> It depends on the problems on the problem you give them. A person could be learning to program for the very first time, and you can still give them projects that are within the scope or completion. You just have to be careful what you ask of them. I mean, you can't really throw out a problem to someone that involves complex database solutions if you haven't taught them the concepts behind a database yet. (I 7)

Working on a project for an actual client also allows students to gain experience with complexities that would probably not occur in an artificially created project. Working on such a project can be very motivating.

> I did a Computer Science course that had lots of practical content. In the final year I had to write from scratch a system that the careers department used. It was partly a knowledge based system that took the know-how of the careers people and encapsulated it into a set of rules. It was designed to save them a lot of the drudgery surrounding allocating interviews and if possible sort out some of the more intractable interview slot issues. This is the first program that I had ever worked on that needed lots of input from a non-programmer and quite of lot of thought to come up with algorithms that worked. It did work and worked much

> better than was expected. It only failed in about 5% of cases and saved the careers department a lot of time. It is the success of this that first made me think that I could solve any problem put in front of me. (S56 interview)

Other participants emphasized the value of team projects. One interviewee mentioned that although she prefers to work on her own, she learned a lot from working on a project with a remote team.

> Learning to work in teams online was a whole different experience that has allowed me to expand how I address collaborative assignments electronically. It is a different kind of beast and one that many need to learn. The complaints are the same but the challenges are different. Identifying each other's strengths is vital to the success of any course and/or organization. This is where one learns who can be trusted, who will be able to meet the requirements and where things may have to be shifted. As this is a life skill that everyone needs, this is truly impressed during the online educational experience. (s73 interview)

Another interviewee discussed the advantages of having different aspects emphasized in different projects.

> There are advantages and values in different kinds of projects where you tackle the job yourself, versus two-man versus a small team. Because, in a small group, you need to learn group dynamics, splitting up problems and coordinating results, and communication. In a two-man project, it's much easier to just split things and then one person goes charging off and you go charging off and you just put it together. Communication isn't as important in a two-man project as it is in a multi-man project. (I 7)

> And then in a one-man project, because you are doing everything yourself, you have a lot more flexibility and control over the order in which you do things, and how you tackle the problem, and that is very educational in and of itself. Because in a lot of situations, you will be called upon to manage an entire project by yourself, from start to finish. You have to know all aspects of the SD process. You have to understand how to collect those requirements, and understand them and refine them, be able to take those requirements and build a design from them, and then from there go into to the actual development of the software [which must be done] in a modularized, maintainable type of way, the skills of actually writing software, and then going about testing the results at the other end, and potentially even communicating the results back to your client, your teacher, or whatever. So, there's is a lot to it. (I 7)

Another aspect of real-world problems is the pressure to accomplish goals. One recommendation is to include competitive projects as well. "I would recommend competing groups against each other…Because… in the real world you get a lot of pressure to get things done, and that will simulate the sort of pressures you will find when you hit the first job." (I1)

Only one participant brought up the difficulties of organizing such projects. "[Real-world projects and experiences are] helpful, not always possible because of costs of tools; better to learn in an internship" (S43 interview)

#### 4.6.4.7.3 Other types of practical and real-world experiences

The most commonly mentioned experience after projects are internships, which give students real-world experiences. One participant mentioned that the internship or work experience should be in multiple areas (S60).

Integrating experience realistic examples is also important. This can include the use of cases and critique of existing software. Examples and assignments should be large-scale and realistic. "Have one (or more) large examples that are used throughout the different courses instead of many tiny 'schoolish' examples" (S39). Examples and project should include experience "in real domains across a number of different target audiences" (s51). It should also include use of tools used in real-world software development (such as those described in the previous section).

Access to real subject matter experts is also a valuable experience. "They would provide clarification for questions arising from the ID's research on the topic and suggest other references" (S74 interview).

Other types of experiences suggested include:

- Access to real world software development shops
- Budgeting exercise
- Cases
- Capstone
- Labs
- Mentorship
- Portfolio
- Simulations
- Students present or publish their own work
- Student-choice exercises
- Studio-like development courses

### *4.6.5 Program traits*

Participants indicated that it is important for programs to be flexible.

> People are different. There are people who can tackle pretty substantial projects in the course of one semester and accomplish meaningful results, whereas there

> are other people who need more time with the fundamentals of the problem, rather than the details of the polish and the things like that. (I7)

One way to add flexibility to a program is to allow specialization.

A number of participants discussed the value of having cross-disciplinary programs. As one explained, "Education encompasses many areas and this needs to be reflected in the degree. The social sciences, instructional design, computing (heavy in problem solving), and usability should all be covered" (S23). Another suggested that the designer of an ideal program should "understand the need for cross disciplinary/faculty approach. Consultation in the development of educational based resources. Appreciation of the different requirements in the different disciplines" (S64).

One participant indicated that the degree should be "managed collaboratively by CS departments and software industry managers" (S10). Another recommended experiences to help students learn how the education industry works. Finally, one indicated that current programs are not very useful for some students.

> [A degree with programming or other technical focus] is extremely important for a majority of software developers, but a good ratio have intrinsic passion and drive that motivates them to become highly skilled in programming languages and other technical aspects of software design/development. Formal training tends to aim well below the skill level of these people. (S66)

#### *4.6.6 Issues with question*

##### *4.6.6.1 Need for a Master's degree?*

Several participants were surprised that the question was aimed at a Bachelor's degree. I had the opportunity to interview one of the survey participants who initially responded "Strange

question since almost everyone in this field [instructional design] has at least an MA" (S 43). Interestingly, in our follow up discussion she became enthusiastic about the idea of a Bachelor's degree in Instructional Design.

> I emphatically do not believe a graduate level degree is required – yet am not aware of undergraduate programs. It's something I'm very interested in promoting – we need more instructional designers and I think we could do a good job of preparing people at the undergraduate level. This is especially important since employers don't want to pay what a person with an MA expects to earn. There are some certificate programs out there and I think they represent a good compromise for the time being. (S43 interview)

One participant included description of both a general Bachelor's degree which would provide foundational skills, and a masters degree which would actually prepare students for a job.

> lots of general education to provide a broad framework for learning to think. I really think that the bachelor's degree should be very broad and then the person moves to an Masters to actually get the skills for the job. The bachelor's degree is about learning to think critically, problem solve and learn to interact with a diversity of people in teams. The Masters is about the JOB. (S68)

Echoing this sentiment, one interviewee suggested a model similar to law school, in which students could pursue an undergraduate degree in a numver of differnet areas which could "expose [them] to important principles that are going to be key to an aspect of instructional design work". This would be followed by specialization in instructional design or a related field at the graduate level.

> Once you get to graduate school, I think you need a degree in instructional design or learning sciences, or at least a certification, to accompany, for example, a computer science degree or media production degree. I think by then, you need to learn about instructional design. (I 9)

#### *4.6.6.2 Other issues with the question*

Two survey participants indicated they had an issue with the set of questions relating to the development of "an ideal bachelors program". They indicated that the question itself did not make sense or was useless because the goal was not sufficiently clear. "There is no ideal program in the absence of a specific set of goals." (S32).

As over a third of the participants who reached this question (30 of 74) did not opt to respond to the open-ended portion of it, I do not know whether others found the question annoying or difficult to answer. By this point in the processes they may have already completed a large number of questions and may have simply not wished to take the time to type in a response to an open-ended question.

# 5 Discussion

## 5.1 Backgrounds: Multiple paths

The findings of this study indicate that people working in the area of educational software design come from a variety of backgrounds, which include multiple formal educational paths and a wide variety of life experiences. The findings suggest that people from all backgrounds can and do take on the entire spectrum of roles included in my definition of an "educational software designer", although some roles are more likely to be played by some people than others.

It is not surprising that those with a background in Computing are somewhat more likely to play technical roles, especially hands-on roles such as programming and database design. However, it is somewhat surprising that the most extreme differences seen are in the roles of software architect and technical requirements gathering/generation. This may be because these roles are usually played by very experienced software designers. The only statistically significant difference between groups on the high-level design and low-level design roles is between those with a background in Computing and those with a background in neither area. I cannot explain this finding; since these are quite technical roles; I would have expected that those with a background in Computing would be much more likely to play this role than those with a background in ID/education only. A comment from one of the participants when reviewing this section as part of member checking would appear to confirm my feeling. As he explained:

> I can tell you why high-level design is done by those with a background in Computing: It is all to do with what managers expect these roles to have and they expect them to have a degree and for it to have been in Computing. It is only after many years of experience that someone can dismiss what degree they have

> and just point a manager at their body of experience on their CV. If you do not have a degree in computing you will not even get an interview for many jobs and if you do it is because your CV has 15+ years of experience showing that you are able to do the job. (S56, member checking)

It is possible that those with a background in instructional design may have interpreted "high-level design" as being related to the instructional design rather than the developer-level design.

It is interesting to note that there is no statistically significant difference between groups in the area of web design and development. This may indicate that this is either easier to pick up than other types of development skills (which is certainly suggested by interviewees, as they describe "front-end" or "web" development as being less technical or difficult than "back-end programming"). However, these are also areas that are included in the IBSTPI standards for Instructional Design (Richey, et al., 2001). It could be this is truly a cross-over area between the two main formal educational paths discussed. Unfortunately the term "web development/design" is somewhat broad, as it might include activities such as designing in Adobe® Dreamweaver® or other tools which provide a lot of assistance in the more technical aspects of web development, but may also include complex Javascript, Java applet development, CGI programming, and so on.

What is perhaps more interesting is that the area of user experience design is nearly identical across all four groups. Fifty-three percent of all participants indicate that user experience design is a part of their job, which makes it the most common role of all of those included. Only a minority of participants indicate that this topic was well covered in their formal education, indicating that many who play this role may be partially or completely self-taught.

Another surprising finding is the lack of significant differences between those with a background in Instructional Design and those who have a background in both fields or in neither field. What might this mean? Although "ID/Ed" and "Both" are the smallest of the groups (consisting of 13 and 12 participants respectively), the statistical procedures employed should be sound with this number of participants (Chi-square requires at least 5 subjects in each cell and there were more than 5 in each cell in all cases). Those who had taken courses in both areas and those who had taken courses in neither area were exactly as likely as each other to have a programming role and were very similar in their response to the database design role. This leads me to wonder exactly what is being learned by those who have some experience in both areas. If those who have had educational experience in both areas do not differ from those who have not had any experience in either area with regards to very technical responsibilities, perhaps many of those who had taken some courses in "both" have focused primarily on Instructional Design, with just a few Computing courses.

Overall, it appears that people with any educational background may play any of a range of software design roles. One thing this type of study is not able to show is the quality or complexity of the software being designed and developed by participants. Therefore, I cannot determine whether roles are played equally well by participants across all backgrounds. It is also possible that although people from all groups indicate they are involved in "programming", the depth or complexity of the code they write may vary depending on the need of the specific project. For example, those without a technical background may rely on others in the team to do more complex programming, or may be more likely to build software on top of a platform which abstracts away some of the complexity of programming, allowing them to focus primarily on the user interface and instructional elements of the design. Conversely, those with a background in

software design may rely on those with instructional design or educational backgrounds in large teams, even if they play some part in the instructional design-related decisions. Software created by individuals with a software design background who start their own company may not require a lot of domain-specific knowledge, as one interviewee explained while describing the knowledge he needed in order to build lecture-capture software. Or, they may focus on a narrow area of their own expertise, as did the interviewee who wrote software focused on classical languages. However, other interviewees were clearly comfortable developing education-related software based on their own personal experience rather than a formal education in instructional design.

## 5.2 Instructional Design Education and Preparation for Management

Instructional design and management are not considered software design roles based on my definition, but were included because interview findings indicated that software designers in this particular field are likely to play these roles as well. It is not at all surprising that those with an educational background in instructional design are more likely to play an instructional design role than those without it. Interview findings indicate that instructional designers who work in software design may still have a primarily instructional design-related position, with additional responsibilities that fall into the area of software design, often those related to design rather than implementation. It is however worth noting that many of those who do not have any education in instructional design currently have instructional design-related roles or have previously done this type of work, indicating that a formal education is not required to function in this area.

The literature and standards would seem to assume that instructional designers will be managing projects related to instructional materials, including the development of educational software. However, the only statistically significant difference between the groups relating to

the supervisory role is between those who have experience in neither area and those with a background in Computing – a 40% difference! A glance at the numbers shows that those with a background in neither area are also more than 10% more likely to be supervisors than those with a background in instructional design and more than 20% more likely to be supervisors than those with a background in both areas. I cannot immediately explain this with the available data, though it may be that those who come from outside rose to a supervisory role because of their content expertise or because of other aspects of their background (e.g. an MBA degree). Several interviewees who came from unrelated fields entered this industry by inventing a new product and starting their own business. However, among survey participants with no educational background in either field, only three have what I consider executive positions, and none work in very small companies (15% work in companies with 6-20 employees, 15% in companies with 21-99 employees, and the majority, 62%, work in organizations with 500 or more employees). Three are faculty members.

These findings would seem to contradict the notion that projects involving instructional design (such as educational software design) will be managed by instructional designers. In fact, the instructional designer may be one expert among many in a team managed by someone from a background in another area of expertise, or someone with no background in any design field at all. Participants who have management experience discuss the importance of bringing together expertise from a variety of areas and of understanding the unique aspects of each of these areas – including visual design, video production, and other specialized fields, as well as software design and instructional design.

There are a number of issues with assuming that students will generally supervise projects after graduating from a particular program. An obvious issue is that entry-level

positions generally do not include direct supervisory responsibilities, which may lead to surprise and disappointment on the part of new graduates. A design and development team will inevitably have to make trade-offs in design decisions based on concerns and constraints at many different levels, and the responsibility for making these tradeoffs may not be in the hands of the instructional designer. Rather, supervisors, project managers, or the group as a whole may make these decisions. This is not to say that project management skills are not important to learn. Participants in this study clearly indicated that these skills are in fact crucial on the job. Project management skills are important at many different levels; each individual on a team must be able to understand how to budget time and resources, and be open to understanding the impacts of time and budget constraints of other aspects of the project on their own design work.

Another potential issue for those who do end up in management is the lack of awareness of the amount of design effort that goes in to other elements of a project, and how design constraints outside of those directly related to instruction may influence the overall design of a product. Unlike in class projects, in which instructional design students often work in a group with peers who are also preparing to become instructional designers, new graduates will need to work together with colleagues from a multitude of backgrounds. An assumption that a background in instructional design with a course or two in web design or media development is sufficient to understand all aspects of what goes into a software development project can lead to false assumptions and frustrations on all sides. For example, certain types of features may be more difficult to develop on some platforms than others, so if a software design team is limited to use of a particular platform, some options that would seem desirable from an instructional point of view may be difficult or impossible to incorporate into a product built on that platform. Switching to a different platform could result in a complete rework of the software design and

recoding of the entire product. This may not be apparent to someone without the relevant expertise.

It is also not helpful to assume that ADDIE or another instructional design model will or can guide an entire design and development project. In fact, software designers must go through their own extended design process to produce high quality software, including developing software architecture, detailed technical requirements, and often high- and low-level design documents before coding begins. These are the precise roles that participants of this study with an instructional designer education were least likely to play or have experience in! Similarly, testing practices in software design cover different types of issues than those learned in instructional design – and both are crucial to developing a high quality product. Testing is an area often under-appreciated and not sufficiently budgeted for, making it even more crucial for professionals to understand the varieties of types of testing and the importance of each. It is likely that similar gaps exist between the preparation of instructional designers and an understanding of other specialty areas such as graphic design, videography, etc.

Those with a background in Computing should be similarly cautious about making uninformed instructional design decisions within software they are developing. Software designers are often cautioned to ensure that a piece of software be designed with the intended user in mind, rather than designing what the software designer himself would like to use. Study participants indicate that this concern is especially crucial in designing educational software. Having attended school does not necessarily make one an expert in education. Nor is reading about educational psychology necessarily sufficient to make sound instructional design decisions. The types of testing taught as part of software engineering or even interface design related courses may be sufficient to determine whether users can successfully move through a

piece of software, but do not verify whether a user has actually gained useful and transferable skills and knowledge from a program intended to be educational.

## 5.3 Interpreting the Gaps and implications for degree programs

As Walker (2010) points out, not everything can be included in a four year curriculum. Walker recommends that curriculum designers set priorities in order to assemble a realistic list of core topics and then allow students to specialize. Walker also stresses that a curriculum must provide "a foundation for long-term study, professional involvement, and productivity" (p. 21). Participants in this study indicate – not only by their own opinions and suggestions, but also by the demographics of the group as a whole – that there are multiple possible paths to a career in educational software design. Participants' near-universal emphasis on the importance of critical thinking, problem solving, and the ability to learn on one's own would seem to back up the importance of developing the types of skills needed to provide just such a foundation.

The overarching message conveyed by the comparisons between what is important on the job and what has been covered by participants' previous formal education is that software designers do not get all of the skills and knowledge they need to succeed in their careers from their formal educational experiences – regardless of the degrees held. There are two possible explanations for this. Clearly, not everything needed on the job was learned as part of a formal degree program, which could indicate troublesome gaps in degree program curricula. However, findings from this study indicate that non-formal, self-learning experiences are a normal part of software designers' lives, as would be expected based on Continued Professional Education literature (Daley, 2000; Driscoll, 2000; Knox, 2006). Certainly software designers, like other professionals, continue to develop expertise once they are on the job (Cross, 2004).

### *5.3.1 Implications for existing degree programs*

An important question for existing Computing and Instructional Design programs to ask is, what really needs to be included in formal educational programs, and what can be left for practitioners to learn on their own? Investigating the gaps between what is important on the job and what was learned as part of participants' own formal educational experiences, together with a detailed analysis of recommendations for an ideal degree program, has pointed out some areas which appear to be good candidates for emphasis in degree programs.

Statements from participants of this study would seem to imply that critical thinking and the ability to communicate well and learn on one's own are a crucial basis for beginning a career in this field. Although participants request more focus on practical skills, the skills they ask for are often at a high level: "experience with a full lifecycle of a project"; "experience maintaining software"; "experience with version control". Again, rather than focusing on a specific technology or content area, participants appear to be suggesting broader experience with real-world applications which they can build upon once they enter the workforce.

This is not to say that there is no place for theory – many participants recommended theoretical "basics" in both Computer Science and Instructional design. These findings mirror those in an earlier study of software designers working across industries, in which participants strongly recommended a foundation in "the theoretical basics" of Computer Science and related areas, followed by multiple intensive realistic practical experiences (Exter & Turnage, 2011). However, the current study reflects a more complex set of recommendations by practitioners whose own backgrounds and roles played varied widely.

The importance of covering Instructional Design and Computing foundations in an ideal degree program varied across participants, though participants were likely to recommend

backgrounds similar to their own. One surprising finding was the emphasis put on Human-computer interaction design and User experience design, which may be alternative educational paths in their own right as well as areas that may be included in either a Computing or Instructional Design degree.

It is interesting to note that participants' recommendations align with a lot of the recommendations in the variety of Computing- and ID-related standards. Participants' concerns also match the industry concerns mentioned in the IEEE/ACM Joint Taskforce's Computing 2008 standards very well (ACM and IEEE Computer Society, 2008), as well as the justifications for changes made in the IBSTPI competencies (Richey, et al., 2001). For example, the IEEE/ACM Task Force's Computer Science curriculum standards include a set of "transferable skills" such as communication, teamwork, self-management of one's own learning, and professional development. These are similar to IBSTPI's "Professional Foundations", which include reference to effective communication and the need to update and improve one's skills, knowledge, and attitudes and the "Implementation and Management" skills, including the ability to plan and manage projects, and promote collaboration (Richey, et al., 2001). Both of these would seem to address participants' concerns about the need to develop communication and team-work skills, as well as preparing students for self-learning and project management related tasks. The IEEE/ACM Task Force's standards, like the other Computing-related standards discussed in the literature review, clearly cover the "computing foundations", and the IBSTPI standards likewise cover the areas that are referred to in the findings section as "instructional foundations". All documents reviewed either allude to or directly address the importance of the development of practical skills.

If all of this is true, why are participants in this study, along with others mentioned in the literature review, still demanding that these changes be made? Although the majority of my participants did not recently complete an undergraduate program, many of their comments revealed concerns they have about recent graduates. I had hoped to learn more about the state of current programs during phase 3 interviews with recent graduates, but unfortunately few survey participants responded to my request for follow-up interviews. Three of the four who responded are currently enrolled in formal education programs, but each of them is a PhD student. Although they discuss "real world projects" and others topics I am interested in, the projects mentioned were generally research related (although at least one participant's research involved design and development work).

Difficulties in implementing the types of changes in curriculum necessary to provide more real-world or practical experiences may have many sources. Faculty members who have not recently worked in industry or who may never have worked in industry may not have direct experience to provide them with the context that makes it so clear to others why this type of teaching is so valuable. They may prefer to teach in the way they were taught, or may fear giving up too much class time that could be used to teach theory and cover the wide range of required standards.

Many current faculty may indeed strive to integrate practical experiences into coursework through team projects, but these projects may be, in the words of one survey participant, the equivalent to teaching students to ride a tricycle, when what they will really need to do in the work world is "work on a busted jalopy rigged with a car bomb" (S66). There may be structural difficulties in implementing the types of changes recommended by my participants, which require complex projects that would ideally continue for longer than the duration of a semester.

Goals like "maintaining a large body of code over time" cannot be realistically fostered by the typical model of 2-4 class projects throughout a semester. Standards do not indicate how multi-semester or multi-year projects can be achieved in a traditional higher-education setting.

In a recent conversation on this topic, Dean Schnabel (Dean of the School of Informatics and Computing at Indiana University and Professor of Computer Science and Informatics) mentioned two practical concerns that would make cross-course, multi-year projects difficult to implement (R. B. Schnabel, personal communication, June 22, 2011). In order to provide a multi-year project, students would need to remain in a cohort across years. This type of cohort program is difficult to organize because undergraduate students frequently transfer in or out of a program, study abroad, or structure their courses around work obligations. Dr. Schnabel also noted that Universities should admit to some limitations in the current system; because faculty members are very autonomous and students take individual courses from a variety of faculty members throughout the program, it would be difficult to coordinate such that a single project could be pushed through multiple courses over time.

Attending a conference on Computer Science Education (ACM SIGCSE's annual conference) allowed me to attend sessions and have informal conversations with faculty members who strive to use just these types of techniques in their own courses. Some of the best examples of complex, lengthy, real-world projects seemed to come from faculty of small, liberal arts colleges. These faculty members enthusiastically recounted their student's responses to these experiences, and discussed what they would do differently in future projects. Similar presentations from faculty from large research universities were often presented in a more research-like format. This led me to wonder whether these faculty members could keep up their initiatives when they no longer "count" as research projects. Faculty members from large

universities also discussed the challenges in providing these types of activities to large classes of undergraduate students. While university programs may be better able to provide such experiences to their smaller groups of Masters students, findings from this and other studies, as well as guidance from the standards, clearly points out the importance of these experiences for undergraduate students.

One example of a program which was able to offer a more extensive experience as part of an undergraduate degree is discussed by Nurkalla and Brandle (2011), who presented a curriculum model for teaching software engineering through a 4-semester long "Software Studio" sequence. Within this sequence, students take four credit hours in each of four consecutive semesters, including a weekly seminar, readings and related online discussions, classroom lab time, and an expectation for an additional eight hours per week spent on team project work. Students are required to work together within their teams, create high-quality code, log their work, and present what they have learned in both written form and in formal presentations. Students are exposed to experiences in working with real customers, maintaining code over time, and in playing a variety of roles as they gain additional responsibility across the four-semester period. One caveat to note: not all computing students at this institution take this course. Students who wish to participate in the "Software Studio" track must request permission from the instructor, who interviews students to determine whether they "are self-starters who demonstrate the ability to perform well without constant oversight" (p. 154). These students must also have sufficient "software development experience, performance in other classes, and… personal maturity" to indicate potential success in this type of environment (p. 154).

In discussions with some of the SIGCSE members who have worked hard to incorporate substantive projects and other real-world experiences into their curriculum, I learned that they

work closely with industry partners, actively implementing changes recommended by a board of advisers from industry on a frequent basis. Having strong and positive ties with industry partners can provide another important advantage: access to internships and/or the ability to work with organizations to develop projects for students to tackle as part of coursework. Developing a report with a "client" and working together to design a project that is feasible for students to accomplish in a set time-frame while still presenting realistic challenges can be difficult, and maintaining ongoing partnerships with organizations that have an interest in fostering students in a particular college program could be a real asset.

Based on experiences with developing an informatics capstone project course, Dr. Dennis Groth (who organized an Informatics Capstone course at Indiana University for over ten years and has written about this experience) found that students were also very motivated to work on projects developed for industry partners, and that the industry partners found work on such projects to be a good indicator of students' future performance (Groth & Hottlel, 2006; D. P. Groth, personal communication, July 6, 2011). The design of this two-semester-long capstone course allows for formal lectures on process-related topics that are common to all projects, but also relies on students to pursue just-in-time learning related to their individual projects, providing another valuable real-world experience.

Wolz, Cassel, Way, and Pearson (2011) suggest another resource that should not be ignored: faculty and students in other disciplines. Their model of "cooperative expertise" involves a partnership between multiple instructors, each leading his or her own course, during which students cooperate towards a common goal. This model allows instructors to overcome the constraints of teaching in a field with a broad range of areas of expertise. It also enables them to provide an authentic experience with designing and developing large-scale projects. Students

have the opportunity to develop their soft-skills in ways that cannot be accomplished when only working with other students with a similar background. In the example discussed in Wolz et al, three courses were integrated across two institutions: a software engineering course for Computer Science majors; a multidisciplinary video game course aimed at Media and CS majors; and an interactive storytelling course for Media majors. These courses were each taught by a separate instructor, and each had its own syllabus, allowing the instructors to meet institutional requirements within their own courses and course syllabi. The faculty members tied the courses together by tailoring course objectives and goals to the cooperative agenda and by planning a schedule of deliverables for each student group. As in a real-life project setting, each student group was dependent on deliverables from the other groups to meet their own schedule. The authors indicated that this program met many of their goals and recommended this technique to allow universities to provide realistic collaborative experiences. The students from the Computer Science program appear to have benefitted the most from this cooperation, possibly because those in the partner classes were already engaged in multidisciplinary programs.

The EPICS model, developed at Purdue University and later replicated at a number of other institutions, uses a service-learning model which allows students to participate in large-scale, multi-year, interdisciplinary real-world projects in aid of a non-profit institution or cause (Coyle, Jamieson & Oakes, 2005). Within this model, community partners are carefully selected to ensure that a proposed project will be supported by the partner throughout the project's lifecycle. The project should also significantly benefit to the community. Finally, it should pose a challenging but reasonable task which is of an appropriate scope (that is, it must require a design and development life-cycle which will span multiple semesters).

After the project has been established, a cross-disciplinary project team is formed (including students from multiple engineering disciplines as well as students from non-engineering disciplines, potentially including a range of topics spanning from the social sciences and education to such diverse fields as English, nursing, visual design, forestry and natural resources, chemistry, and management). These teams are "vertically integrated" by mixing freshmen, sophomores, juniors, and seniors, allowing the more advanced students to spearhead the project while the more junior members will be around to continue the project into future semesters. This allows projects to continue for multiple years, as new students continue to be recruited. Student teams spend the first semester meeting with a project partner to define the project and develop a project proposal. In subsequent semesters, the team will develop a prototype system to present to the project partner and revise as necessary. Development continues for multiple semesters, until a project is developed that meets the needs of the project partner. Finally, the system is deployed, at which time the team provides training, collects feedback, and make changes based on that feedback. Students are responsible for supporting and maintaining the project through future semesters. Throughout this period, a faculty adviser and TA meet weekly with the team to provide technical supervision. Institutional support is crucial for building and maintaining an EPICS program. Based on a study of the program at Purdue,

> The most critical elements in the success of an EPICS program are leadership of the program by one or more faculty members and support by the appropriate departmental and college administrators. This ensures that a high-quality design and service learning experience will be provided to all EPICS students in courses that are approved by the faculty. Beyond these essential elements, the level of student enrollment in EPICS depends upon a combination of degree requirements

> in different disciplines, available space, the number of potential faculty and industry-based advisers, and the teaching credit that is offered for advising an EPICS team. (p. 10).

The Purdue team found that finding sufficient partners in the community was not a limiting factor, suggesting that contacts within the community, the university, and local government agencies are all good sources for potential student projects. Therefore, institutional resources and support were the deciding factors in making the EPICS program a success.

The last two examples describe projects not necessarily related to the area of educational software. However, educational software development could clearly fit well into the types of projects described by Wolz et al (2011) and Coyle, Jamieson, and Oakes (2005).

Finally, when designing a practical course, instructors can benefit from learning from the signature pedagogies of other fields, as was suggested by Shulman (2005). In one example relevant to the Computing field, Cennamo et al (2011) made recommendations for a potential Computer Science studio based on an ethnographic study which compared student activities in HCI studio courses to those of students in similar courses offered within fields with a long history of studio education (Industrial Design and Architecture). The researchers found that students in the HCI courses were less likely than their counterparts in the other design fields to come up with truly original ideas, consider fully the place of the user's experience, or use low-fidelity prototypes or other techniques to overcome limitations in their ability to work with or test an initial design. The authors indicated that one major difference between the courses was the amount of class time spent engaged in studio activities. The three HCI courses studied include two semester-long courses which met formally for 37 hours each, and a quarter-long course that met for 27 hours. In comparison, both the Industrial Design and Architecture courses

included four-hour blocks of studio time three times a week throughout the semester, as well as providing studio space accessible to students 24 hours a day, seven days a week. The authors indicated that the "severe time constraints" were a limiting factor for the HCI courses, but they also recommended a number of changes that appeared to help HCI students in subsequent semesters move further into developing more original ideas, through an emphasis on experimentation and idea refinement. This included the use of low-fidelity paper or white-board based prototypes, and having students put themselves into the role of the user to experience their own design from an alternative viewpoint. They further recommended requiring students to generate a number of designs before settling on a specific solution. In order to facilitate this, the authors suggested that computer science instructors separate design from implementation, by not requiring that all or most designs be implemented. The authors stressed that this approach could be taken at all levels of computer science instruction - when designing algorithms, programs, or specifications, as well as when designing an interface.

Although these examples focus on Computer Science education, the complexities and challenges of providing realistic projects and other experiences in Instructional Design programs are likely very similar.

### *5.3.2 Ideal program for educational software designers*

The above discussion does not answer the higher level question asked of participants regarding an "ideal" program: what *type* of degree should be offered to students planning to work in this field (or what type of degree should they pursue)? The support for some sort of hybrid program was surprising to an external reviewer of my data, who had expected that participants with a Computing background would suggest a degree in Computer Science or

Software Engineering, and those with a background in Instructional Design would recommend a degree in Instructional Deign (see Appendix D: Notes from external review of coding).

However, there are several concerns with this idea. Is it practical? As one participant pointed out, "I don't think it is realistic for any institution to offer some sort of hybrid thing. Cause, the market is too small for something like that" (I4). Results of this study are not sufficient to determine whether this participant is correct about the size of the likely market, but this might be a topic worth pursuing.

Another participant illuminated an even more basic concern when responding to a question about the importance of a specific trait of the program: "Important for what? For my own job? Yes. For all possible jobs in a development team? No." (S36). This clarifies an issue with the questions asked; although I would like to find out what an ideal program would be like for what I call educational software designers, the question specifically asked what would be ideal to prepare someone "for your current position". I cannot be sure how others interpreted the question but some aspects of participants responses seemed closely tied to people's specific roles or products (e.g. focus on game design, etc), while others were clearly more general. Perhaps this points out yet another set of questions. Should people be getting role-specific as well as domain-specific education? Clearly people move between roles and frequently play multiple roles, and over time they may work in a variety of contexts and domains. Also, if those with backgrounds in "neither" can do the same things as those with multiple degrees in one or both areas, what does that say?

One issue with creating very specialized degrees is that professionals often move from one area to another, bringing their skills and unique perspectives with them. Over-specialization may hinder this type of movement, especially if employers begin to narrow their expectations in

hiring. There appears to be value in putting together a team with members who have both education and experiences in a variety of areas. This realization argues for caution in assuming that employees need a very specific, specialized degree to serve in a particular role. This view does not necessarily undermine the importance of degrees in Computing or Instructional Design related fields – rather, it points out the benefit of having individuals with expertise in each of these areas, as well as individuals with expertise in other specialty areas.

## 5.4 Role of experience and self-learning and implications for degree programs

Regardless of the formal educational path followed, self-learning strategies such as experimentation and learning from peers and online sources were seen as important. The high importance given by all participants on the ability to work well in teams, communicate with specialists in other areas, and perform many different roles speaks to the interactive and quickly shifting nature of this work – providing further evidence of the need to continue learning and growing throughout one's career. The relatively low importance given to "knowledge of specific programming languages" and "knowledge of web languages/technologies" by professionals who certainly must have at least some programming skills further indicates the degree to which participants feel that this type of knowledge can be picked up easily. This point, taken together with a rating of "Very Important" given by over 90% of participants for the importance of "Ability to Teach Myself", would seem to indicate that self-learning is an expected aspect of a job in this field, and something that formal educational programs should prepare students for.

As was mentioned earlier, participants' attitudes towards self-learning activities are not surprising, considering the literature on Continued Professional Education (Daley, 2000; Driscoll, 2000; Knox, 2006). Participants expect to learn on the job, and generally learn on an as-

needed basis, in order to address their immediate, real-world problems. They also learn from experience over time, developing as may be expected based on Cross (2004)'s description of the typical development of expertise in design fields.

### *5.4.1 Possible implications for degree programs*

Do the findings discussed here indicate that formal education is not really needed? After all, if those without a related degree can hold similar positions as those who have an education in the area and if experienced professionals indicate they regularly learn what they need on the job, where is the need for a formal education?

In our conversation, Dean Schnabel at Indiana University responded that the purpose of a university program is not to replicate what will be encountered on the job. Rather, a university education should provide general growth, along with a breadth of understanding within a specific field including more than what an individual would likely encounter on one particular job (personal communication, June 22, 2011). In an email message sent later that day, Dean Schnabel indicated that following our conversation, he discussed this topic with Dean Groth, Associate Dean for Undergraduate Education and Associate Professor of Informatics and Computing at Indiana University. Based on their conversation, Dean Schnabel wrote:

> In a computer science or related degree students will not only learn programming and project skills, and theoretical foundations, but also be exposed to a variety of areas of computing, e.g. databases, artificial intelligence, operating systems, networks, etc. I think the self-taught people are unlikely to have this breadth and that may limit what they can do in the computing field (personal communication, June 22).

Participants themselves did not, in general, indicate that post-secondary degrees were of little use. However, participants' suggestions did appear to indicate that the current structure of professional programs is not optimal for what they perceive as the needs of graduates. Some of the participants' recommendations may seem to suggest the value of moving back towards a more "liberal arts" model, with a focus on critical thinking, written and verbal communication and a variety of courses that would give an appreciation and understanding of the human mind. On the other hand, other requests, particularly those relating to the development of practical skills and design judgment, would seem to recommend a model more similar to Schön's reflection-in-action model (Schon, 1987). Combining these two seemingly disparate models may look similar to Brook's suggestion to "sandwich" real work experiences (including associated company training) between periods of academic education (Brooks, 2010). There may be structural challenges in pursuing such a model. The standards themselves get in the way of these types of changes – although they call for the skills that can be developed by such structural changes to the curriculum, they also get in the way of any attempt to make them by providing a list of very specific topics which must be covered, leaving programs uncertain how they can make the time to provide more "general education" requirements or more unstructured, project-based experiences. Furthermore, although this model would meet Dean Schnabel's goal of providing "general growth" in a student, it may mean that not every student will be exposed to all areas of computing considered important by the standards or by educators – depending on the nature of the "real-world experiences" encountered. For example, not all real-world projects, even substantial ones, will include in-depth experience with "databases, artificial intelligence, operating systems, networks," and other areas considered essential to a well-rounded background in Computer Science. Although these areas may not be found to be

crucial across the majority of participants in this study (and were generally relegated to the "computing specialty areas" during coding based on the frequency and manner in which they were discussed by participants), they were for the most part identified as highly important in an earlier study I conducted with participants across a number of other industries (Exter & Turnage, 2011).

Doing away with semester boundaries altogether, which may be ideal for providing more realistic experiences, would be impossible within most institutes of higher education. Even changing the content of individual courses or course sequences may be difficult when a department must contend with school-wide or university-wide policies. Unfortunately, as discussed earlier, bringing changes in piece-meal, for example by introducing team projects into individual courses, may not give the desired results. A reconceptualization of the purpose of "education" may be necessary. As pointed out by Lifelong Learning literature, education may be most useful in preparing adults to continue learning throughout their career, rather than in attempting to prepare them for a very specific role.

Such dramatic changes cannot, of course, be recommended based on this study alone. However, continuing to take a close look at these topics is one of the aims of my research agenda.

### *5.4.2 Hypothetical Degree Program of the Future*

Although the findings of this study are not on their own sufficient to argue for a reorganization of university programs, they do imply the need to carefully consider whether the current structure of university programs best meets students' needs, and whether some modifications to the structure would significantly benefit students. Based on the findings and the literature as discussed in this section, I have three main recommendations. None of these are

radical, but I believe that a change to the way we think about organizing university programs and issues such as faculty autonomy and the desire to cover both breadth and depth across a wide range of topics would be necessary to fully implement them.

The recommendations fall into three main areas:

1. Incorporate significant, complex, coordinated real-world experiences as a major component of the overall curriculum.
2. Include liberal arts courses which foster development of communication and critical thinking skills, spread throughout the four-year timeline.
3. Make trade-offs to allow for sufficient time in the curriculum to allow for these changes.

Each of these items is discussed in greater detail in the sub-sections below. The details of these recommendations are geared towards a Computing-related degree, but I believe that these recommendations could be easily adapted for an IST degree. Based on both the literature and the findings of this study, gaps in existing degree programs are similar – in type of experience if not in specific content material.

#### *5.4.2.1 Significant, complex, coordinated real-world experiences.*

As was discussed in the previous section, many programs do offer real-world-like experience through class projects and more comprehensive experiences through a 1- or 2-semester capstone or senior project course. Internships are also a good way for students to gain real-world experience. However, based on feedback from participants in this and earlier studies, these experiences may not be sufficient to prepare students for many of the most important aspects of their future jobs. Class projects tend to lack the scope and complexity of problems seen in the real world, and even year-long capstone projects lack many of the key characteristics of working in industry, including maintaining multiple versions of code over time, updating code

written by others, and working within a group with fluctuating membership and team members with expertise in different areas.

However, the purpose of schooling is not to simply replicate the experiences one would gain in the first year or two in the work world. Practical experiences should take place in a safe, scaffolded environment that allows for failure and encourages learning from both failures and successes. Experiences within such an environment should be crafted and scaffolded such that students are encouraged to see the big picture, as well as gaining specific design and development skills by working on individual components and gaining experience with parts of theory that are not really relevant to smaller projects. They will also allow students hands-on practice with topics that are not currently thought of as theoretical or discussed in great detail, including various types of testing methods and techniques, change control, etc.

Such experiences can best be provided through a multi-year project in which students must play different roles over time and must maintain designs and code created by others. This does not necessarily mean that students must be in a "cohort"; rather, students can be brought into the project at different points, with more senior students serving as team leads and mentors for the less experienced students, thereby gaining experience with management practices (as occurred in the multi-semester projects described by Nurkalla and Brandle (2011) and Coyle, Jamieson and Oakes (2005)). If at all possible, the projects should be done for a real client and within a real context.

These experiences would be set up in such a manner that students would be required to learn on their own from peers, written and online resources, experimentation, and knowing when to ask faculty mentor for help. However, the "safe" scaffolded environment would need to be structured such that that faculty mentors are aware of student progress and able to step in and

encourages students to consider things or ask the right questions at the right time. This requires active monitoring by faculty mentors who themselves have significant large-scale design and development experience. If large class sizes provide difficulties, TAs can play part of this role. However, it is important to recognize that TAs and upper classmen are not likely to have the necessary rich experience across a range of content areas that may be required at various points throughout the project. Therefore, experienced faculty members should remain involved in a level that provides them enough details about what is going on that they will know when to step in. Alternatively, volunteers currently working in or retired from positions in industry may be willing to play a mentorship role.

Although teamwork is important in providing a foundation for future work and in providing the means to work on sufficiently complex problems, student team projects have their own difficulties and drawbacks. It is important that faculty mentors and TAs recognize the potential downsides of putting students into teams and are prepared to step in when team issues go beyond what students are able to handle on their own. This will ensure that students are able to continue learning new content and other important lessons along with developing the ability to to deal with team members in an appropriate way. Weekly seminar topics can and should include topics such as working within a team, working with team members with varying backgrounds (e.g. instructional designers, graphic designers, etc), and also how to successfully work within a group in which all necessary skills may not be represented or evenly represented at the outset.

From a practical perspective, such an initiative must be organized at a departmental level, so that everyone understands what goes into the project (including human resources and implications for other courses and the overall course sequence). It is also important that all faculty members understand the load that this project places on students as well as the type of

work being done by students at various levels. This will allow other faculty to leverage what is being learned in the project in their own classes. The faculty member leading the project should also be able to count on certain core material being covered within the courses, and plan to build upon these core skills as part of the practical experience course.

Sufficient time (credit hours) should be allotted in the schedule to make it possible for students to dedicate significant time to the project. If a discussion or lecture section is included, consider making the discussion section 1 credit hour and limit time spent in lecture to one hour per week, leaving an additional significant number of credit hours as a place holder for the work students will be doing on the project. Because a large amount of work is being done by faculty leading this project, significant credit (course load) should be given for the work done preparing for and overseeing the course every semester.

It would be valuable to consider teaming up with faculty and students from other departments (as was described by Wolz et al (2011) and Coyle, Jamieson and Oakes (2005). This will help make projects more realistic for the CS students, who can develop content expertise from working with their peers from across campus. It will provide benefit to partnering departments by allowing students from other areas access to the capability to design, develop, and evaluate significant technology projects they are able to envision but not fully produce on their own. Additional faculty resources and enrollment of students from other departments or schools would help keep the course sustainable, and the cross-campus collaboration could result in interesting projects that can be showcased by university administrators. Such projects may also open up multiple avenues for various types of research and grant funding.

Although, based on participants' comments in this study, it does not seem to be crucial for the project to be in the same domain that students eventually end up working in, projects

related to education may be ideal. The Computing-related department can partner with instructors and students in an Instructional Technology department or others within a school of education to design systems intended for use in schools which can be further refined based on educational research. Such a project would be likely to be highly motivating, as it would provide complex demands from an interesting set of users. Using research data to drive improvements will help both the Computing and Education students learn more about the concept of iterative, user-focused development. Seeing the project in use by children or fellow students for an important purpose would no doubt also be highly motivating to many.

#### *5.4.2.2 Liberal arts and sciences courses spread throughout the curriculum*

Findings of this study indicate that liberal arts courses taken as part of general education requirements are valuable in providing very important critical thinking skills and creating well-rounded graduates. They can also play a role in developing skills that are currently lacking in many graduates, such as technical writing and relevant business and financial practices.

It would be beneficial to spread liberal arts courses across the four years of an undergraduate degree, instead of grouping them at the beginning of a student's academic career. This will allow students to take a decent number of courses in their major during freshman year, enabling them to decide up-front whether the major is really a good fit. Taking courses within their major courses as early as possible will also allow students to build a foundation for participating in practical experiences, allowing for the types of multi-year course offerings described in the previous section and making it possible for students to participate in internships and other important formative experiences earlier in the program. Students may also come to appreciate the liberal arts courses more as they go further on in their program, especially if they

can see the links between what is being learned in the liberal arts courses and what they are doing within their major area of study.

Within the "liberal arts" or general education requirements, I can see value in requiring students to take some courses that are the same as those taken by all students at the university, as well as offering a set of courses specifically tailored to the needs of students in the Computing program. General liberal arts courses including topics such as history, literature, sciences, and so on will help students to develop critical thinking and writing skills and a grounding in skills and knowledge across multiple domains. Multi-disciplinary or team-taught courses might be especially useful in meeting those goals. However, it would also be useful to offer a set of courses that meets some of the general education requirements and which are geared specifically towards Computing majors. These could focus on topics such as technical writing and business practices specifically related to the computing industry. Multi-disciplinary or team-taught courses could allow students to gain multiple types of expertise. These could also be carefully integrated with the practical computing courses discussed in the previous section.

#### *5.4.2.3 Trade-offs to provide curricular space*

Adding a significant amount of additional practical experience and general education requirements will of course have an impact on a 4-year curriculum, and curricular designers will need to make trade-offs between these experiences and other potential topics to be covered. As Walker (2010) pointed out, it is not possible to fit every skill or topic that could potentially be useful in the future within a four year program, and if too many topics are covered, it is unlikely that students will be able to retain everything.

My preliminary recommendation would be to lower the number of "core" Computing competencies and focus on those that are necessary to build on through carefully planned

guidance during practical experiences. It also is important to think carefully about what non-Computing requirements are necessary and valuable for most students. For example, based on the findings of this study, it may make sense to lower the total number of math requirements for Computing majors, and focus on topics such as discrete math which are directly applicable to Computing theory. On the other hand, additional Communications-related courses would be a valuable addition, as suggested in section 5.4.2.2.

One or more areas of "specialization" will allow students to dig deeper into particular areas (for example, computer hardware, 3d imaging, or artificial intelligence). Not only will this provide specialized knowledge to be used on the job, but, more importantly, it will give students an experience of delving into a set of topics in depth. Regardless of where they end up in the future, students will know they can acquire a new specialty as needed through self-study or enrollment in carefully chosen courses.

**5.5 Finally, it is important to ensure that classes are offered in a timely manner, to allow everyone to fit all required courses into a four year period. Including more topics in the "practical" experience hours which are offered each semester will help in this regard. Spreading liberal arts courses throughout the curriculum may also help, as students will be able to fit both their general education and computing-specific requirements in throughout the four year period when desired courses become available, instead of concentrating general education requirements at the beginning of the program and relying on specialized courses to be available in the last two semesters. The Role of Hiring Managers**

Although employers can be valuable partners in fostering changes in education policy, hiring practices can also have very negative consequences. As was discussed in section 2.3.1.7 of the literature review, job advertisements in Computing-related areas often contain an extensive and very specific list of required skills. By creating job requirements which ask for such specific skill-sets, they make it impossible for graduates – and schools – to argue that their critical thinking and communication skills and grounding in the foundational theoretical concepts in Computing or Instructional design prepares them for employment. Instead, these job requirements will drive schools further into pursuing the latest programing language or technique – an impossible goal since the precise combinations of specific versions of technologies asked for by employers will continue to change more quickly than students can be graduated from a program in which these technologies are covered.

Furthermore, over the duration of a career, entire new fields may become important and existing employees may begin to serve new roles before educational institutions catch up and design related programs. For example, study results highlight the importance of user experience design, an area most had no little or no formal educational background in. Study participants indicate that they regularly educate themselves in such new disciplines, although they may or may not ever return for a formal degree. They also may be required to play a new role, such as user experience designer, while continuing to play other roles, such as lead developer.

Hiring managers also do themselves a disservice by presenting such requirements for employment. As participants of this study explain, a passion for learning should be an expected trait of an individual working in this type of field. In this fast-changing world, the technology used today may not be the ideal solution for the design problem faced tomorrow. An individual with many years of experience in a specific tool or technique may not necessarily be the best

person for the job. In fact, results of this study indicate that the best hiring decision would be to select an individual with a proven track record of teaching himself or herself what is needed to do the job. Not only will this prepare the potential employee to learn to use new technologies if asked to do so, it will also make it much more likely that this individual will actively look for appropriate new technologies and recommend them for use on the project, rather than falsely sticking to a stale skill-set learned at school. "Passion for learning", "critical thinking", and "communication skills" may be harder to quantify or determine by reviewing a resume, but are likely to be much more important indicators of a potential employee's future value to the organization than a list of specific programming languages or platforms used for a specified number of years by an applicant.

In a discussion with Jeremy Podany, Director of Career Services for the Indiana University's School of Informatics and Computing (J. Podany, personal communication, July 7, 2011), he revealed that although some job ads may not make it clear, employers looking for entry-level Computer Science and Informatics students are in fact looking for just these skills. Mr. Podany indicated that most employers he works with assume that graduates have the necessary technical skills, and use interviews to determine whether these potential new employees have the character and "chemistry" needed to work in their corporate environment. The "chemistry" factor, according to Mr. Podany, relates to how well a potential employee would fit in the corporate culture, and relates to "soft skills" including work styles. These factors are not generally specifically mentioned in the desired skills list on most job advertisements, though they might be reflected in a section such as "About Us".

Mr Podany also directed me to look at "Job Choices for Science, Engineering, and Technology Students 2011", a magazine aimed at students who will soon graduate, which is

produced by the National Association of Colleges and Employers and sent to university career service offices. One article in this magazine includes a figure summarizing findings of a report done by this organization on what employers want in "the perfect candidate". This includes the following ten traits: Communications skills (oral and written), a strong work ethic, initiative, interpersonal skills, problem-solving skills, relevant work experience (through an internship or co-op assignment), analytical skills, teamwork skills, flexibility and adaptability, and technical skills. The article points out that

> The job description will provide you with a list of required qualifications – a particular major or group of majors, a specific skill set, a minimum GPA, and so forth – but employers have a substantial list of abilities, qualities, attributes, and "soft" skills they also seek in new hires. (Job-search success for the class of 2011, 2010, p. 13)

This would seem to imply that employers do, in fact, value the very skills pointed out in this study as being crucial to success in this field. However, job ads typically do not reflect this. Hiring managers should consider whether not including this information is really beneficial in recruiting appropriate employees. In the meantime, faculty and career service offices can encourage students to understand the value of highlighting these non-technical skills through cover letters and interviews, as well as helping students understand why non-technical coursework and participating in team projects and similar experiences can be important to their future career.

Finally, once employees have been hired, managers should understand the importance of providing time and resources for self-education. Brooks (2010) recommends that structured training and mentorship be provided to novice designers. This recommendation was supported

by study participants' recollections of their own early experiences and the value they derived from working closely with more experienced colleagues, and is consistent with recommendations in literature relating to Continued Professional Education. Managers need to be supported by the organization in providing the time and resources necessary for self-education of employees. As many participants indicated that self-paced, independent learning was much more valuable than formal training courses, it would be helpful if organizational policy recognized the value of informal as well as more formal on-the-job learning.

## 5.6 Limitations

Because I was unable to identify a single specific resource which would enable me to contact or identify information about this population, a number of different recruitment strategies were used, including posting invitations in listservs, linkedIn groups, and other discussion forums, as well as the use of a snow-ball technique starting with the researcher's own personal contacts. There was a relatively large representation from some sources, especially the ACM SIGCSE and SIGITE (Computer Science Education and Information Technology Education Special Interest Groups of the Association for Computing Machinery), while other sources resulted in few or no responses to the survey. I suspect that this distorted the sample in some areas, especially in the number of participants who are current faculty members in higher education. Therefore, the findings are not meant to be generalized to a larger population. Rather, they represent the variety of backgrounds and attitudes among this group which raise questions to be explored further.

The findings of this study are based on participants' own perceptions of their learning and the importance of various forms of learning. Participants' memories may not always be exact, especially in the case of very experienced professionals who may have graduated several decades

ago. Furthermore, participants may not necessarily be able to identify the optimum way to learn or acquire particular skills, knowledge, or attitudes. However, understanding the experiences of these apparently successful professionals may well lead to valuable insights into ways to improve formal education as well as more structured non-formal educational experiences for novice professionals.

A few specific issues were identified in the Phase 2 survey instrument. As mentioned in the Findings section, the questions relating to degrees held were clearly not interpreted as intended by some participants. This was evidenced by the fact that a number mentioned only a single graduate degree, with no earlier degrees reported. Follow-up interviews with two participants in this category revealed that they had, indeed, left out multiple additional degrees. Because of this issue, I was not able to analyze data at the level of granularity I had initially hoped for. This may have allowed me to answer some of the open questions described in the discussion section.

Another question that was: "Over the span of your professional career, which of the following roles have you played in addition to the ones you hold now?" Quite a few participants also included roles they currently played, but some did not include the full set currently played. Since I am not certain how it was interpreted, I decided to leave this item out of the findings reported. Finally, one required item had an "other" field but no checkbox next to the "other" field. Therefore, even if something was entered into the "other" field, at least one additional closed-ended option had to be chosen, even if participants did not agree with any of the available choices. Some participants might have been forced to choose an answer they did not completely agree with, or drop out.

Finally, I did not receive the degree of respond in the third phase of the study that I had hoped for. Although over 40 volunteers were contacted for follow-up interviews in Phase 3, only 4 completed the follow-up interview. Although these responses added interesting aspects to the data, they were not sufficient to address some of the open topics. I had especially hoped to learn more about recent graduates, but although three participants were currently enrolled in PhD programs, none of those who responded could discuss recent experiences in a Bachelor's or even Master's program in a Computing or Instructional Design related program. Because of this lack, data from Phase 3 was merged with Phase 1 data, providing interview data from 13 individuals, in addition to open-ended responses from 74 of the survey participants on select topics.

## 5.7 Areas for Future Research

A planned future study of computing professionals across a range of industries will use a refined version of the questionnaire used in this study (correcting issues found during analysis, as well as modifying questions aimed at educational software or instructional technology to more general terms relating to the domain participants work in). Another study may look at the types of non-formal educational experiences sought by computing professionals, and the approaches they use in self-learning. Eventually, the researcher hopes to replicate this type of study in other design fields, including Instructional Design.

# 6 Works Cited

ABET Computing Accreditation Commission. (2010). Criteria for accrediting Computing programs: Effective for evaluations during the 2011-2012 accreditation cycle. Retrieved from http://www.abet.org/Linked%20Documents-UPDATE/Program%20Docs/abet-cac-criteria-2011-2012.pdf

Abran, A., Bourque, P., Dupuis, R., & Moore, J. W. (2001). *Guide to the Software Engineering Body of Knowledge*. Piscataway, NJ, USA: IEEE Press.

ACM and IEEE Computer Society. (2008). Computer Science curriculum 2008: An interim revision of CS 2001.

Andriole, S. J., & Roberts, E. (2008). Technology Curriculum for the Early 21st Century. *Communications of the ACM, 51*(7), 27-30.

Atlee, J. M. J., LeBlanc, R. J., Lethbridge, T. C., Sobel, A., & Thompson, J. B. (2006). Reflections on Software Engineering 2004, the ACM/IEEE-CS Guidelines for Undergraduate Programs in Software Engineering *Software Engineering Education in the Modern Age* (pp. 11-27). Berlin, Germany: Springer.

Boling, E., & Smith, K. M. (2007). Artifacts as tools in the design process. In M. D. M. J. Michael Spector, Jeroen van Merrienboer, Marcy P. Driscoll (Ed.), *Handbook of Research on Educational Communications and Technology* (3 ed.). Mahwah, NJ: Routledge.

Brooks, F. P. (2010). *The design of design: Essays from a computer scientist*. Upper Saddle River, NJ: Addison-Wesley.

Cennamo, K., Douglas, S., Vernon, M., Brandt, C., Scott, B., Reimer, Y., & McGrath, M. (2011). *Promoting creativitiy in the computer science design studio.* Paper presented at the SIGCSE '11, Dallas, Texas.

Cervero, R. M. (2001). Continuing professional education in transition, 1981-2000. *International Journal of Lifelong Education, 20*(1/2), 16-30.

Cooper, S., & Cunningham, S. (2010). Teaching Computer Science in Context. *Inroads, 1*(1), 5-8.

Cox, S., & Osguthorpe, R. (2003). How do Instructional Design professionals spend their time? *Techtrends, 47*(3), 45-47, 29.

Coyle, E. J., Jamieson, L. H., & Oakes, W. C. (2005). EPICS: Engineering projects in community service. *International Journal of Enginering Educaiton, 21*(1), 1-12.

Creswell, J. W., & Clark, V. L. P. (2007). *Mixed Methods Reseach*. Thousand Oaks, CA: Sage.

Cropley, A. J. (1989). Lifelong education: Interaction with adult education *Lifelong education for adults: An international handbook* (pp. 9-12). Oxford: Pergamon.

Cross, N. (2001). Designerly ways of knowing: design discipline vs. design science. *Design Issues, 77*(3), 49-55.

Cross, N. (2004). Expertise in design: An overview. *Design Studies, 25*(5), 427-441.

Daley, B. J. (2000). Learning in professional practice. *New directions for adult and continued learning, 86*, 33-42.

Denning, P. J. (2001). When IT becomes a profession *The invisible future: the seamless integration of technology into everyday life book contents* (pp. 295-325). New York, NY: McGraw Hill.

Denning, P. J. (2008). The profession of IT: Voices of computing. *Communications of the ACM, 51*(8), 19-21.

Denning, P. J., Athale, R., Dabbagh, N., Menasce, D., Offutt, J., Pullen, M., . . . Sadhu, R. (2001). *Designing an IT college.* Paper presented at the IFIP TC3 Seventh IFIP World Conference on Networking the Learner: Comptuers in Education.

Dewar, R., & Astrachan, O. (2009). Point/counterpoint: CS education in the U.S.: Heading in the wrong direction? *Communications of the ACM, 52*(7), 41-45.

Dewar, R., & Schonberg, E. (2008). Computer Science education: Where are the software engineers of tomorrow? *CrossTalk, 21*(1), 28-30.

Driscoll, M. P. (2000). Motivation and Self-Regulation in Learning *Psychology of Learnign for Instruction*. Needham Heights, MA: Allyn and Bacon.

Exter, M. E., & Turnage, N. H. (2011). *Exploring experienced professionals' reflections on computing education.* . Manuscript submitted for publication.

Fisher, J., Alvarez, J., & Taylor, R. (1978). *A survey of how practicing programmers keep up-to-date first results including their implications for computer science education.* Paper presented at the Proceedings of the ninth SIGCSE technical symposium on Computer science education, Pittsburgh, Pennslyvannia.

Ghassan Alkadi, T. B., Robert Schroeder. (2010). The someimtes harsh reality of real world Comptuer Science projects. *Inroads, 1*(4), 59-62.

Gibbons, A. S. (2000). *The practice of instructional technology*. Paper presented at the Annual International Conferece of the Association for Education Communications and Technology, Denver, CO.

Green, S. B., & Salkind, N. J. (2008). *Using SPSS for Windows and Macintosh: Analyzing and understanding data* (Fifth ed.). Upper Saddle River, NJ: Pearson Education, Inc.

Groth, D. P., & Hottell, M. P. (2006). *Designing and developing an Informatics capstone course.* Paper presented at the 19th Conference on Software Engineering Educaiton & Training.

Gudzia, M. (2010). Does contextualized computing education help? *Inroads, 1*(4), 4-6.

Hanna, Y., Yap, V., Fong, K. W., Fletcher, J., & Bancroft, C. Study of Job Skills-Set Required of IS Graduates for Work in Instructional Design. Retrieved from http://vinceyap.com/uploads/6635_FinalResearchReport.pdf

Hauer, A., & Daniels, M. (2008). *A learning theory perspective on running open ended group projects (OEGPs).* Paper presented at the Prcedings of the Tenth Australasian Computing Education Conference (ACE 2008), Wollongoing, Austrailia.

Houle, C. O. (1980). *Continued Learning in the Professions*. San Fancisco, CA: Jossey-Bass.

IEEE Conferences and Meetings. (2008) Retrieved November 21, 2008, from http://www.ieee.org/web/conferences/home/index.html

Kenny, R. F., Zhang, Z., Schwier, R. A., & Campbell, K. (2005). A review of what Instrucitonal Designers do: Questions answered and questions not asked. *Canadian Journal Of Learning And Technology / La Revue Canadienne De L'Apprentissage Et De La Technologie, 31*(1). Retrieved from http://www.cjlt.ca/index.php/cjlt/article/view/147/140

Kirk, R. (1999). *Statistics: An introduction* (Fourth ed.). Orlando, FL: Holt, Rinehart and Winston.

Knox, A. B. (2000). The continuum of professional education and practice. *New Directions for Adult and Continuing Education, 86*, 13-22.

Knox, A. B. (2006). The continuum of professional education and practice. *New Directions for Adult and Continuing Education, 86*, 13-22.

Korkmaz, N. (2011). *How is development of design judgement addressed in Instructional Design educaiton?* Doctoral dissertation, Indiana University, Bloomington, Indiaan.

Kuchinke, K. P. (1997). Employee Expertises The Status of the Theory and the Literature. *Performance Improvement Quarterly, 10*(4), 72-86.

Kumar, D. (2010). Reflections: Languages, Wars and False Dichotomies. *Inroads, 1*(3), 10-11.

LACS. (2007). A 2007 model curriculum for a liberal arts degree in computer science. *Journal on Educational Resources in Computing (JERIC), 7*(2), 1-34.

Larson, M. B. (2005). Suvey of the Alignment of Prepration and Practice. *Techtrends, 49*(6), 22-32,68.

Larson, M. B., & Lockee, B. B. (2007). Preparing Instructional Designers for different career environments: A case study. *Educational Technology Research and Development, 57*(1), 1-24.

Lawson, B. (1997). Route maps of the design process *How Designers Think: The Design Process Demystified* (Third ed.). Boston, MA: Architectural Press.
Lawson, B. (2004). Schemata, gambits, and precedent: Some factors in design expertise. *Design Studies, 25*(5), 443-457.
Lengrand, P. (1989). Lifelong education: growth of the concept *Lifelong education for adults: An international handbook* (pp. 5-9). Oxford: Pergamon.
Lethbridge, T. C. (2000). What knowledge is important to a software professional. *Computer, 33*(5), 44-50.
Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic Inquiry*. Newbury Park, CA: Sage.
Livingstone, D. (2001). Adults' informal learning: definitions, findings, gaps and future research. *T-Space at The University of Toronto Libraries: Centre for the Study of Education and Work (CSEW)*. Retrieved from http://www.oise.utoronto.ca/depts/sese/csew/nall/res/21adultsifnormallearning.pdf
Job-search success for the class of 2011. (2010). *Job Choice 2011: Science, engineering, technology students*.
Mott, V. W. (2000). The development of the professional expertise in the workplace. *New Directions for Adult and Continuing Education, 86*, 23-31.
Nelson, H. G., & Stolterman, E. (2002). *The Design Way: Intentional Change in an Unpredictable World*. Englewood Cliffs, NJ: Educational Technology Publications.
Nurkkala, T., & Brandle, S. (2011, March 9-12). *Software studio: Teaching professional software engineering.* Paper presented at the SIGSCE '11, Dallas, Texas.
O'Donnell, K. (2006). National household education surveys program of 2005: Adult education participation in 2004-2005. *National Center for Education Statistics*. Retrieved from http://nces.ed.gov/pubsearch/pubsinfo.asp?pubid=2006077
Petroski, H. (1992). Engineering as hypothesis *To Engineer is Human: The role of Failure in Succesful Design* (pp. 40-52). New York, NY: Vintage Books.
Popyack, J. L. (2010). Are you ready for teh renaissance? *Inroads, 1*(1), 31-32.
Radcliffe, D. J., & Colletta, N. J. (1989). Nonformal education *Lifelong education for adults: An international handbook* (pp. 60-63). Oxford: Pergamon.
Richey, R. C., Fields, D. C., & Foxon, M. (2001). Instructional design competencies: The standards. . Syracuse University: Syracuse, NY.
Rowe, P. G. (1991). Procedural aspects of design thinking *Design Thinking* (pp. 39-113). Cambridge, MA.
Schon, D. (1987). Educating the reflective practitioner, as presented at the 1987 Meeting of the American Educational Research Association. Retrieved November 18, 2008, from http://educ.queensu.ca/~russellt/howteach/schon87.htm
Shulman, L. S. (2005). Signature pedagogies in the professions. *Daedalus, 134*(3), 52-59.
Smith, K. M. (2008). *Meanings of "design" in instrucitonal technology: A conceptual analysis based on the field's foundational literature.* Unpublished doctoral dissertation, Indiana University, Bloomington, Indiana.
Spector, J. M., Klien, James D., Reiser, Robert A., Sims, Roderick C., Grabowski, Barbara L., de la Teja, Ileane. (2006). *Competencies and Standards for Instructional Design and Educational Technology*. Paper presented at the ITFORUM.
Srinivasan, L. (1989). Nonformal education: Instruction *Lifelong education for adults: An international handbook* (pp. 212-214). Oxford: Pergamon.

. Standards for the accreditation of school media specailist and educational technology specialist programs. (2005). In R. S. Earle (Ed.). Bloomington, IN: Association for Educational Communications and Technology.

Surakka, S. (2004). *Analysis of job advertisements: What technical skills do software developers need?* Paper presented at the Fourth Finnish/Baltic Sea Conference on Computer Science Education, Koli, Finland.

Teddlie, C., & Tashakkori, A. (2009). *Foundations of Mixed Methods Research: Integrtign Quantitative and Qualitative approaches in the Social and Behavioral Sciences*. Thousand Oaks, CA: Sage.

The Joint Task Force on Computing Curricula. (2004). Software Engineering 2004: Curriculum guidelines for undergraduate degree programs in software engineering [electronic version]. Retrieved from http://www.acm.org/education/curricula.html

Tough, A. M. (1989). Self-directed learning: Concepts and practice *Lifelong education for adults: An international handbook* (pp. 256-261). Oxford: Pergamon.

Vincenti, W. G. (1990). Chapter 7: The anatomy of engineering design knowledge *What engineers know and how they know it: Analytical studies from aeronotical histories* (1 ed., pp. 200-240). Baltimore, MA.: Johns Hopkins Univesrity Press.

Walker, H. (2010). Eight principles of an undergraduate program. *Inroads, 1*(1), 18-20.

Wolz, U., Cassel, L., Way, T., & Pearson, K. (2011, March 9-11). *Cooperative expertise for multidisciplinary computing.* Paper presented at the SIGCSE '11, Dallas, Texas.

# 7 Appendix A: Phase 1 Semi-structured interview protocol

*INITIALS:*

*DATE:*

*Thank you for your willingness to participate in this study.*
*Have you had a chance to look over the study information sheet that I sent you? Do you have any questions?*

*I would like to remind you once again that we can end the interview at any time you feel uncomfortable or wish to stop. Also, if you say something inadvertently that you do not want to be included in the transcript – such as proprietary information – just let me know during or after the interview and I will omit that segment from the transcript.*

*Do you mind if I type while we talk?*

*Do you mind if I record our conversation? The recording will only be used to help me make a complete and accurate transcription.*

**PRESS RECORD**

1 Current Role

1.1 Where do you currently work?

1.2 What is your official title at <company>?

1.3 Is your current job related to the design or development of software? (if not, ask about previous roles).

1.4 Does your current job relate to the development of educational or instructional software? Can you please describe what type? (if not instructional or educational or related (e.g. support systems for instructional design efforts), ask about previous roles)

1.5 What was your work history prior to starting this role?

1.6 How did you become involved in *educational* software design?

1.7 What does your current role entail? Are there any aspects of this job which you feel are unique to working in a [instructional design/educational software design] related company?

1.8 [If this has not come out sufficiently in the previous questions:] What types of skills and knowledge are most important for the role(s) you currently fill in your job?

2 Formal Education

2.1 What is your formal educational background?

2.2 What are the most important things you learned within [CS/SE/other "technical"] courses as applies to your current role? [If applicable]

2.3 What are the most important things you learned in within [education/IST/Ed Tech] courses as applies to your current role? [If applicable]

2.4 What did you learn in other courses [e.g. GenEds or courses taken as part of an unrelated major] that turned out to be applicable to your current role?

2.5 Are there things you do within your current job which you felt your formal educational background did not adequately prepare you for?

3 Non-Formal Education

3.1 Where do you go to learn more about technical skills/knowledge (e.g. programming languages, technologies, software design concepts) you need on the job? [If multiple sources, e.g. training courses, peers, books, internet, internal resources, experimentation:] how do you decide when to go to each source?

3.2 Where do you go to learn more about non-technical skills/knowledge you need on the job? [If multiple sources, e.g. training courses, peers, books, internet, internal resources, experimentation:] how do you decide when to go to each source?

4 Recommendations for Educational Programs

4.1 Ask first: what type of degree do you think is most useful for people working in your role, working on educational software?

4.2 4.2 Follow up: Do you think people benefit from industry-specific courses or just focus on technical?

4.3 If you could design an ideal undergraduate or Master's program to prepare someone for the role you currently fill, what would it look like?

# 8 Appendix B: Phase 2 Survey instrument

## The Educational Background of Designers / Developers of Software for K-12 and Higher Education

Exit this survey

### 1. The Educational Experiences of Designers/Developers of Educational Software

4%

Thank you for agreeing to participate in this survey. It is a part of a larger research study, which aims to provide a greater understanding of the formal and non-formal educational experiences and needs of software designers working on educational software. Data collected in this study will be used in a dissertation study and may also be included in research publications or conference presentations.

Depending on the nature of your experiences, the survey may take between 20 and 40 minutes to complete. Your responses will remain anonymous, although you may optionally provide an email address if you are willing to participate in a follow-up interview or to enter the drawing for a $50 Amazon gift certificate. If you have any questions about this survey or about the study, please contact mexter@indiana.edu.

Do not use the "back" and "forward" buttons on your browser while taking the survey. To continue where you left off at a later time, simply use the link again on the same computer and browser. If you do not have cookies enabled or you clear the cookies, you will not be able to continue at a later time. Your personal data will not be stored on the server.

***1. Have you ever been involved in the design/development of software used for educational or instructional purposes in elementary, secondary, or higher education?**

- yes
- no

Next

## 2. Current Employment

7%

Please answer the following questions in relation to your current or most recent position. If you changed jobs recently, please respond in relation to the most recent job in which you worked on some aspect of software design/development.

**1. What is your formal title at your current job?**

***2. Which of the following role(s) do you fill in your current position? (choose all that apply)**

- [ ] Software architecture
- [ ] Business requirements gathering/generation
- [ ] Technical requirements gathering/generation
- [ ] High-level design
- [ ] Low-level design
- [ ] Programming
- [ ] Database design
- [ ] Web developer/Web designer
- [ ] User Experience Design
- [ ] Quality Assurance
- [ ] Instructional Design
- [ ] Supervisory
- [ ] Other (please specify)

*** 3. What types of software are developed by your organization? (choose all that apply)**

- [ ] Learning Management System (LMS) or Course Management System (CMS)
- [ ] Components that reside in a LMS/CMS
- [ ] Drill and practice application
- [ ] Self-paced instruction (stand-alone application)
- [ ] Self-paced instruction (web-based)
- [ ] Educational interactive simulation
- [ ] Educational game
- [ ] Software not used for educational/instructional purposes
- [ ] Other (please specify)

*** 4. What is the size of the company/organization you currently work in?**

- ○ Single-person business or independent contractor
- ○ Less than 5 employees
- ○ 6 - 20 employees
- ○ 21-99 employees
- ○ 100 – 499 employees
- ○ 500 or more employees
- ○ Don't Know

✱ **5. What type of organization do you currently work in?**
**Check the most accurate response:**

- Educational software development company which focuses on software used in higher education
- Educational software development company which focuses on software used in k-12 school settings education
- Educational software development company which creates software for use in multiple contexts
- Department within a larger company which produces some educational or instructional software
- Department in a university which develops software primarily intended for use in a distance education program
- Department in a university which develops other types of software for use at that university or across universities
- Department, research center, or other group in a university which develops educational software for use in educational settings outside of the university
- Other (please specify)

Prev Next

## 3. Working in Educational Software Design

11%

**1. What caused you to begin working in educational software design/development? (choose all that apply)**

- [ ] Lots of experience in the educational software industry
- [ ] Background in instructional design
- [ ] Ongoing interest in education/educational support
- [ ] Makes me feel needed/valued
- [ ] Good working environment
- [ ] Interesting design problems
- [ ] Contacts in the industry
- [ ] Just a job I found

Other (please specify)

***2. What skills and knowledge do you believe are unique to developing software for education (as opposed to software for other industries/purposes)? (choose all that apply)**

- [ ] Not unique – this job requires the same skills that I would need to gather requirements and develop software for any specialized group of users
- [ ] Not Unique – I know this content well because I have been working with it for an extended period of time, but I would have learned about any domain after having worked in it for the same amount of time
- [ ] Educational theory
- [ ] Instructional theory
- [ ] Instructional design experience
- [ ] The ability to educate clients and co-workers on topics that are not well known or understood in this field
- [ ] Other (please specify)

Prev Next

## 4. Formal Education

[redacted] 15%

Please answer the following for your most recent degree. If you are currently working on a degree, choose the degree you are working on.

***1. Most recent degree obtained or are currently working on.**

- High school or equivalent
- Some college courses(individual courses taken without intent to pursue a degree program)
- Some college but no degree (within a degree program)
- Associates
- Bachelors
- Masters
- Doctoral
- Other (please specify)

**2. Degree in (If multiple majors, please check all)**

- [ ] Computing (CS, SE, etc.)
- [ ] Computing & Business (CIS, MIS)
- [ ] Software Usability (e.g. Human Computer Interface Design, User Experience Design)
- [ ] Other technical field (CE, EE, IT, Telecommunications)
- [ ] Education related (Early Childhood, Elementary, Middle, or Highschool teaching degree)
- [ ] Educational technology/Instructional Technology
- [ ] Instructional Design
- [ ] Physical Science (e.g. Physics, Biology, Chemistry, Astronomy, etc.)
- [ ] Social Science (e.g. Psychology, Sociology, Anthropology, etc.)
- [ ] Art (e.g. Fine Arts, Graphic Arts)
- [ ] Teaching Certificate along with a degree in a specific topic area
- [ ] Other (please specify)

**3. What was your minor/specialty area? (If you had multiple minors or specialty areas, please check all)**

- ☐ Computing (CS, SE, etc.)
- ☐ Computing & Business (CIS, MIS)
- ☐ Software Usability (e.g. Human Computer Interface Design, User Experience Design)
- ☐ Other technical field (CE, EE, IT, Telecommunications)
- ☐ Education related (Early Childhood, Elementary, Middle, or Highschool teaching degree)
- ☐ Educational technology/Instructional Technology
- ☐ Instructional Design
- ☐ Physical Science (e.g. Physics, Biology, Chemistry, Astronomy, etc.)
- ☐ Social Science (e.g. Psychology, Sociology, Anthropology, etc.)
- ☐ Art (e.g. Fine Arts, Graphic Arts)
- ☐ N/A
- ☐ Other (please specify)

**4. Year Graduated/Will Graduate**

**5. Name of school (optional)**

**6. While pursuing this degree, I worked in a(n):**
**(check all that apply)**

- [ ] Internship/Co-op
- [ ] Graduate Assistantship which involved software design/development
- [ ] Graduate Assistantship which involved instructional design
- [ ] Graduate Assistantship which involved teaching
- [ ] Full-time or Part-time job in software design/development
- [ ] Full-time or Part-time job in instructional design
- [ ] Full-time or Part-time job involving teaching
- [ ] Other education-related job
- [ ] Unrelated job
- [ ] Did not work

***7. Did you pursue or complete any other degrees?**

- ○ Yes
- ○ No

Prev Next

## 5. Formal Educational Experiences

37%

✱1. Did you take any courses related to software design or development as part of your formal education?

- Yes
- No

Prev Next

## 6. Formal Educational Experiences (Continued)

41%

✱1. How important were each of the following *software design-related courses* to you as a professional software designer/developer?

| | Unimportant | Moderately Important | Very Important |
|---|---|---|---|
| Basic theoretical courses (e.g. "Data-structures & Algorithms", "Database Design") | | | |
| Object oriented design | | | |
| Courses on specific programming languages | | | |
| Interface design | | | |
| Networking related courses | | | |
| Operating Systems | | | |
| Courses dealing with hardware, firmware, assembly language | | | |

✱2. To what degree did *your coursework* prepare you for the following areas?

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Working directly with users | ○ | ○ | ○ |
| Working with specific software development environments/ IDEs | ○ | ○ | ○ |
| Working with change control software (software that keeps track of multiple versions of code) | ○ | ○ | ○ |
| Maintaining code over time | ○ | ○ | ○ |
| Testing practices | ○ | ○ | ○ |

Prev Next

## 7. Formal Educational Experiences (Continued)

[redacted] 44%

✱1. Did you take any courses related to education or instructional design as part of your formal education?

○ Yes

○ No

Prev Next

## 8. Formal Educational Experiences (Continued)

48%

***1. How important were each of the following *education related courses* to you as a professional software designer/developer?**

| | Unimportant | Moderately Important | Very Important | N/A |
|---|---|---|---|---|
| Instructional Design: Theoretical courses | ○ | ○ | ○ | ○ |
| Instructional Design: Practical Courses | ○ | ○ | ○ | ○ |
| Educational Theory courses | ○ | ○ | ○ | ○ |
| Other education related courses | ○ | ○ | ○ | ○ |

***2. To what degree did *your instructional design-related coursework* prepare you for the following areas?**

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Designing instructional materials (other than software) | ○ | ○ | ○ |
| Designing instructional software | ○ | ○ | ○ |
| Developing instructional software | ○ | ○ | ○ |
| Testing instructional software | ○ | ○ | ○ |

Prev Next

## 9. Formal Educational Experiences (Continued)

52%

***1. How important were each of the following educational experiences to you as a professional software designer/developer?**

| | Unimportant | Moderately Important | Very Important | N/A |
|---|---|---|---|---|
| Mathematics | | | | |
| Natural Sciences (e.g. Physics, Chemistry, Biology, Astronomy, Earth Sciences, etc.) | | | | |
| Social Sciences (e.g. Psychology, Sociology, Anthropology, Political Science, etc.) | | | | |
| Communications/English (or your primary language if English is not your primary language) | | | | |
| Foreign Language(s) | | | | |
| Business | | | | |
| Internship/Coop Experiences | | | | |
| Graduate Assistantship | | | | |

✱**2. To what degree were you prepared for the following by *the combination of all of your coursework*?**

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Experience with the full life-cycle of a project | ○ | ○ | ○ |
| Business aspects of the industry I work in | ○ | ○ | ○ |
| Legal aspects of the work I do | ○ | ○ | ○ |
| Technical jargon used on the job | ○ | ○ | ○ |
| Domain/industry specific jargon used on the job | ○ | ○ | ○ |

Prev Next

## 10. Professional Preparation

56%

The following 3 pages each include 2 short sets of questions.

On each page, you will first be asked to indicate how important specific **knowledge or skills** have been ***over the course of your professional life***. You will then be asked to what ***degree these were stressed in your formal educational (university) experiences***.

Your responses will help us to understand the degree to which people with various backgrounds feel prepared for their current careers.

Prev Next

## 11. Professional Preparation

59%

***1. How important have the following skills been to you *over the course of your professional life*?**

| | Unimportant | Moderately Important | Very Important |
|---|---|---|---|
| Ability to work well in teams | | | |
| Ability to perform many different roles | | | |
| Strong skills in one particular role/specialty area (e.g. requirements gathering, user experience design, visual design, database design, low-level coding, etc.) | | | |
| Ability to communicate with people in different specialty areas (e.g. visual designers, programmers, instructional designers, usability experts, subject matter experts, etc.) | | | |

***2. To what degree were each of the following skills stressed *across all college/university degree program(s)* you attended?**

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Ability to work well in teams | ○ | ○ | ○ |
| Ability to perform many different roles | ○ | ○ | ○ |
| Strong skills in one particular role/specialty area (e.g. requirements gathering, user experience design, visual design, database design, low-level coding, etc.) | ○ | ○ | ○ |
| Ability to communicate with people in different specialty areas (e.g. visual designers, programmers, instructional designers, usability experts, subject matter experts, etc.) | ○ | ○ | ○ |

Prev Next

## 12. Professional Preparation (cont'd)

63%

***1. How important have the following been to you *over the course of your professional life*?**

| | Unimportant | Moderately Important | Very Important |
|---|---|---|---|
| Critical/analytical thinking | ○ | ○ | ○ |
| Problem solving | ○ | ○ | ○ |
| Ability to teach myself what I need to know | ○ | ○ | ○ |

***2. To what degree were each of the following stressed *across all college/university degree program(s)* you attended?**

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Critical/analytical thinking | ○ | ○ | ○ |
| Problem solving | ○ | ○ | ○ |
| Ability to teach myself what I need to know | ○ | ○ | ○ |

Prev Next

## 13. Professional Preparation (cont'd)

67%

***1. How important have the following types of knowledge been to you *over the course of your professional life*?**

| | Unimportant | Moderately Important | Very Important |
|---|---|---|---|
| Knowledge of specific programming languages | | | |
| Knowledge of web languages/technologies | | | |
| Interface design principles | | | |
| User experience design principles | | | |

***2. To what degree were each of the following types of knowledge stressed *across all college/university degree program(s)* you attended?**

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Knowledge of specific programming languages | | | |
| Knowledge of web languages/technologies | | | |
| Interface design principles | | | |
| User experience design principles | | | |

Prev Next

## 14. Self-learning Approaches

70%

The following 3 pages each include 2 short sets of questions.

You first be asked to indicate how important a series of **learning strategies** has been ***over the course of your professional life***. You will then be asked to what degree these were ***stressed in your formal educational (university) experiences***.

The purpose of these questions is to understand self-learning practices of working professionals in your field.

Prev Next

## 15. Self-learning Approaches

74%

✱**1. How important have each of the following learning strategies been to you *over the course of your professional life*?**

| | Unimportant | Moderately Important | Very Important |
|---|---|---|---|
| Using your peers as a resource | | | |
| Using examples you find in books or online to help you learn a new algorithm or way of approaching a problem | | | |
| Learning how to resolve a problem by searching for error messages or other key-words online | | | |

✱**2. To what degree were each of the following learning strategies *stressed in the college/university degree program(s)* you attended?**

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Using your peers as a resource | ○ | ○ | ○ |
| Using examples you find in books or online to help you learn a new algorithm or way of approaching a problem | ○ | ○ | ○ |
| Learning how to resolve a problem by searching for error messages or other key-words online | ○ | ○ | ○ |

Prev Next

## 16. Self-learning Approaches (cont'd)

78%

***1. How important have each of the following strategies been to you *over the course of your professional life*?**

| | Unimportant | Moderately Important | Very Important |
|---|---|---|---|
| Creating your own sample code/prototype system before incorporating changes into the product | | | |
| Working on a project in small increments and testing each piece before continuing | | | |
| Reverse engineering existing products | | | |
| Keeping detailed notes about your own design decisions | | | |
| Writing detailed comments in your code | | | |

✱**2. To what degree were each of the following strategies *stressed in the college/university degree program(s)* you attended?**

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Creating your own sample code/prototype system before incorporating changes into the product | ○ | ○ | ○ |
| Working on a project in small increments and testing each piece before continuing | ○ | ○ | ○ |
| Reverse engineering existing products | ○ | ○ | ○ |
| Keeping detailed notes about your own design decisions | ○ | ○ | ○ |
| Writing detailed comments in your code | ○ | ○ | ○ |

Prev Next

## 17. Self-learning Approaches (cont'd)

81%

***1. How important have each of the following strategies been to you *over the course of your professional life*?**

| | Unimportant | Moderately Important | Very Important |
|---|---|---|---|
| Studying other people's code/design documents to learn better coding/documentation style | | | |
| Applying things you learned in one project to new, unrelated projects | | | |
| Recognize when new technologies are similar to ones you have seen before | | | |
| Looking for a high-level view of a smaller problem (such as learning about the system-wide architecture when you are working on one component) | | | |
| Know when to reuse code and when not to | | | |
| Put aside pre-existing paradigms if necessary | | | |

***2. To what degree were each of the following strategies *stressed in the college/university degree program(s)* you attended?**

| | Not at all well | Somewhat | A great deal |
|---|---|---|---|
| Studying other people's code/design documents to learn better coding/documentation style | ○ | ○ | ○ |
| Applying things you learned in one project to new, unrelated projects | ○ | ○ | ○ |
| Recognize when new technologies are similar to ones you have seen before | ○ | ○ | ○ |
| Looking for a high-level view of a smaller problem (such as learning about the system-wide architecture when you are working on one component) | ○ | ○ | ○ |
| Know when to reuse code and when not to | ○ | ○ | ○ |
| Put aside pre-existing paradigms if necessary | ○ | ○ | ○ |

Prev Next

## 18. Suggestions for the development of an Ideal Educational Program

85%

**✱1. If you were to design an ideal bachelor's degree program to prepare someone for your current position, what type of degree would it be?**

- Computer Science
- Software Engineering
- Information Systems
- Human Computer Interaction Design
- Instructional Systems Technology/Instructional Design
- Other Education-related
- Hybrid program or double major including CS (or similar) and Education (or similar)
- Doesn't matter/any path would work
- A degree has little value for working in this field.
- Other (please specify)

**✱2. Is it important to have a domain-specific specialization or focus within this type of degree program (such as "education" or even a more specific area such as "language education" or "science education")? *Pick the answer that best reflects your feelings.***

- Yes, it is important to focus on the domain one plans to work in
- Yes, students need to work within one domain to get practice in a realistic project, but the specific domain is not important
- No, not important
- Other (please specify)

✱ **3. Is it important to have formal training in programming languages and other technical aspects of software design/development? *Pick the answer that best reflects your feelings.***

- ◯ Yes, it is important
- ◯ Yes, important, but not as important as a solid background in education or instruction
- ◯ Yes, it is important, but not as important as the ability to think critically
- ◯ No, not important – people can learn that on their own
- ◯ Other (please specify)

**4. What traits would the ideal bachelor's degree program have?**
**(Please provide any level of detail you feel is applicable, from recommended courses to specific topics or activities, and/or a general statement of philosophy about what the program should be like.)**

Prev Next

## 19. Demographics

89%

You are almost done! Just a few more questions about you and your work experience.

**1. Gender**

- Male
- Female

***2. In what country do you currently work?**

***3. Did you complete any of your formal education in a country other than the one in which you currently work? If you did, please list the country or countries in which you pursued post-secondary education.**

- No
- Yes (please specify countries in whcih you attended university/college)

Prev Next

## 20. Demographics (Employment History)

93%

For the following questions, include years worked in internships/coops, volunteer positions, and full- or part-time employment, but NOT including class projects.

***1. How many years have you worked in software design/development?**

- Less than 1 year
- 1-4 years
- 5-10 years
- 11-15 years
- 16-20 years
- 21+ years

***2. How many years have you worked on software design/development of software used primarily for educational purposes?**

- Less than 1 year
- 1-4 years
- 5-10 years
- 11-15 years
- 16-20 years
- 21+ years

***3. How many years of instructional design experience have you had?**

- None
- Less than 1 year
- 1-4 years
- 5-10 years
- 11-15 years
- 16-20 years
- 21+ years

***4. How many years of teaching experience do you have?**

- None
- Less than 1 year
- 1-4 years
- 5-10 years
- 11-15 years
- 16-20 years
- 21+ years

***5. Over the span of your professional career, which of the following roles have you played in addition to the ones you hold now?**
**Check all that apply:**

- [ ] Software architecture
- [ ] Business requirements gathering/generation
- [ ] Technical requirements gathering/generation
- [ ] High-level design
- [ ] Low-level design
- [ ] Programming
- [ ] Database design
- [ ] Web developer/Web designer
- [ ] User Experience Design
- [ ] Quality Assurance
- [ ] Instructional Design
- [ ] Supervisory
- [ ] Other (please specify)

Prev Next

## 21. Follow-up

96%

**1. We will be conducting follow-up interviews with some survey participants. If you would be willing to participate in a follow-up interview, please provide am email address we can contact you at.**

***Your email address will only be used to contact you for a follow-up interview. It will not be shared with third parties, and no record of your email address will be kept after data collection for this study is complete.***

**Email address to contact for follow-up interview (optional):**

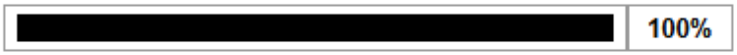
**2. To thank you for your participation in this survey, we would like to offer you the opportunity to win a $50 Amazon gift certificate. The chance of winning is approximately 1 in 50.**

***Your email address will only be used to contact you if your name is chosen. It will not be shared with third parties, and no record of your email address will be kept after data collection for this study is complete. If you did not provide your email address in Question 1, you will not be contacted for a follow up interview.***

**Email address to contact for chance to win Amazon Gift Certificate (optional):**

Prev Next

## 22. Thank You




**Thank you for participating in this research study. Please press the button at the bottom of this page.**

***If you have colleagues who might be interested in participating, please direct them to this survey. You may do so by forwarding the following text via email:***

---

*Subject:* Research Study on the Educational Background of Designers/Developers of Software for K-12 or Higher Education

Dear Educational Software Designer,

I am writing to invite you to participate in a survey. This survey is a part of a larger research study, which aims to provide a greater understanding of the educational experiences and needs of computing professionals who design or develop software used in K-12 or higher education. Data collected will be used as part of a dissertation study and may also be included in research publications or conference presentations.

Depending on the nature of your experiences, the survey may take between 20 and 40 minutes to complete.

As a thank-you for your time, you will be offered the opportunity to be entered into a pool to win a $50 Amazon gift certificate. The chance of winning is approximately 1/50. In order to be entered into the pool, you will be asked for your email address at the end of the survey. You may also optionally provide an email address if you are willing to participate in a follow-up interview.

Should you choose to enter your email address, it will not be used for any other purpose and your responses will remain anonymous.
To participate in the survey, please go to the following URL:

http://www.surveymonkey.com/s/LT8M383

If you have any questions about the study or procedures, please contact me at mexter@indiana.edu.

Thank you in advance,

Marisa Exter
Doctoral Candidate, Instructional Systems Technology
Indiana University, Bloomington, Indiana
mexter@indiana.edu

---

Prev Done

# 9 Appendix C: Phase 3 interview protocol: Sample of a personalized email

Thank you for participating in a survey regarding your educational background and work experiences on December 20, 2010. While taking the survey, you indicated that you may be willing to participate in a follow-up interview. I would like to offer you the option to respond either via email or via a phone interview, whichever you are more comfortable with. The questions I would like to address are listed below.

---

1. In the survey, you indicated that you plan to graduate with a Doctoral degree in 2015. What is your major area of concentration? You did not indicate any prior degrees. Have you completed other degrees? If so, could you please give the type of degree (Bachelors, Doctoral, etc), year of graduation, and major for each?

2. Does any of the coursework for the degree you are currently pursuing involve engaging in real-world or realistic projects? If so, could you briefly describe what these projects entailed (e.g. duration, type of project, group vs individual project, whether it was for a real client or a hypothetical problem)? Was this helpful in preparing you for your current or previous professional role(s)?

3. Have any courses or activities you have participated in as part of this degree program focused on one or more specific domains (e.g. "Educational Software", "Games", etc.)? Was this valuable in preparing you for your current or previous professional role(s)? If so, why?

4. In your survey response, you indicated that degree program(s) you attended did a good job at covering the following areas. For each, could you please briefly indicate how this skill or topic was covered or fostered within the courses you took?

a. Designing and developing instructional software
b. Testing practices
c. Business aspects of the industry I work in
d. Working in teams

5. Were there any experiences that you felt were really lacking in your own educational background?

6. Were there any areas relevant to your current professional position that you felt the program(s) you attended excelled at?

As you may or may not recall, the survey included an open-ended question regarding your suggestions for an "ideal bachelor's degree program to prepare someone for your current position".

You indicated you felt an ideal program would be a hybrid degree including CS (or similar) and Education (or similar). You indicated that it is important to focus on the domain one plans to work in (e.g. "education" or a more specific area such as "language education"), and that formal training in programming languages and other technical aspects of software design/development

is important, but not as important as the ability to think critically. You further recommended the following traits for the ideal bachelor's degree program: "Critical thinking Logic thinking Programming experience Gaming concepts Database management skills Artistic training (3D modeling) Various levels of mathematics capstone course"

1. The question asked you to address the program in terms of preparation for "someone in your current position". What type of roles do you believe this type of program might prepare them for?

2. You recommended artistic training, specifically 3D modeling. Is this something you would recommend in a degree program for educational software design generally, or is this a skill very specific to your current role?

3. You also recommended gaming concepts. Would you recommend this to people pursuing a career in educational software design generally, or only those who will focus on educational games specifically?

4. You recommended various levels of mathematics. What types of mathematics would you recommend specifically?

5. Individuals who participated in the survey indicated that the following areas are important in developing an ideal program. To what extent do you believe that each of these should be incorporated into the program? Do you have any additional suggestions on good ways to incorporate these competencies into a degree program?

- Foster creativity
- Foster critical thinking skills
- Foster the ability and interest in continuous on-the-job learning
- Develop artistic or visual design skills
- Gain experience with skills and tools used in real-world problems on the job
- Give lots of practical experience
- Provide a solid foundation in software engineering theory and practices
- Provide a solid foundation in software development/programming theory and practices
- Provide a solid foundation in instructional design theory and practices
- Provide a solid foundation in user interface design theory and practices

6. Is there anything else you would like to add on this topic?

---

Please review the attached Study Information Sheet for more information about the study and your rights and protections before responding to this email. If you would prefer to participate in a phone interview instead, I would be happy to schedule one with you. The phone interview would likely take about 30-40 minutes.

If you have any questions about the study or procedures, please contact me at mexter@indiana.edu.

Thank you in advance,

Marisa Exter
mexter@indiana.edu
Doctoral Candidate, Instructional Systems Technology
Indiana University, Bloomington, Indiana

# 10 Appendix D: Notes from external review of coding by experience colleague

**Review of Qualitative Findings (themes): 5/23/2011**

***Reviewer's Background***

This reviewer has worked both individually and with me on related research projects. She also has a work history in computing, degrees in both CIS and IST, and is currently teaching Computer Science. She is very familiar with CS standards, as she is currently participating in a re-design of a CS program.

***Review Process***

I had a skype meeting with the external reviewer on May 23, 2011. I sent her a printout of the coding hierarchy, as displayed in the NVivo software. This contained only the list of themes and sub-themes, not the actual transcript text. However, if she had a question about the meaning of any of the themes, I read relevant sections of text out to her.

She looked through the entire hierarchy and provided the following feedback:

**Comments on arrangement of themes and sub-themes**

1. Move ideal program->program traits->communication skills to a subtheme under program→program traits→traits to foster in ideal people"
    a. "Computing foundations" vs "Computing specialty areas": She agrees with the naming and content of these themes.
    b. Her response to "computing foundations": "[the courses listed under 'computing foundations'] are courses that would be in ANY CS program, the things that our field considers to be the core of what we do. They stay the same no matter what domain or industry you are practicing them in."
    c. Her response to "computing specialty areas": she feels this is appropriately named and that these do not belong in "computing foundations". AI, security, etc. are all topics which may be used more heavily in some domains than others. They are skills some employers might look for to fill a specific role, but would not assume that everyone has. For example, Lockheed martin, DOE, MacAfee, Norton – look for people with specialties in security. Lucent Technologies may look for people with firm foundation in hardware and R&D. Walmart would look for people who generally

"code well". In several cases in my study, people indicated the importance of "web security" for those working on e-learning or LMS systems.

2. She did not feel that "Domain-specific foundations" was a coherent category, and recommended I break this up between "Computing Specialty" and a new category, "Education Specialty Areas".
    a. Move to "Computing specialty areas"
        i. game design,
        ii. simulation (assuming that by this they meant a physics engine – *I will look again at the context of the places this was mentioned and try to determine what was meant. Generally there was not a lot given but I may be able to determine what was probably meant based on the type of software they work on*)
        iii. "Computation as a problem solving method": Reviewer explains: "There is a new field of computation science. This involves new cross-disciplinary programs , for instance a program that combines a major in chemistry or mathematics or physics + a computational degree. The purpose of these programs is to learn how to use Computer Science within your specific domain."
    b. "Educational specialty areas"
        i. Content areas
        ii. Domain-spec teaching
        iii. In-depth courses in application areas
3. The theme "Nonformal ed->Nonformal ed lead to choose higher ed" seems to be kind of an outlier in this area. Consider moving it somewhere else.
4. My question: how can I simmer down "other" categories, or sub-categories under nonformal ed types….
    a. Can group some together ---even if the text coded says something slightly different, but says it about the same category. Examples:
        i. **Help you get past a mental block** or **Help you to work through the problem** would include sub-themes where co-workers or other people help you push past where you are stuck in some way. Includes the current codes:
            1. Person helps find patterns you haven't seen before
            2. People – when a problem is difficult to understand
            3. Help get to the next level
        **ii.** **Experts** or **Looking for expertise:** For my purposes, it does not matter if they are inside or outside experts. They are people who really know the thing that you need to learn how to do. Includes the current codes:
            1. Expert consultants
            2. SME
            3. Outside consultants

4. Whoever has info

5. Other than above, nothing jumps out that doesn't make sense to her. All themes and sub-themes seem distinct and make sense.

**Additional feedback offered:**

1. She found it "Interesting" that 21 people felt a hybrid program would make sense. She would not have expected that a large number of people would say this. She would have expected that most people would recommend something similar to their own degree (either Computing or ID related, or something else).
2. Interesting that "mentorship" is only mentioned by one person
    a. Note: I explained that I want to be very cautious about making it for survey participants a lot of the responses were very brief and turse to a pretty broad open-ended question. So, I don't feel that I can say for sure that people WOULDN'T think it is important, if I asked them this directly. Reviewer agreed.
3. It made her feel good that what I see in Nonformal education area follows the same lines as what we saw in an earlier study, even though I allowed the themes to emerge and did not try to code it to the same structure as in the earlier study. She felt that although the findings are not exactly the same, they are very similar and the general message is the same.

**Specific questions I asked her:**

***Q:*** I merged responses to questions regarding "what did you find particularly useful in your own education" and "what would you recommend for an ideal degree program" (phase 1 and phase 3 follow-up questions on this topic) or "What would you recommend for an ideal Bachelor's degree program" (Phase 2 and Phase 3 follow-up questions on this topic). Do you agree that it is ok for me to merge these two together? I found a lot of overlapping codes or similar areas, so I felt that this would make sense.

***A:*** Yes. If it is important enough that they remember after all this time, it makes sense they are also recommending it to others.

***Q:*** On EXB's recommendation, I did use multiple codes on the same text segments. That is, within one text segment, 2 or more codes may overlap. For example, while recommending a trait for an "ideal program" a participant would illustrate by describing an element that lacking in their own formal education. This description may go on for a paragraph or so, and contain mention to several aspects (courses, activities, or traits) that I coded as aspects that would be part of an ideal program. Do you agree that this makes sense? I would like to be able to report them both ways –both to point out what was missing in existing programs, and to use this as part of the recommendations for program improvements.

***A:*** It makes sense that you found a lot of overlapping codes between "weren't covered in my program" and "ideal program". It makes sense that you would use it in both those ways.

*[NOTE: I gave some specific examples and the conversation went on longer than implied here but I did not record all of it verbatim.]*

***Q:*** For the themes "Working in Ed SD" and "Formal Education": For some of those areas, specifically the types of software created, types of roles played, and Majors and degree types held by interview participants, I do not intend to directly report on these qualitative findings. They were primarily used to inform development of the survey. Therefore, I plan to report the quantitative findings (which represent a much larger group of participants) and only draw from the qualitative data in these sections for interesting or illustrative quotes.

***A:*** That makes sense.

***Q:*** In the section "Working in Ed Sd", I have separated "important skills, knowledge and attitudes" (things which they indicated were specifically important to them on the job) from "Skills and knowledge unique to Ed SD". The latter were skills and knowledge they felt were unique to working in this domain. Does this make sense, now that you have reviewed the themes?

***A:*** [after reviewing again]: Yes.

***Q:*** Based on your own research and experience, does everything here ring true to you?

***A:*** Yes, it seems that the things you are getting from what your participants said lines up quite a bit with what other people we talked to said and my own feeling about the field and how it works. I don't see anything that goes against what I thought they should be.

***Q:*** Is there anything you see in these findings that surprises you?

***A:*** No, not really. Other than, you know the part where they talked about – the one spot where we were talking about earlier – when we were looking at the Ideal Programs and actual type of degrees they were recommending, the number of people who indicated a hybrid program. My expectation would be that whichever degree they had gotten, that would be the one they would remember. That is the only thing – and it wasn't shocking, just interesting to me. But other than that…there is nothing…too surprising.

# 11 Appendix E: Member checking

## 11.1 Response #1 (Phase 1 participant I1)

Hi Marisa,

Great content. I found the thesis inline with my experiences.

Trivia: I did notice the occasional minor defect in layout, but you are probably filtering yourself.

p113 Review bracket use (
p171 Error!Reference Source not found
p205 Limitation header (move to next page)
And Appendix B was all in bold.
Hope that helps

I would enjoy passing on to the right authorities at <university employed at> as soon as it is published. It does hint at changes in course structure for software engineers.

## 11.2 Response #2 (Phase 3 participant S56)

Marisa

I don't have many comments. I just skimmed chapters 4 & 5. These three things jumped out at me:

Page 112:

"However, participants who either had education in computing fields or were self-trained seemed more confident in their ability to pick up new programming languages and technologies than those with a background in ID or education only."

The word 'seemed' is a rather weak word. I would say that most people who have computing degrees or self-trained were sure that they could pick up a new programming language. Nearly all programming languages are the same it is just syntax and APIs that make them different.

Page 189:

"The only statistically significant difference between groups on the high-level design and low-level design roles is between those with a background in Computing and those with a background in neither area. I cannot explain this finding; since these are quite technical roles; I would have thought that the pattern here would be similar to that seen in software architecture and technical requirements gathering/generation."

I'm not sure if I have mistaken what you are trying to say but I can tell you why high-level design is done by those with a background in Computing: It is all to do with what managers expect these roles to have and they expect them to have a degree and for it to have been in Computing. It is only after many years of experience that someone can dismiss what degree they have and just point a manager at their body of experience on their CV. If you do not have a degree in computing you will not even get an interview for many jobs and if you do it is because your CV has 15+ years of experience showing that you are able to do the job.

It is only after about 15+ years of experience that when I had interviews my degree and its subjects were not mentioned but, of course, I do have a degree in Computing so maybe all was OK just because of that fact.

Page 204:

"For example, study results highlight the importance of user experience design, a relatively new area."

Nothing particularly new about user experience design. One of the projects I worked on from 1990-1997 had a whole team dedicated to getting the user experience correct. Admittedly, it was in the world of defence and involved a lot of interaction modelling to make sure the users of the command and control system on a Royal Navy Frigate could determine what needed to be done from the information presented and take the correct action before the ship was sunk.

## 11.3 Response #3 (Phase 1 participant I5)

Marisa:

Thanks for giving us an opportunity to review your draft. I got through as much of it as I could, enjoyed what I read, and only found one small typo correction for you:

p. 116, section 4.3.1, par. 1, sent. 2: "... so much TO learn, so much to know." (first 'to' is missing)

I'd love to take you up on your offer of a copy of the finished dissertation, too.

# 12 Appendix F: List of Participants Quoted in the Text

**Table 23** ***Interviewees Quoted in the Text***

| | Formal Education | Current Roles Played | | | | | | | Former Work Experience | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Types of Courses Taken | Design (Software Architecture, Requirements, High/Low Level Design) | Development (Programming, DB design, Web design/ development) | User Experience Design | Quality Assurance Testing | Instructional Design | Supervisory | Owner | Experience in Software Design[b] | Experience in ID[b] | Teaching Experience[b] |
| **I 1** | Both | No | Yes | No | Yes | No | No | No | A lot | - | A little |
| **I 2** | Computing | Yes | No | Yes | No | No | Yes | Yes | A lot | A fair amount | - |
| **I 3** | Neither | No | Yes | Yes | No | No | Yes | No | A lot | - | - |
| **I 4**[a] | Computing | No | No | No | No | No | Yes | No | A fair amount | - | - |
| **I 5** | Both | Yes | Yes | Yes | No | Yes | Yes | Yes | A lot | A lot | A little |
| **I 6** | ID/ Education | Yes | Yes | No | No | Yes | Yes | No | A fair amount | A fair amount | A fair amount |
| **I 7** | Computing | Yes | Yes | No | No | No | Yes | Yes | A lot | - | A lot |
| **I 8**[a] | Both | No | No | No | No | Yes | No | No | A fair amount | A fair amount | A fair amount |
| **I 9** | ID/Education | Yes | No | Yes | No | Yes | Yes | No | Unclear | A lot | A little |

[a] Do not meet the criteria for participation based on current roles, but have previously filled relevant roles.
[b] Not all interview participants specified a specific number of years. For those who gave a specific years, "A little" indicates 1-4 years, "A fair amount" indicates 5-10 years, and "A lot" indicates 11 or more years. For those who did not give a specific number of years, I assigned these terms given my sense of the extent of their experience. A "-" indicates that the topic did not come up during the interview and that the individual probably does not have experience in this area.

**Table 24** ***Survey Participants Quoted in the Text***

| | Formal Education | Current Roles Played | | | | | | | | | | | Former Work Experience | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Types of Courses Taken | Software Architecture | Business Requirements | Technical Requirements | High Level Design | Low Level Design | Programming | DB Design | User Experience | Quality Assurance | Instructional Design | Supervisory | Experience in Software Design | Experience in Instructional Design |
| **S04** | Computing | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No | 21+ Years | None |
| **S09** | Computing | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | 21+ Years | None |
| **S10** | ID/education | Yes | No | No | No | No | Yes | No | Yes | Yes | No | No | 21+ Years | 16-20 Years |
| **S11** | Both | No | No | No | No | No | No | No | No | No | No | No | 5-10 Years | 5-10 Years |
| **S12** | Computing | Yes | Yes | Yes | Yes | Yes | No | No | No | Yes | No | No | 16-20 Years | None |
| **S15** | ID/education | No | Yes | Yes | Yes | Yes | No | No | No | Yes | No | Yes | 5-10 Years | 5-10 Years |
| **S17** | Both | No | No | No | Yes | Yes | No | No | Yes | No | No | Yes | 21+ Years | 1-4 Years |
| **S23** | Both | No | No | No | No | No | No | No | No | No | No | No | 21+ Years | 5-10 Years |
| **S31** | Both | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | 16-20 Years | 21+ Years |
| **S32** | Both | Yes | No | No | No | No | Yes | Yes | Yes | Yes | No | Yes | 21+ Years | 1-4 Years |
| **S36** | Computing | Yes | Yes | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | 21+ Years | 11-15 Years |
| **S37** | Neither | Yes | No | No | No | No | Yes | No | No | No | No | No | 21+ Years | 21+ Years |
| **S38** | Both | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | 21+ Years | 16-20 Years |
| **S39** | Computing | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | 21+ Years | None |
| **S40** | Computing | Yes | Yes | Yes | Yes | Yes | Yes | No | No | Yes | No | No | 21+ Years | 5-10 Years |
| **S43**[a] | Both | No | Yes | No | Yes | No | No | No | No | Yes | Yes | Yes | 21+ Years | 21+ Years |
| **S44** | Both | No | No | No | No | Yes | No | No | Yes | No | No | Yes | 5-10 Years | 5-10 Years |
| **S51** | Computing | Yes | Yes | Yes | Yes | Yes | No | No | No | Yes | No | Yes | 21+ Years | 21+ Years |
| **S54** | ID/education | No | Yes | No | Yes | Yes | No | No | Yes | Yes | Yes | Yes | 21+ Years | 5-10 Years |
| **S56**[a] | Computing | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | 21+ Years | None |
| **S57** | Computing | Yes | Yes | Yes | No | No | Yes | Yes | Yes | No | No | No | 5-10 Years | None |
| **S62** | Neither | No | No | No | No | No | No | No | No | Yes | Yes | No | 11-15 Years | None |
| **S63** | Both | No | No | No | No | Yes | No | No | No | No | Yes | Yes | 21+ Years | Under 1 |

| | Formal Education | Current Roles Played | | | | | | | | | | | Former Work Experience | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Types of Courses Taken | Software Architecture | Business Requirements | Technical Requirements | High Level Design | Low Level Design | Programming | DB Design | User Experience | Quality Assurance | Instructional Design | Supervisory | Experience in Software Design | Experience in Instructional Design |
| | | | | | | | | | | | | | | Year |
| **S64** | Both | No | No | No | No | No | Yes | No | Yes | No | No | No | 16-20 Years | 11-15 Years |
| **S66** | Computing | Yes | Yes | Yes | No | No | Yes | No | No | Yes | Yes | No | 5-10 Years | None |
| **S67** | ID/education | No | Yes | Yes | Yes | No | No | No | No | No | No | Yes | Under 1 Year | 1-4 Years |
| **S68** | Both | No | No | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | 5-10 Years | 16-20 Years |
| **S69** | ID/education | No | No | No | No | Yes | No | Yes | Yes | Yes | No | Yes | 21+ Years | 1-4 Years |
| **S73**[a] | ID/education | No | Yes | Yes | Yes | Yes | No | No | No | Yes | No | Yes | 21+ Years | 1-4 Years |
| **S74**[a] | Both | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | 21+ Years | 21+ Years |
| **S75** | ID/education | Yes | Yes | Yes | Yes | Yes | Yes | No | No | Yes | No | Yes | 16-20 Years | 16-20 Years |

[a]Participated in follow-up interview

# 13 Curriculum Vitae

**Marisa E. Exter**
522 W. Skyline
Bloomington, IN 47404
(812) 333-3986
mexter@indiana.edu

## EDUCATION

***PhD in Instructional Systems Technology, Computer Science Minor***
Dissertation topic: Educational Background of Designers/Developers of Educational Software

***Masters in Computer Science***, Illinois Institute of Technology, December 2003

***Bachelors in Computer Science***, Elmhurst College, May 1999

## TEACHING EXPERIENCE

***Adjunct Faculty Member,*** Ivy Tech Community College, CIS-157: Web Site Development, Fall, 2006.
Adapted and taught course. Course activities included interactive lectures on web design and practical workshops on use of web-design tools, independent work and coaching time, and student presentations.

***Technical Coach and Assistant Instructor,*** EDUC-R541: Instructional Development & Production I, Spring 2006.
Lead synchronous critique sessions for blended graduate class of residential and online distance students. Provided technical support and assistance throughout semester.

***Co- Instructor,*** Indiana University, EDUC-K361: Assistive Technology in Special Education, Fall 2005.
Assisted in design of and co-taught residential undergraduate course related to use and evaluation of AT.

## PROFESSIONAL WORK EXPERIENCE

***Director of Design, Development and Testing and Member of Research Team,*** Critical Web Reader Project, Indiana University, 2010-present
Participated in high-level project planning for new product development and services. Day-to-day management of team of 5-9 members. Developed high-level and database design documents for enhanced toolset. Conducted build and testing for new features. Participated in research and grant writing.

***Lead Developer (Graduate Research Assistant),*** Critical Web Reader Project, Indiana University, 2005-2010
Served as Lead Designer/Developer and Research Assistant in the development of a tool to promote critical reading of web-based materials in educational settings. Assisted in research and grant writing.

***Instructional Designer (Intern, Full Time)*** DyKnow, May-August 2005
Created white papers and evaluation tools for the DyKnow Vision™ interactive note-taking system.

***Software Engineer,*** Lucent Technologies, 1999-2004
Served as designer and lead programmer on guided installation and configuration system, and developer of system stability infrastructure for a family of multi-processor telecommunication products. Worked closely with systems engineers, testers, and technical writers throughout the life-cycle of each product.

***Website developer,*** Arthur Anderson, May-December, 1999
Maintained dynamic components of company website and developed Y2K testing for company website.

***Software Developer (Intern),*** Lucent Technologies, 1997-1999
Participated in a team which designed and developed a suite of change control software. Ported hardware test-bed software to a graphical user interface.

## PEER-REVIEWED PUBLICATIONS

Exter, M. E. & Turnage, N. M. (In Review). *Exploring experienced professionals' reflections on computing education.* Manuscript submitted for publication to Transactions on Computing Education.

Damico, J.S., Baildon, M., Exter, M., & Guo, S. (December 2009/January 2010). Where we read from matters: Disciplinary literacy in a 9th grade social studies classroom. *Journal of Adolescent & Adult Literacy*, *53*(4)

Exter, M.E., Korkmaz, N., Harlin, N.M., & Bichelmeyer, B.A. (2009). Distance education students' responses to sense of community within a fully online graduate program. *Quarterly Review of Distance Education, 10(2),* 177-194.

Exter, M. E., Wang, Y., Exter, M. F. & Damico, J. S. (2009). Designing a tool to support critical web reading. *Techtrends*, *53*(1), 23-28.

Exter, M.E., Harlin, N.M., & Bichelmeyer, B.A. (2008). Story of a conference: Distance education students' experiences in a departmental conference. *Internet and Higher Education, 11* (1), 42-52.

## EDITED BOOK SECTIONS

Exter, M. E., & Flick, J. (2010). Engaging online graduate students through volunteering: Pitfalls and possibilties. In J. A. Jaworski (Ed.), *Advances in Sociology Research* (Vol. 9): Nova Science Publishers, inc.

## PUBLISHED CONFERENCE PAPERS

Exter, M. E., & Ochoa, T. A. (2006). Interactive assistive technology: a preliminary analysis of the use of DyKnow Vision and Wacom Graphire 3 4X5 USB tablets in a special education teacher preparation course. In D. A. Berque, J. C. Prey & R. H. Reed (Eds.), *The impact of tablet PCs and pen-based technology on education* (pp. 57-65). Purdue, Indiana: Purdue University Press.

## NON-PEER-REVIEWED PUBLICATIONS

Exter, M. E. (2005). A research based approach to encouraging effective note-taking: Best practices and supporting technologies. Retrieved January 1, 2006, from http://dyknow.com.

Exter, M. E. (2005). Encouraging active learning: Best practices and supporting technologies. Retrieved January 1, 2006, from http://dyknow.com.

Exter, M. E. (2005). Providing timely feedback: Best practices and supporting technologies. Retrieved January 1, 2006, from http://dyknow.com.

Exter, M. E. (2005). Encouraging effective note-taking: An annotated bibliography. Retrieved January 1, 2006, from http://dyknow.com.

Exter, M. E. (2005). Facilitating active learning: An annotated bibliography. Retrieved January 1, 2006, from http://dyknow.com.

Exter, M. E. (2005). A research based approach to providing useful feedback : An annotated bibliography. Retrieved January 1, 2006, from http://dyknow.com.

**JURIED PRESENTATIONS**

Exter, M. E. (2010, October). *The Educational Experiences of Software Designers working in Education Related Fields.* Round table presentation at the annual meeting of the Association for Educational Communications and Technology, Anaheim, CA.

Exter, M. E., Flick, J. (2010, October). *Engaging Online Graduate Students through Volunteering: Pitfalls and Possibilities.* Paper presentation at the annual meeting of the Association for Educational Communications and Technology, Anaheim, CA.

Exter, M.E., & Harlin, N.M. (2010, April). *The Formal and Non-Formal Educational Experiences of Software Designers*. Poster session presented at the American Educational Research Association (AERA). Denver, CA.

Exter, M.E., & Harlin, N.M. (2009, October). *The Formal and Non-Formal Educational Experiences of Software Designers.* Paper presented at the annual meeting of the Association for Educational Communications and Technology, Louisville, KY.

Harlin, N.M., Exter, M.E., & Boling, E. (2009, October). *Software Designers' Use of Precedent.* Paper presented at the annual meeting of the Association for Educational Communications and Technology, Louisville, KY.

Exter, M. E., Korkmaz, N. & Boling, E. (2009, October). *Use of Critique in an Instructional Design Course: Perceived Value and Impact on Students' Design Thinking.* Paper presented at the annual meeting of the Association for Educational Communications and Technology, Louisville, KY.

Korkmaz, N., Exter. M. E., Harlin, N. M. & Bichelmeyer, B. A. (2009, October). *Students' Feelings of Sense of Community within an Online Graduate Program.* Paper presented at the annual meeting of the Association for Educational Communications and Technology, Louisville, KY.

Harlin, N.M., Exter, M.E., & Boling, E. (2009, February). *Software Designers' Use of Precedent.* Poster presented at the annual meeting of the American Educational Research Association. San Diego, CA.

Damico, J. S., Baildon, M. C., Yazzie-Mintz, T., Riddle, R. L., Exter, M. E. (2009, February). DiverseITy (Diverse IT): Using a Technology Tool for Curriculum Innovation in TE. Interactive symposium session at the annual meeting of the American Educational Research Association. San Diego, CA.

Exter, M. E., Korkmaz, N., Harlin, N. M., Bichelmeyer, B. A. (2008). *Distance Education Students' Responses to Sense of Community within a Fully Online Graduate Program.* Paper discussion at the annual meeting of the American Educational Research Association (AERA). New York, NY.

Exter, M. E., Wang, Y., Exter. M. F., Damico, J. S. (2008). *Designing a Tool to Support Critical Web Reading.* Paper discussion at the annual meeting of the American Educational Research Association (AERA). New York, NY.

Harlin, N. M.., Exter, M. E. & Bichelmeyer, B. A. (2008). *Story of a Conference: Distance Education Students' Experiences in a Departmental Conference.* Poster presented at

the annual meeting of the American Educational Research Association (AERA). New York, NY.

Korkmaz, N., Exter, M. E. & Bichelmeyer, B. A. (2008). *Students' Thoughts about Their Interactions with Peers and Peer Feedback in a Blended Course: A Case Study.* Paper presented at the annual meeting of the American Educational Research Association (AERA). New York, NY: 2008.

Korkmaz, N., Exter, M. E. & Boling, E. (2008). *Students' Perceptions of Peer Critique in a Blended Instructional Design Course.* Paper discussion at the annual meeting of the American Educational Research Association (AERA). New York, NY.

Exter, M. E. & Ochoa, T. A. (2007). *The use of an interactive note-taking system: A pilot study in a teacher education course.* Paper and poster presented in special poster session at the annual meeting of the American Educational Research Association (AERA). Chicago, IL.

Cheng, J., Exter, M. E., Korkmaz, N., Clark, L. V., Tian, L., Yoon, S. & Bichelmeyer, B. (2007). *Introducing the logic model as a framework for distance education program evaluation.* American Educational Research Association (AERA). Chicago, IL: 2007.

Damico, J. S., Baildon, M. C, Exter, M. E. & Guo, S. (2007). *Accessing prior knowledge and adjudicating between different perspectives: Students examine competing websites in social studies.* Paper presented at the annual meeting of the American Educational Research Association (AERA). Chicago, IL.

Exter, M. E. & Ochoa, T. A. (2006). *Interactive assistive technology: A preliminary analysis of the use of DyKnow Vision and tablet pens in a special education teacher preparation course*. Paper presented at the annual Workshop on the Impact of Pen-Based Technology on Education (WIPTE). Purdue, IN.

Treat, A. R. & Exter, M. E. (2005). *Virtual Schools for the Gifted.* Poster presented at the annual conference of the National Association for Gifted Children (NAGC). Louisville, KT: 2005.

## FUNDED GRANTS

***Office of Educaitonal Research, Ministry of Educaiton, Singapore: Funded project: "Using Web-based Tools to Support Source Work and Inquiry in Social Studies"*** (2009-2012)
I contributed to the planning and wrote portions of the 3-year, $450,000 (Singapore) grant application, as well as contributing to a contract made between the National Institute of Education in Singapore and Indiana University.

***GPSO Travel Grant,*** Indiana University, 2008
This was a university-wide grant competition. I received $300 to partially funding travel expenses to attend and present at a national conference.

***Larson Award (Travel Grant) Recipient***, 2008
This was a department-wide grant competition. I received $500 to fund travel expenses to attend and present at a national conference.

## PROFESSIONAL SERVICE

***Graduate Student Mental Health Working Group.*** Indiana University, 2010-2011.
Member of evaluation sub-committee of a university-wide working group charged with addressing mental health issues amongst the Indiana University graduate student population.

***AERA SIG Design and Technology Officer (Web Content Manager).*** 2008 - 2011
Produce/update materials on SIG-DAT's website.

***IST Conference, Advisor to Chair & Member of Submission-Review Committee,*** Indiana University, 2008.
Assisted current conference chair in the development of the conference. Submission-Review committee activities included assigning reviewers for submissions, making final decisions on submission acceptance, and assisting with conference scheduling.

***IST Conference Chair*****,** Indiana University, 2007.
Organized two-day professional conference for residential and distance students, faculty and alumni (total attendance of approximately 125). Conference activities included a keynote speaker, panel discussion of invited speakers, presentations of research and practice by participants, a job fair, social gatherings, and special interactive sessions.

***IST Conference, Breeze coordinator,*** Indiana University, 2006.
Coordinated live and recorded Breeze sessions, which allowed Distance Education students, alumni, and others to participate remotely

***DyKnow Vision Workshops,*** Indiana University, 2005-2006***.***
Provided workshops on the use of interactive note-taking systems in the classroom to a number of undergraduate classes, as well as an open workshop for professors and graduate students.

***IST Conference, Chair of Food Committee,*** Indiana University, 2005.

## REVIEWS

***Invited for Peer-Review of Journal articles***

**Software: Practice and Experience,** 2011.
**THEN journal,** 2008.
**TechTrends**, 2005.

***Volunteered for Peer-Review of Conference proposals***

**American Educational Research Association (AERA),** 2009.
Conference proposals (3 for Division J, 2 for SIG-DAT), Session proposal (1 for Division J)
**Workshop on the Impact of Pen-based Technology on Education (WIPTE)**, 2008.
Conference proposals (3).
**American Educational Research Association (AERA),** 2008.
Conference proposals (2 for each of 2 SIGs).
**IU IST Conference,** 2008.
Conference and session proposals (2) and member of submission-review committee.
**American Educational Research Association (AERA),** 2007.
Conference proposals (2 for each of 2 SIGS).
**Workshop on the Impact of Pen-based Technology on Education (WIPTE)**, 2007.
Conference proposals (2).
**IU IST Conference**, 2007.
Conference proposals (11) and member of submission-review committee.

## PUBLIC SERVICE

***Adoption Councilor***, Monroe County Humane Association, 2004-present
Interview clients and provide education and assistance in selecting appropriate pets.

## ASSOCIATIONS

***American Educational Research Association (AERA), former SIG-DAT officer***
***Association for Educational Communications and Technology (AECT)***
***Graduates in IST (GIST)***
***Association for Computing Machinery (ACM), member if SIG-CSE and SIG-ITE***
***Institute of Electrical and Electronics Engineers (IEEE)***
***Phi Kappa Phi honor society***

## HONORS, AWARDS, AND SCHOLARSHIPS

***Chancellor's Fellowship***, Indiana University, 2004-2008
***Graduated Summa Cum Laude***, Elmhurst College, 1999
***Elmhurst College Honors Program***, Elmhurst College, 1996-1999
***Dean's List***, Elmhurst College 1995-1999
***Dean's Scholarship***, Elmhurst College, 1995-1999

## TECHNICAL SKILLS

***Operating Systems:*** Unix, Windows 9X, ME, 2000, XP, Vista
***Technologies:*** HTML/XHTML, CSS, CGI, SQL
***Programming Languages:*** Perl/Perl-Tk, C#, C, C++, Java, Visual Basic, Javascript
***Web Design:*** MS .NET Visual Studio, Adobe DreamWeaver, DotNetNuke, SharePoint
***Data Analysis Software:*** SPSS, NVivo
***Other Skills:*** OOD, OOP, database design, software engineering/end-to-end project planning experience, technical project management