AC 2008-1014: AWAKENING INTEREST AND IMPROVING EMPLOYABILITY: A CURRICULUM THAT IMPROVES THE PARTICIPATION AND SUCCESS OF WOMEN IN COMPUTER SCIENCE

Yvonne Ng, College of St. Catherine

Yvonne Ng, M.S.M.E, teaches computer science and engineering for non-majors at the College of St. Catherine. Educated as a mechanical and aerospace engineer, she worked in industry as an automation design engineer and contract programmer. She made computer science a more appealing topic for her all-women undergraduate student body by presenting this technically valuable course in a more comprehensive manner. She is currently the coordinator of the Center of Excellence for Women, Science and Technology where she administers the college's National Science Foundation and Clare Boothe Luce scholarships for Science, Technology, Engineering and Mathematics (STEM) majors and facilitates various recruiting, advising and placement activities for STEM majors and minors.

Awakening Interest and Improving Employability:

A Curriculum that Improves the Participation and Success of Women in Computer Science

Today more than ever, engineering and technical fields must be accessible to multiple populations. The challenge of creating a diverse workforce, however, involves more than simply putting a variety of bodies in a classroom. What is taught and how it is taught greatly influence whether students will continue taking technical courses.

The College of St. Catherine has developed a model of teaching computer science (CS) to students that both motivates them to take classes beyond those required and makes them more employable. The traditional CS curriculum is infused with industry requirements, guided by the established Parallel Curriculum Model (PCM).

Since 2000, 37% (36/84) of the women who took the first CS course at the College ultimately enrolled in at least 3 CS courses or took at least one more CS course than was required by their majors. The College of St. Catherine—the nation's largest college for women—succeeded despite the fact that most students enrolling in the courses had no intention of majoring in CS. How were so many women motivated to take more CS courses during a time when many CS programs across the nation lost students—especially women—starting with the introductory course?

To answer this question, a survey was sent out to students/alumnae with a 55% response rate (20/36). Qualitative analysis was used to determine the themes that emerged. Strategies for instruction and evaluation were then identified for 1) encouraging students to continue in their CS education and 2) improving their employability.

Although many participants took the first CS course because of a requirement, four main strategies motivated them to take more CS courses:

1) Creating engaging and interactive ways of learning core concepts and typical common practices needed and expected by employers in the industry, such as using and developing test code, troubleshooting, and design documentation
2) Using projects to create a meaningful product that used core concepts and developed transferable skills, such as team work, project management, and communication skills
3) Building a social community *within* the learning environment that supported and motivated students throughout their computer science education. This was

essential in a program that had few to no majors at any given time to tutor students.
4) Incorporating career development, such as resume, portfolio and evaluations, explicitly into the lessons to prepare students for obtaining, securing, and succeeding in a job or computer-based career.

Seventy-five percent of the study participants worked or are currently working in an Information Technology (IT) related position. They cited the above four elements as invaluable for their initial employment and continued success.

PCM guided our curriculum development to ensure that all aspects necessary for a well-educated student were addressed. This model was originally developed for gifted and talented education. However, it shows promise for technical curricula that prepare students with the essential skills needed to succeed in industry and that awaken interest and excitement about computer science, a field that is quickly losing students, particularly women.

1 Introduction

Insanity is doing the same thing again and again and expecting a different result.
~ Diamond commercial

1.1 National Crisis: Computing Expertise Needed

In 2006, the National Center for Women and Information Technology reported that only 50% of the 1,000,000 additional computer and information-related jobs anticipated by 2014 will be filled by U.S. computer science bachelor's students.[7] As Caroline Simard, a research associate for the Anita Borg Institute for Women and Technology reports, "Despite popular beliefs about the impact of offshoring on hi-tech jobs, numbers show that the demand for high-level high-tech jobs such as software engineers have increased since 2000." [10, p. 1]

These institutions, along with others, advocate recruiting women as an untapped source of talent. With fewer girls choosing CS and fewer women completing computer-related degrees[6], this may seem like a long shot. Still, a successful investment strategy may involve both long-term strategies (with a focus on girls) and short- to mid-term strategies. The latter requires giving women an educational foundation that allows them to enter computer-related professions after completing a college degree that may not be in CS.

Doing this would open more doors since women often choose a computer-related education later in life. To many, computers are more of "an 'acquired taste' that emerges over time. … [T]hey may come to computing at a later stage in their education, perhaps after having majored in some other discipline." [3, p. 3] Women who have taken at least some CS courses have an easier time following this path later in their education. Thus, one way to approach the impending crisis is to "sow seeds" by encouraging women to take more CS courses, whether or not they plan on making it their major.

At the College of St. Catherine, a number of majors require students to take at least one computer science course. What they experiences in that course is key. Normally introductory courses (CS1) focus on programming exclusively, requiring multiple discrete applications to be implemented. Alternatively, they cover a wide span of topics from hardware to Turing engines to ethics. With the failure of many introductory programs to entice students, particularly women, to continue in computer sources, this traditional approach may not be the most productive.

1.2 The Value of Educating Women

> *Women are the canaries in the coal mine.* ~ Lenore Blum, Carnegie Mellon University Computer Scientist

Encouraging women, especially those not intending to major in computer science, to take even one CS course is a challenge. Computer science educators across the country puzzle over how they can increase the number of participants, and eventually the number of graduates. They believe that several factors are responsible for the decline, including

- CS has a negative public image (geeky, isolated, mathematical, difficult) [9]
- CS is represented as mainly programming, which has a negative image in students' minds. [4]

Educators seek ways to pull women into the field because "factors driving women away will eventually drive men away as well." [4] Modifying the CS curriculum so that it will encourage women to engage in more coursework may entice other underrepresented populations—and maybe even more men. Thus, experimenting with women has broader impacts than just getting more women into the field.

As a women's college, the College of St. Catherine is inherently interested in the challenges related to gender. Literature shows that women have different motivations for entering computer science and staying in, such as understanding the context of learned knowledge and social implications.[1] Additionally, while boys can be motivated purely by the "end goal," girls respond better to a positive social experience. Thus, motivating men to "stick it out" in a discipline can be largely accomplished through job shadowing and career planning, but women also need a group of people they enjoying being with. This explains how girls who have a great math or science club experience in high school can become discouraged at a college where they feel isolated in their math or science classes, as well as why women graduating from highly active science and math programs in college find industrial work without a social aspect to be undesirable. [5]

The College of St. Catherine has the added challenge of not having a CS major. Students who decide to major in CS take 1st and 2nd year courses at our institution and then complete their degrees through one of the partner colleges in the Associated Colleges of the Twin Cities (ACTC). Thus, the focus of our courses must serve both the potential CS majors and the non-majors.

Finally, due to financial obligations, our students often must obtain jobs soon after graduation. With some computer science knowledge under their belts, their employability, and often their

earnings, are greatly improved. Thus, we decided to look to industry for our curriculum guidelines.

Not all technical jobs are the same. In fact, there are different "micro-climates," ranging from management information systems (MIS) departments for non-IT companies to industrial research and development to start-up companies. Each requires different technical and professional knowledge and skills. In fact, the former (MIS departments) make up 79% of all IT jobs today. [9]

Our strategy was to create a technical program that would prepare a liberal arts major to succeed in jobs that bridge technical and non-technical personnel. If a student found a technical job interesting, she would have the confidence and foundation needed to take more CS courses or even pursue a degree in a computer-related area later in her career.

1.3 Industry's Knowledge Gap

In 2000, IEEE's *Computer* magazine conducted a survey of software professionals to determine "which educational topics have proved most important to them in their careers and to identify the topics for which their education or current knowledge could be improved." [6 p. 44] These findings identified which parts of the standard CS curriculum to keep and which to modify.

According to the findings, core knowledge such as data structures, algorithm design, and specific programming languages were both important and taught in the standard curriculum. However, professionals indicated three of the four top areas were either learned on the job or had been forgotten since formal education: 1) software and design patterns, 2) object-oriented concepts and technology, and 3) requirements gathering and analysis. Other top areas learned on the job included analysis and design methods; testing, verification, and quality assurance; project management; confirmation and release management; human-computer interaction/user interfaces; and databases.

Business and art topics of high importance in the respondents' careers, but which were learned on the job, included 1) ethics and professionalism, 2) technical writing, 3) giving presentations to an audience, and 4) leadership. Today, this need for a well-rounded, project-based computer professional who can communicate and work well with others has not changed. Numerous reports from industry indicate the need for "technology workers with more experience and a broader set of skills such as leadership and interpersonal communication skills." [10]

Interestingly, these high knowledge gap areas are the very topics that could create a social atmosphere that would *retain* women. If these topics were taught as integral parts of a computer science program, the students would better see how these "other" skills are useful in the technical arena. This approach is more complete than having students take separate courses in public speaking, technical writing, or management and expecting them to apply these skills to a computer-related project. At the College of St. Catherine, we decided to use projects to focus the courses in our minor program.

1.4 Overview of Paper

Section 2 describes the curriculum created, starting with how the Parallel Curriculum Model (PCM) and the educational methodology of moving the novice student to the professional was

used to develop a comprehensive computer science minor program. Section 3 describes Grounded Theory Methodology which was used to independently determine the success of the designed curriculum in achieving the objectives of increased employment opportunities, increased desire to learn computer science, and a solid foundation to learn CS in the future. It also presents a summary of the findings, the model that was created from the qualitative data, strategies and consequences that emerged from the implemented curriculum. Section 4 discusses the findings in light of the design objectives and the transferability of the findings to other populations and areas. The qualitative results are detailed in the Appendix.

2 What We Created: More Than Just Assigning a Project

Great teachers are the engineers, not technicians, of education.

2.1 To Educate, Not Train: The Parallel Curriculum Model and the Novice to Professional Methodology

Becoming project-focused caused some concerns in faculty. The shift to more concrete tasks from theoretical ones may result in training students rather than educating them. Technicians who are trained are taught an existing system so they can work within that system or execute that system, for example, working on or fixing a jet engine. Engineers, however, are educated in the underlying principles so that they can create new systems or leverage emerging technologies in an existing system. For example, an engineer can design a new jet engine or can use new materials or scientific phenomena to improve existing engines.

This is not to say that a technician could never create a new jet engine – or contribute to the development of one. With experiences, technicians can often close the knowledge gap that exists between engineers. In the Novice-to-Professional methodology of instruction, this happens frequently, with students entering the educational system as novices, learning on the job to the Apprentice level, then the Practitioner and then Expert where they are then viewed as a budding professional. Technicians, in fact, build from the concrete knowledge to the more abstract.

In many ways, students, especially those who are non-majors, progress better from a more concrete approach at first so they can see immediate results. This is another advantage of a project-focused curriculum.

Curriculum models allow educators to put knowledge about intelligence, knowledge, learning, thinking skills, and curriculum into a framework that ensures that students do not remain only in the concrete tasks. They remind the curriculum designer of the underlying principles that transition training to education.

The Parallel Curriculum Model (PCM) is based on several theories of cognitive psychology and educational pedagogy. PCM classifies the knowledge of professionals into four categories or parallels[6], which are:

- **Core**: Concepts, understanding, and knowledge that a professional is expected to know. For example, in computer science, this includes programming constructs such as input,

output, variables, arrays, conditionals and loops; object-oriented concepts; typical algorithms and data structures; and the von Neumann computer architecture model.

- **Practice**: Skills and methods that require the application of core concepts to typical problems encountered by the professional. In computer science, this includes designing with flowcharts or pseudocode; testing, verification, and quality assurance; requirements gathering and analysis; release management; and risk analysis.
- **Connection**: This includes the ability of the professional to connect with other disciplines, cultures, or perspectives across different locations or even time. This is an area that liberal arts majors usually excel. With the growing influence of technology on society, it is becoming more important to accreditation organizations. In computer science, this can include connections to art and psychology for product presentation or user interface design; connections to communications and business for presentations, project management, or team coordination; and connections to engineering and mathematics for microprocessor architecture and algorithm analysis.
- **Identity**: This is the ability of students to see themselves as professionals in the field. For non-majors, it is the ability to see how their interests, majors, or career goals intersect with professionals in these fields. It also relates to confidence in understanding or contributing to the field. In computer science, this can include the belief that computer science is attainable, understandable and useful. [8]

PCM provides a way to frame the curriculum of each course in a major or minor. Instructors use the parallels to determine the primary and secondary priorities which are then reflected in the evaluation and instructional activity design. Identifying priorities allows the instructor to be flexible and make changes "on the fly" if students lack assumed abilities or if they learn the required concepts quickly and can handle more challenges.

2.2 Objectives: Employment, Desire, Foundation

PCM language clarifies the educational value of projects in a computer science curriculum with respect to the objectives. The ability to work on projects develops **employability** because students use, *practice*, and *connect core* knowledge in a project. If students can chose a project that they can *identify* with, they will have the **desire** to learn more and to make the best possible product. By using *core* knowledge and common *practices* to create a successful concrete product, students demonstrate the **academic foundation and confidence** needed to continue in CS.

Since we are women-focused, two additional aspects were added to the project-focus of the courses. PCM analysis articulated how these requirements educated our students into computer professionals:

1. Each course requires students to work on a **team project**. Since a social context is important to women, we believed this to be a necessary condition for students' confidence or *identity*. Industry *practices*, however, made this a necessity regardless of individual students' actually desires to work in teams. We required this of our students because we wanted to ensure our students had the fundamental instruction and expertise with teams. Team projects can be more complex. The *core* concepts become tools students utilize to create more impressive products. Common *practices* like UML

documentation and version control allow them to be more effective teams. With team management and negotiation skills, they can better *connect* to each other and still respect their individual differences in working style, creativity and problem solving. With more complex products, more impressive products are implemented. An impressive project motivates most students by generating pride in their work. This, in turn, allows them to *identify* themselves as budding professionals.

2. Projects in industry always have **purpose**, usually tied to marketing requirements. Whenever possible, we required projects to serve some customer's needs: A teacher trying to drill students in history facts, a gaming company trying to break into the young girl computer game market, or a campus office or organization wanting computer tools. We established computer science as a way to help others. Because their products must satisfy a particular person's needs, students must *connect* with the client by using their project management and oral and written communication skills. Students decide which of the many *core* concepts they know will best serve the client and realize the *practices* help them meet the required deadlines. By helping others who couldn't have made the product themselves, students are identified by others and *identify* themselves as having technical expertise.

2.3 Design Considerations

For a course to be project-focused, the instructor must do more than simply assign a project.

1. **Instructional activities must tie to the project**. Just as the project must have purpose, the class activities and assignments must have purpose. This shifts students' attitudes towards coursework from "Why are we doing this?" to "When are you going to teach us this – soon, I hope!" This arrangement also helps students identify gaps in their knowledge and motivates them to use all their resources (instructor, text, references, tutors, each other) to fill that gap. Specific ways we addressed these issues were:
    a. **Lectures explain concepts needed to achieve basic requirements and help connect this material back to the project.** For example, particular graphical user interfaces (GUI) can be demonstrated and manipulated to satisfy the basic requirements so students can see how these concepts link together to satisfy the project's purpose. This helps students realize that the basic task is achievable.
    b. **Labs introduce and develop common practices which will make projects easier to do or will allow students to earn higher grades**. For example, labs teach students how to use and modify test code. This builds the skills needed to achieve basic requirements.
    c. **Homework has students practice core concepts or common practices on their own to prepare for the projects.** For example, a homework assignment has students learn a new command and write test code for it. This builds the skills needed to learn new commands so students can earn higher grades on the project.

2. **Explicit instruction is needed to transform novices to apprentices.** Each student usually enters a new course as a novice. Any time students must perform higher than novice, some instructional activity must be done to get them to at least an apprentice (C-grade) level. Other activities, such as homework problems or lab challenges can develop the underlying principles to help students achieve practitioner (B-grade) or expert (A-

grade) levels. While most instructors are aware of this development for technical topics, they don't always identify the non-technical ones that can affect a project-based curriculum. Specific ways our program addressed this were:

a. **Project phase reviews are required to model project management skills**. Students usually are novices to project management; thus it is necessary to model this process for students. Phase reviews are an industry practice where team embers (student teams) check in. In the course, students are told what they need to report and demonstrate. The project manager (instructor) then evaluates whether they are where they should be. Suggestions, corrections, reprimands, and contingency plans are shared or developed during the review. In this way, students learn project management by example.
b. **Team process skills are taught.** Too frequently, instructors simply throw students in teams, expecting that they will figure out how to work together. Since teamwork is required for a good product, time needs to be taken to instruct students how to divide tasks, communicate progress, and manage versions.

3. **Projects are mindfully designed with particular objectives in mind.** Project requirements and evaluations should assess the appropriate parallels. Remember that they can also develop desired behaviors.
   a. **Projects must require certain core knowledge.** For example, for projects to get a C, they must input something from the user, do some mathematical manipulation with the inputs, use this information to make decisions, and output different values based on the decisions. Utilization of more core knowledge (e.g. use of arrays, objects, or new commands) would result in a higher grade.
   b. **Projects must allow students to use their creativity**. By defining general product requirements, listing measurable basic requirements, and providing examples of how this could be achieved, students have room to create something they identify with. For example, the product must be an educational grid-based game that utilizes a two-dimensional grid, objects, and read the educational questions and answers from a file. Examples include Connect-4, Jeopardy, or Othello.
   c. **Projects must be complex enough to require teams and cannot be done at the last minute.** This motivates team interdependence and fosters project management and planning skills.
   d. **Projects are evaluated based on absolute achievement of requirements, not relative performance to create a cooperative environment among students.** In industry, several teams must often work together so that the whole company can benefit. An absolute evaluation system encouraged a cooperative environment: If one team figured out a concept, they were encouraged to teach the concept to the other team, who then learns it, modifies it, or rejects it for the particular of its own project. The first team does not withhold the information from fear that other teams will out-perform it, and members reinforce their knowledge by teaching others the concept.
   e. **Public presentations of final projects develop communication and project management skills and allow students to be self-motivated to learn more.** Presenting to an external audience keeps students on schedule, forces risk

management, and motivates students to learn good communication skills so they can show off their hard work. Additionally, since lectures indicate the lowest bar of achievement, most students are personally motivated to learn what they need to achieve higher levels in front of their peers.

Effective curriculum design recognizes that students can't learn everything at once. In other words, all aspects cannot be equally important in a single class. Although ideally, students achieve as many objectives as possible, the objectives must be prioritized. PCM was used to streamline the 5 courses that make up the CS minor (Table 1). As students progress through the minor courses, different parallels are emphasized.

Table 1: Computer Science minor designed using Parallel Curriculum Model

	CSCI 111 Algorithms and Programming I	CSCI 112 Algorithms and Programming II	CSCI 207 Computer Organization Principles	CSCI 208 Algorithm Analysis and Data Structures	Independent Study (see notes below)
CORE	** I/O Variables & Arrays Control structures Objects & Functions	* Object design & impl List, queue, stack ADT Binary trees & BST Recursion Search, sort algorithm Big O Analysis (basic)	** Data representation Data manipulation Assembly language Von Neumann arch Datapath Optimization techniq	** Big O Analysis & recurrence relations Standard algorithms (incl. Greedy, stable marriage, salesman) Heaps, priority queues, graphs, hash tables	*May learn new core material or may use existing knowledge in new way*
PRACTICE	* Using Test code Testing	** Data design techniques Risk analysis	Implementation • Considerations • Robustness • Efficiency • Cost effectiveness • Size/Scalability	Project management	* *Put all skills from minor into practice*
CONNECTION	Timelines Presentations	Team coordination UML documentation	* Electronics basics Software design considerations	* Mathematical proofs Analysis's role in design process (Working with marketing when avail)	*Connect to major, future career, or own interests*
IDENTITY	Confidence in basic skills Excitement about what can be done with computers	Confidence in using common designs Excitement about building larger products	Confidence in how programs "get into" hardware (take magic out)	Confidence in choosing appropriate design for problems (Confidence in working with non-technical partners)	** *Chose topic of interest to "get obsessed" with* *Confidence in completing product with a purpose*

** are primary focus parallels for the course * are secondary focus parallels

Notes about independent study: 1) If taken after 3 courses, student knows process but design of final product is unknown and will be designed by student, 2) if only 2 courses under belt, instructor needs to be more explicit about the missing practices (see chart) during the project. An idea of possible final designs should be discussed at the beginning of the project so student doesn't go in blind.

For each course, the primary and secondary objectives were always clearly identified. This allowed the instructor to modify the instruction as necessary for a class or even an individual. For example, though students were expected to start team projects in CSCI 111, some students needed more work with the core concepts. During individual performance evaluations, they were encouraged to stay "on probation," doing extra homework problems ("probie preps") in lieu of the project so that they had time to solidify the concepts and do better on exams. Meanwhile, the students who were doing satisfactorily on core concepts moved on to team projects that would help them solidify their understanding and make them more employable. This way, each student was challenged within her abilities and supported when needed.

3 What We Accomplished: Methodology and Findings

Educating women to lead and influence.

-- Mission statement of College of St. Catherine

3.1 Research Questions

Now that the curriculum has been developed and in place for about 7 years, the main research question is: "Did this PCM-designed curriculum achieve the objectives?" In particular, the objectives were:

- Does the computer science curriculum improve students' employability?
- Do the students have a desire to take more computer science courses?
- Do students have the foundation needed to be able to learn more CS in their future? In this case, we are interested in both the academic foundation and the confidence needed to learn more.

3.2 Small Numbers: Institutional Data is Not Enough

Since one of the original objectives was to increase women's desire to take CS courses, it is wise to look at the number of women who have the potential to major in CS, but are not necessarily planning to. The enrollment in CSCI 111, the first programming (CS1) course, is a good metric to determine this pool of potential majors. Since 2000, 84 students have enrolled in this class.

Most studies look only at graduating majors. If we did this, the numbers are discouraging: 4 CS majors and 9 information systems majors graduated since 2000. This constitutes 15% of the initial pool. However, it is difficult to compare these to national co-educational statistics since our population is 100% women and most national numbers are reported in percentage of women in a co-educational environment.

Further complicating the situation is the fact that due to their majors, some students are required to take some CS courses. Thus, the pertinent question for measuring success in desire to take computer science is how many students took classes beyond what was required. This was not

easily determined by enrollment in one specific course because some majors required a second CS course.

3.3 Who to Study and Why

We decided to survey the population of students who took *at least* 3 computer science courses *or* who took at least 1 course *more* than what was required for their majors. This population (Table 2) included CS majors and minors, information systems (IS) minors, students who majors in IS when it required 3 CS courses, and those students we knew who took one more CS course than required for their major without receiving a formal minor. This also included students who took the CS course for non-majors (CS0) and who went on to take CSCI 111 as well as students who have completed CS courses beyond their major requirements but who may not yet have graduated. This totaled 36 students or 43% of the original pool of 84.

Table 2: Surveyed Population

4	CS majors
5	CS minors
3	Math/CS minors
5	IS majors (3 CS courses)
8	IS minors
1	IS major (took CS0 class)
5	Took more than required
5	in pipeline
36	surveyed

3.4 Qualitative Evaluation: Grounded Theory Methodology

Grounded theory methodology was used to analyze the qualitative data. Instead of presuming a theory and then experimenting to determine the validity of the theory, this methodology has the theory "generated from the data," [11, p. 159] allowing the subjects to speak for themselves. The generated theory is then tested against the gathered data to ensure that it is "conceptually dense." [11, p. 169] It is best used when trying to determine process patterns that underlie how subjects and situations relate to each other to produce the observed results. A grounded theorist "can claim predictability for [the theory], in the limited sense that *if* elsewhere approximately similar conditions [are obtained], *then* approximately similar consequences should occur." [11, p. 169]

The overall process is outlined in Creswell and illustrated in Figure 1.[2] Subjects were recruited by direct email with a link to an online survey. They completed the survey anonymously, and the quantitative results were summarized and the answers to all open-ended questions (such as "What do you remember doing in these classes?" and "Why did you take more CS courses than required by your major?") were analyzed using grounded theory methodology.

First, the data was categorized. A constant comparative approach was used to reduce these categories to the key ones. In this approach, "the researcher attempts to 'saturate' the categories – to look for instances that represent the category and to continue looking (and interviewing) until new information obtained does not further provide insight into the categories." [2, pp. 150-151] These categories were then analyzed to identify themes.

Using the categories and themes, a model was then developed to determine their relationship with the central phenomenon of interest: what made the subjects take more computer science courses. This was then visually represented to highlight the causal conditions that influenced the

central phenomenon, the strategies identified, the characteristics of this context, and the consequences of using these strategies in this context.

3.5 Findings

Twenty of the 36 subjects responded (55% response rate). As expected, most were not Computer science or computer-related majors: the majority were math majors (55%) while only 10% (2/20) were CS majors and 25% (5/20) were IS majors. The remaining majors included Financial Economics, Engineering, Education, Sociology, Art, Accounting and Theology. Some students were double majors.

A majority of respondents were CS minors (50%) while 10% (2/20) were Math/CS minors and 5% (1/20) were IS minors. Other minors included Math, Marketing and Management.

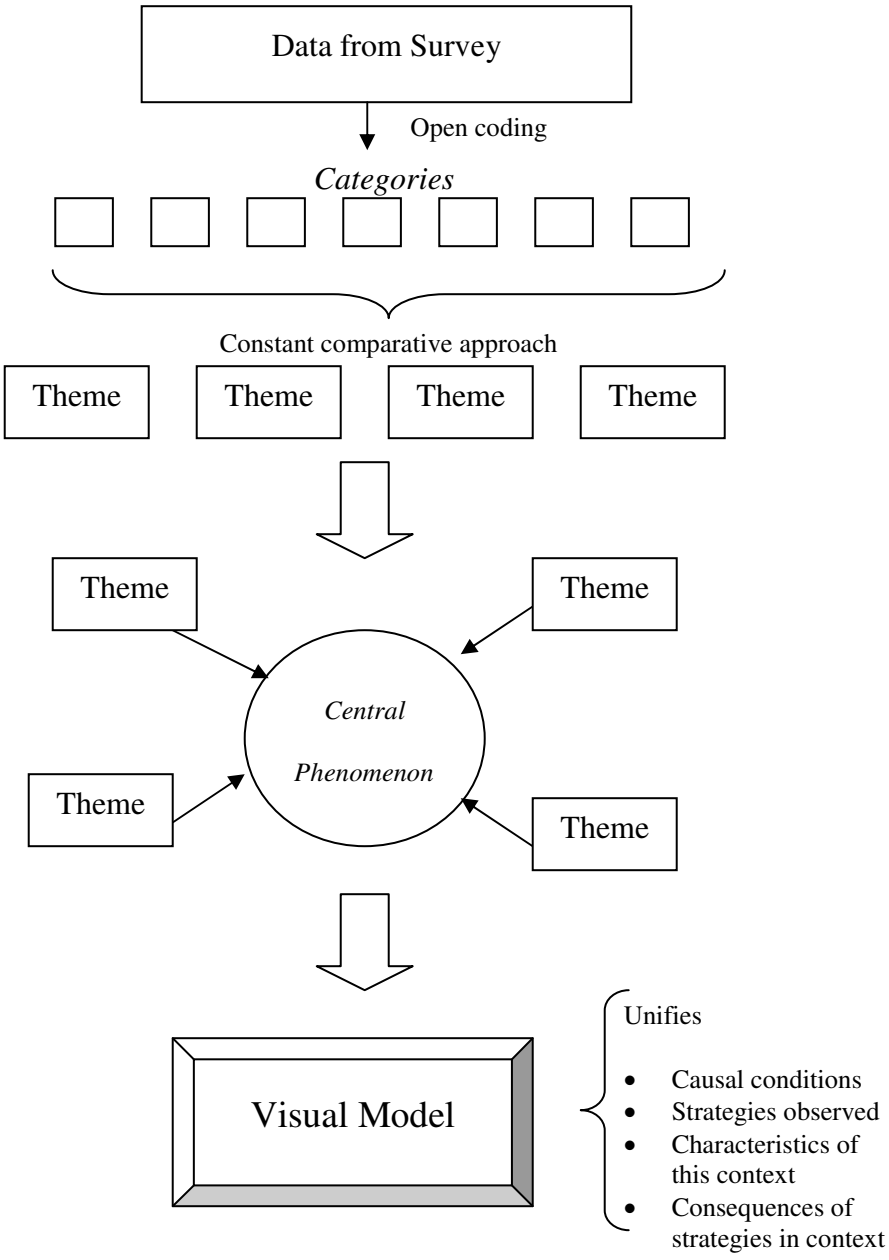

Figure 1: Grounded Theory Methodology

Eighteen subjects (90%) have or will have graduated by the end of 2007. Fifteen (75%) worked or are currently working in a computing-related position such as quality assurance, IT associate, web site specialist, programmer, software engineer, and system, implementation, or application technical support analysts in a variety of small and large companies.

Interestingly, 70% of the subjects reported taking more computer science courses than was required by their majors. 75% indicated that they wished that they had taken more computer science courses. 80% believed that their CS courses or experience helped them in their

employment – either to be interviewed, hired, or promoted, or in the actual duties they were able or asked to perform.

3.5.1 Model

The qualitative data was analyzed for major and minor categories. From those, themes were identified and a visual model was built to describe the qualitative results.

Major Categories

Eight major categories were identified. At least 50% of the respondents gave voice to these categories:

1. ***Meaningful Work*** (20/20 or 100%)

Projects were cited by all participants as extremely valuable for several reasons: 1) Projects gave purpose to the core content. Subjects cited the opportunity to use content and practices in a meaningful project. They also indicated that when specifications allowed them to exercise creativity or to intersect the final product with interests or values, the project was enjoyable and gave them the motivation to complete it, learn more, or work with others. 2) Students developed project management skills. 3) Students refined communication skills.

2. ***Computer Science Concepts*** (19/20 or 95%)

As expected, participants valued the content itself. Learning specific languages and skills were cited as useful, but more important was the fact that the content was generalizable.

3. ***Employment*** (19/20 or 95%)

With each course, participants believed that the computer abilities and skills helped them become more employable. This was cited as a motivation to enroll in and continue CS courses. Some of the graduates indicated that it also proved to be true.

4. ***Relationships*** (18/20 or 90%)

Positive relationships with the professor, teammates, classmates, and tutors helped develop confidence and excitement about the CS courses. These were encouraged by carefully designed projects that encouraged collaboration rather than competition, so that everyone benefited by sharing knowledge and insight. Furthermore, graduates indicated how this helped them in industry, which needs technology experts who can work collaboratively with both technical and non-technical personnel.

5. ***Enjoyment*** (16/20 or 80%)

Whether they were required to take the first course or had a raw interest in the material, the participants indicated that they kept taking computer science courses simply because they enjoyed them.

6. ***Instructional Methods*** (16/20 or 80%)

Participants felt that a variety of teaching and learning activities were important in allowing them to grasp, retain and master the material. Specifically, they felt that more innovative instructional

methods for delivery—such as active learning and model-coach-fade—and time for practice and participation were paramount. However, they also recognized the need for the traditional instructional methods of reference books, tests, and assignments/labs.

7. ***Computer Science Practices*** (15/20 or 75%)

Participants believed that computer science practices developed their critical thinking, problem solving and solution generation skills, which are invaluable in industry.

8. ***No time*** (12/20 or 60%)

A good number of participants indicated that they discontinued taking CS courses because their schedule did not permit it.

Minor Categories

Four minor but noteworthy categories were also identified. At least 35% of the respondents gave voice to these categories:

1. ***Required*** (9/20 or 45%)

A good number of respondents indicated that they took the first computer science course because it was required. They admit that they might not have taken it if it were an elective.

2. ***Career Development*** (8/20 or 40%)

Once students identified themselves as able to do computer science, they appreciated the coaching that would help them present themselves to potential employers as computer professionals. This included resume and portfolio building, individual 360° performance evaluations, and connections with and introductions to professionals in industry.

3. ***Interested in Minor*** (7/20 or 35%)

Some of the participants surveyed expressed an interest in a minor. Five of the seven participants who expressed a desire to have a minor actually completed one. Reasons varied from employment opportunities, to the number of CS options at the College of St. Catherine, to a late-blooming interest.

4. ***Limited Offerings*** (7/20 or 35%)

A few participants complained about the limited offerings at the College. Since there is no major program at the time of the study, only one full-time CS professor was employed, who taught 4 regular courses and advised for independent studies. Students are able to take courses at nearby colleges through the ACTC consortium agreement, but participants felt that more offerings would have been helpful for them and the college.

Themes

From these categories, four major themes were identified:

1. Major requirements and perception of employment opportunities got students into a computer science course.

2. Once in a computer science course, students learned standard CS concepts and practices taught in active ways and focused on a meaningful project.
3. Enjoyment of the material and tasks as well as the value of the technical and non-technical skills learned kept students taking computer science courses.
4. Participants left the computer science program because they achieved their objective (CS major or minor), they ran out of time, or they ran out of course options.

Visual Model

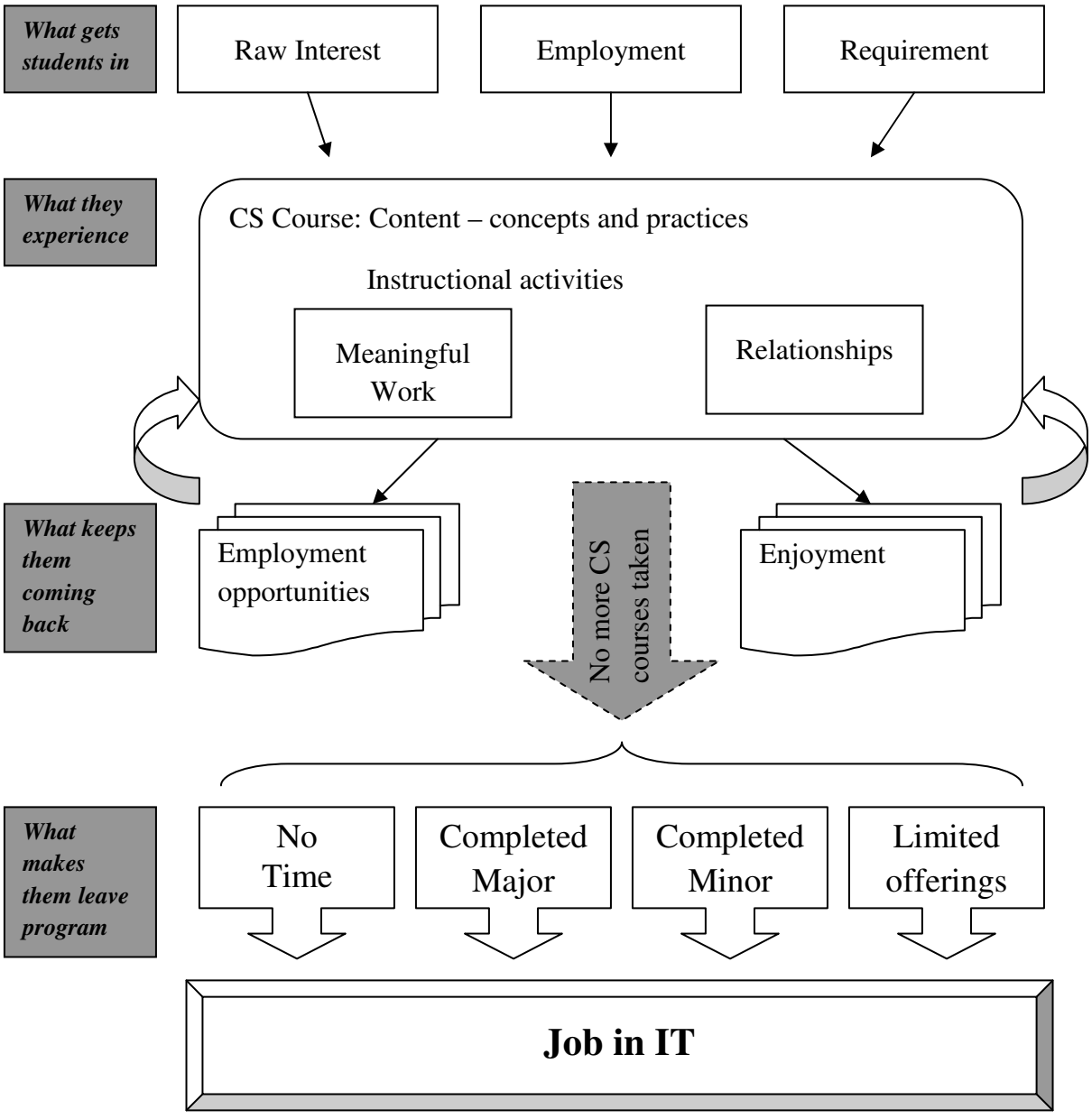

Figure 2: Model of What Motivated Students to Take More CS Courses

The central phenomenon of interest was what caused these women to take at least 3 computer science courses or at least one computer science course more than was required for their major. Figure 2 illustrates how these fit together.

Students get into the first computer science class because they have a raw interest in computers, because they believe that computer science will help them be more marketable or because computer science is required by their major.

Once in the first computer science course, each participant indicated that what they did in the class greatly influenced their desire to continue taking courses. This correlates with the Parallel Curriculum Model concepts: **Core** *content* of the courses – programming constructs, specific languages – were found both interesting and useful to the subjects. Moreover, common ***practices*** such as program design, implementation, debugging, testing, and the ability to think critically were also found to be interesting and generalizable to other problems or jobs. Since these are the two areas that are always covered in introductory CS1 classes, it was encouraging to see that the subjects not only got through but that students enjoyed them.

Two other components to the curriculum were also cited as important. They include activities that helped students see the **connection** of the technical concepts to something *meaningful* to them, such as a project, their own creativity, or simply having fun. They also cited activities that built strong *relationships* with classmates, the instructor, or teammates. Presentations and career development coaching also helped them connect their classroom learning to *employment* opportunities. This *enjoyment* of the courses helped their build their **identity** as someone who could be successful in this field. From here, students may have completed a computer-related computer science major or minor, or may have graduated without such formal credentials. However, most of the participants, despite not having computer science majors, were able to be gainfully employed in computer-related positions. Most indicated that their computer science education helped their marketability.

Reasons participants stopped taking courses included: completing their computer-related major or minor, running out of time due to graduation, or being limited by course offerings on campus.

3.5.2 Strategies

From this analysis, several strategies emerge to increase the voluntary enrollment of students in computer science courses:

1. **Requiring an introductory course**. This seemed extremely important to get students into the class for the first time.
2. **Using a variety of instructional and learning activities.** The standard concepts can and should be included. However, the means by which they are taught should actively involve students and allow them time to practice the material. Practices such as critical thinking, problem solving and solution generation skills should also be explicitly taught and practiced.
3. **Using projects to make standard content meaningful.** If the project is meaningful to the students (e.g., requires them to exercise creativity, or will be used by someone), they are more motivated. Additionally, if homework assignments and concepts in class help

students develop skills that they can directly relate to the project, it gives coursework a purpose by helping students realize why they are learning the material.

4. **Creating a social community**. A community within the class can be created by having activities within class, requiring team projects, and providing lessons on how to handle team-related management issues. Additionally, a larger community can be built by using past students as tutors or teaching assistants within and outside the classroom.
5. **Incorporating career development activities.** Since students are highly motivated by employment opportunities CS has to offer, it is important to help them connect their in-class learning with employment opportunities. This could be as simple as resume and portfolio creation or guest lectures. It could also include more involved activities such as connections with the college's Career Services office, networking with local companies, or 360° evaluations, in which students evaluate themselves, instructors evaluate students, and both discuss the similarities and differences. Such evaluations can also include a time for goal setting and for informal correction of behavioral problems.

3.5.3 Consequences

There were two main consequences to these strategies:

1. **Increased *enjoyment* of computer science**. This provided internal motivations and fostered participants' confidence to take another computer science course. With their firm foundation in the fundamentals, they were technically equipped to learn more if they desired later in their careers.
2. **Increased *employment* opportunities.** Because students could identify the skills and abilities they learned in computer science courses, they were externally motivated to continue developing these by taking more classes. Moreover, securing internships gave them the confidence that they could learn more—in another course or on the job, even if they did not have a computer science major.

4 Conclusions

4.1 Discussion: Did PCM-designed Curriculum Achieve Objectives?

Grounded theory analysis combined with the quantitative results revealed consequences that directly correlated with the desired objectives. Taking each objective individually:

- Has the curriculum improved employment opportunities?
  Yes. One consequence of this curriculum was the increased employment opportunities. Specifically, 75% of the participants worked or are working in an IT-related field. 80% of the participants believed that their CS courses or experiences helped them in their employment. Specific reasons included the CS knowledge they learned, CS-encouraged practices such as critical thinking and problem solving, project management, communication and team skills.

- Has the curriculum fostered the desire to learn more CS?

Yes. Another consequence of this curriculum was increased enjoyment of CS courses. 70% of the participants reported taking more CS courses than their major required, and 75% wished they had taken more. Their main reasons for wanting to take more included the perceived employment opportunities and enjoyment of the courses they did take.

- Do the students have the foundation (academic and confidence) to learn more CS? Yes. Academically, all participants who started a CS or IS major successfully completed the major. Additionally, one participant is currently in graduate school for software engineering. Regarding confidence, participants cited lack of time, course offerings on campus or money as the reasons they stopped taking courses. None indicated they thought the material was too difficult for them.

4.2 Transferability of Process

Five strategies emerged from this study of the curriculum designed with PCM.

1. Requiring an introductory course
2. Using a variety of instructional and learning activities
3. Using projects to make standard content meaningful
4. Creating a social community
5. Incorporating career development activities

When applied to other contexts, some strategies were more successful and valuable than others.

Co-ed computer science course with majors

When dealing with majors and a co-ed environment, the *employment* consequence seems to be the most valued. Although PCM was not originally used in the initial design of this course, most of these strategies (a variety of instructional and learning activities, project-based learning, creation of a social environment, and career development coaching) were implemented with an upper level co-ed graphics course. This course had 5 men and one woman from an nearby ACTC school who were CS majors. They consisted of 50% of the course.

Generally, both men and women greatly appreciated the project-based learning and team-management skills. Most said that it was their most productive project experience because the required phase reviews guided their progress continually, ensuring that they did not wait until the last minute to finish the project, as they normally did when left on their own.

As majors, they were already genuinely interested in the computer science material. However, the focus on employment was a new and welcome perspective. In fact, they were in need of guidance: in an April evaluation meeting, four seniors indicated that they had not yet secured employment or graduate school for after graduation that May.

Engineering for everyone course

In our "engineering for everyone" course, Makin' and Breakin': Engineering in Your World, most of the students are non-majors. It is one option to satisfy their lab requirement. Again, the

population is entirely women, and engineering is not an area most females are interested in or aware of. PCM was used extensively in designing the course, so most of these strategies (a variety of instructional and learning activities, project-based learning, and creation of a social environment) were implemented with a class that consisted primarily of first and second year students, but some juniors and seniors attended as well.

As was expected, the *enjoyment* consequence was the most valuable, with many of the students not only finding the projects challenging but creative, interesting and relevant to the technical information on structures, materials, mechanisms, simple machines, hydraulics, pneumatics, and electricity. They continually indicated that they now saw the world in a different way and expressed confidence in dealing with the engineered products in their world: cars, housing, machinery, and so on.

4.3 Teaching to Teachers: Passing the Tradition On

This curriculum shifted students' attitudes. Often, they perceive school as pretend, assignments are something you do to get a good grade, and good grades get you jobs. By focusing on industry requirements and being transparent about the connection of project assignments to employer expectations, students started to see courses, computer science courses in particular, as useful and important building blocks to get to a career.

The Parallel Curriculum Model helped ensure that students were educated, not trained, so that they had the skills and confidence to understand emerging technology, manage projects, communicate with technical and non-technical personnel, and work effectively in teams. This study showed that this curriculum was successful with liberal arts women in non-CS majors. They can work in IT jobs and leave college with a strong desire to learn computer science.

The College of St. Catherine is now testing how transferrable this curriculum is by using adjuncts and visiting professors as short-term instructors in the CS courses. Guided by this PCM-designed curriculum, we hope to verify that these strategies hold for multiple instructors and can adapt with new technologies over time.

References

1. Belenky, M. F., Clinchy, B. M., Goldberger, N. R., & Tarule, J. M. (1997). *Women's ways of knowing.* NY: HarperCollins Publishers.
2. Creswell, J. W. (1998). *Qualitative inquiry and research design: Choosing among five traditions.* Thousand Oaks, CA: SAGE Publications, Inc.
3. Cuny, J., & Aspray, W. (2000). *Recruitment and retention of women graduate students in computer science and engineering: report of a workshop June 20-21, 2000.* Retrieved December 15, 2007, from Computing Research Association website, http://www.cra.org/reports/r&rwomen.pdf.

4. Dean, C. (2007). Computer science takes steps to bring women to the fold. *New York Times, April 17, 2007.* Retrieved December 14, 2007, from http://www.nytimes.com/2007/04/17/science/17comp.html?emc=eta1.
5. Lee, J. D. (2002). More than ability: Gender and personal relationships influence science and technology involvement. *Sociology of Education, 74*(4). Retrieved February 20, 2007 from ProQuest database.
6. Lethbridge, T. C. (2000). What knowledge is important to a software professional? *IEEE Computer, 16*, 44-50.
7. National Center for Women and Information Technology. (2006). *By the numbers.* Retrieved December 15, 2007 from http://www.ncwit.org/pdf/Stat_sheet_2007.pdf.
8. Ng, Y., & Maxfield, L. (2006, June). *The parallel curriculum model: Understanding engineering educational innovations to optimize student learning.* In Proceedings of the 2006 American Society for Engineering Education Annual Conference and Exposition, Chicago, IL. CD-ROM.
9. Ramsey, N., & McCorduck, P. (2005). *Where are the women in information technology?* Retrieved December 14, 2007, from Anita Borg Institute for Women and Technology website, http://anitaborg.org/files/abi_whereareth ewomen.pdf.
10. Simard, C. (2007). *The business case for gender diversity.* Retrieved December 14, 2007, from Anita Borg Institute for Women and Technology website, http://anitaborg.org/files/businesscasegenderdiversity.pdf.
11. Strauss, A., & Corbin, J. (1998). Grounded theory methodology: an overview. In N. K. Denzin & Y. S. Lincoln (Eds.), *Strategies of qualitative inquiry* (158-183). Thousand Oaks, CA: SAGE Publications, Inc.

Appendix

Major Categories

Eight major categories were identified. At least 50% of the respondents gave voice to these categories:

1. ***Meaningful Work*** (20/20 or 100%)

Projects were cited by all participants as extremely valuable for several reasons: 1) Projects gave purpose to the core content. Subjects cited the opportunity to use content and practices in a meaningful project. They also indicated that when specifications allowed them to exercise creativity or to intersect the final product with interests or values, the project was enjoyable and gave them the motivation to complete it, learn more, or work with others. 2) Students developed project management skills. 3) Students refined communication skills.

- The projects held my interest and helped me to learn the necessary concepts.
- Keep the projects so we can see why we're bothering to learn certain methods, styles and pieces of programming. The four projects over the course of each [CS] programming class made it worth while.
- My CS courses helped me obtain an internship with the UnitedHealth Group Uniprise Technologies Enterprise Architecture team after graduation. The internship required someone of a technical background with good written and oral communication skills. During my CS coursework, we had to complete various projects and present them,

along with a report, so this experience was very beneficial when I was interning. In addition, my CS courses aided me to where I am at right now with Accenture. One of my current job responsibilities entails that we work with other teams and our team members to resolve issues, so again, the team work skills that I built during college proved beneficial.

- I found the 112 project very helpful in putting my knowledge together. Up until that point, we worked on small projects, applying [one] concept at a time to a program. I remember the 112 project being a time where I felt all of this fragmented knowledge started to come together to form a more cohesive understanding.

2. ***Computer Science Concepts*** (19/20 or 95%)

As expected, participants valued the content itself. Learning specific languages and skills were cited as useful, but more important was the fact that the content was generalizable.

- I always felt, throughout my CS studies at the college, that I was building a good basic set of skills and understanding for how computers and programming worked and how to apply them.
- I really felt I was preparing myself, developing my qualifications, as I completed these courses. And now as I start my corporate career (in which I work extensively with computers and program languages)… I see first-hand how a deeper knowledge and background in computer science would be extremely beneficial.
- My testing experience and Java knowledge helped me to gain the skills that I use now. I also believe that my classes helped me to become less intimidated by unfamiliar computer jargon and concepts. This is especially valuable for me because a lot of my job involves learning new skills, and I know that even if I don't understand them right now, I am capable of learning them.
- Furthermore, since I currently work in a technical field, all the programming courses I took have enabled me to pick up new programming languages easily, such as Perl and SQL, because the concepts are similar as are the languages.

3. ***Employment*** (19/20 or 95%)

With each course, participants believed that the computer abilities and skills helped them become more employable. This was cited as a motivation to enroll in and continue CS courses. Some of the graduates indicated that it also proved to be true.

- I am thoroughly satisfied and pleased at my CS experience. All the sweat and hard work put into my projects really paid off. In addition, the various constructive criticisms that took place really helped me in the work force. It has helped make me a stronger worker because I am not afraid of receiving feedback (whether it be good or bad) and taught me how to take all the feedback as a way to improve my work productivity and [become a] company asset.
- I would say that all the projects I had done in CS courses help me a [lot] for my employment. CS courses taught me to deal with real world workplace. I have used a lot of examples I learned from CS courses for interviews and my current employment. For instance, work in group, responsibility, time management, patience…. I am proud to say that I have used CS courses in my current employment.
- I wish I would have taken more because it would really have benefited me with my current internship which is in the IS department of a Financial firm.

- I have listed the programming languages that I've learned/become familiar with throughout my CS courses on my resume, which have been pointed out in interviews and (I am certain) was a contributing factor in my being hired. My technical writing, communication, and presentation skills, as well as my ability to work well within team dynamics, are some of the reasons I continue to move up the corporate ladder. All these personal qualifications were first encouraged/strengthened through my CS experiences at St. Kate's. Probably the most striking example of how my CS courses/experience has assisted my employment was from a collaborative project in my CSCI 208 class. I had the opportunity to work on a team, which met regularly with a group of advanced marketing students, to build a website. The marketing students would communicate the desired outcome, while my team and I would take these requirements and translate them into manageable tasks, ultimately producing a quality product. This is EXACTLY the type of work I did for Student Technical Services at 3M. I would meet with business contacts to discuss how they would like their sites migrated/built on 3M's new platform, and then made sure that a timely, quality product was handed-over.

4. ***Relationships*** (18/20 or 90%)

Positive relationships with the professor, teammates, classmates, and tutors helped develop confidence and excitement about the CS courses. These were encouraged by carefully designed projects that encouraged collaboration rather than competition, so that everyone benefited by sharing knowledge and insight. Furthermore, graduates indicated how this helped them in industry, which needs technology experts who can work collaboratively with both technical and non-technical personnel.

- I think it is important that the atmosphere is cooperative rather than competitive. Students should be able to help each other work on their projects and answer each other's questions.
- [I would recommend taking] small classes. ... It is easier to get personal attention and get to know your classmates. I would also encourage group work, in class, as well as out of class. I did that with my friends while at St. Kate's. We learned faster, as we explained things to each other, and we got our work done faster, and also had fun doing it.
- I found [the courses] very fun and challenging. I liked the people I was with and feel that I developed very valuable skills, both for my own interests and marketability.
- CS courses helped me because I am now the technology go to person at my small company. It was an added benefit to my already diverse skills. The ability to speak to a non-technical person while getting the technical points across in a succinct manner is also valuable
- I think the team project in the upper level CS class was valuable, as well, in that it allowed us to apply concepts we'd learned into a team environment, which added the complexity of how to break down this type of project among teams, how to form a project plan/schedule that delegates responsibilities to different people, and how to work together with others on a team to make the whole thing come together.

5. ***Enjoyment*** (16/20 or 80%)

Whether they were required to take the first course or had a raw interest in the material, the participants indicated that they kept taking computer science courses simply because they enjoyed them.

- It was fun – I really started liking computer science, at least the programming and hardware aspects of it, and decided to learn more about it and study as much as [college initials] offers.
- [P]rojects are fun. It's fun to be creative. I think that's why I enjoyed all the projects I had done in classes.
- I was interested by the approach to the material which made it seem much more relevant and interesting than I had previously thought.
- It's very satisfying to get the results of computer programs and put the concepts into concrete terms, as well as having the final projects and portfolio to show at the end of the courses.

6. ***Instructional Methods*** (16/20 or 80%)

Participants felt that a variety of teaching and learning activities were important in allowing them to grasp, retain and master the material. Specifically, they felt that more innovative instructional methods for delivery—such as active learning and model-coach-fade—and time for practice and participation were paramount. However, they also recognized the need for the traditional instructional methods of reference books, tests, and assignments/labs.

- I learned well from the teaching methods used. As I remember, these were explanations/examples, demonstration (in which the class would be encouraged to participate in either acting out a concept and/or suggesting input/direction as the solution(s) were revealed so it was more active learning, not passive), then trial/error on our own time.
- I liked the style of teaching that was used- try and then correct. Students were given problems to try out by the next class and then come to class with questions/problems as to why it didn't work or what worked about it and what connections were made in the process of "doing it yourself." The constant repetition of concepts are what make them stick in my brain.
- I think it would be beneficial to continue to use a variety of instruction methods for CS courses. Lecture and in-class activities are very helpful, but I believe a book is also good to have as it serves as a reference and possibly as a resource for another explanation or view of the material being covered. Assignments covering a specific portion of course material are useful for gaining immediate feedback, but longer term projects are also beneficial as they can serve to help students pull all of the pieces together.
- It was helpful to have the class broken into different periods of work, where we would shift focus to different concepts. Even though these were often my longest classes, they didn't feel that way when we were able to change gears this way. I thought moodle and the self-tests worked well.

7. ***Computer Science Practices*** (15/20 or 75%)

Participants believed that computer science practices developed their critical thinking, problem solving and solution generation skills, which are invaluable in industry.

- [I remember] Grasping the basic concepts of programming and creating a strong foundation for self learning – creating flow charts, writing algorithms, analyzing algorithms, programming, debugging, and technical writing.
- [I] Learned how to design out a plan and carry it out and manage time by splitting up the tasks. … Learned skills in debugging and dealing with issues on a project. … We continued learning Java with animation, as well as more complicated design concepts and how to implement them, such as linked lists, stacks, and queues.
- [The courses] helped me do problem solving. I [became] really good at it. The skills I learned from the earlier Computer Science course (HTML, JavaScript) were helpful. The process of critical thinking and problem solving was really helpful to my job. It doesn't matter what application I used to do my job or what languages I used, it always came in handy.
- Perhaps, more importantly, I learned how to think about concepts and ideas in a way that allows me to not just know the syntax needed in order to achieve a specific result in a particular language, but also to structure, plan, and organize a programming project. In each class there were a number of projects that, compared to projects in other classes, were structured very much like those you might encounter in the "real world."

8. ***No time*** (12/20 or 60%)

A good number of participants indicated that they discontinued taking CS courses because their schedule did not permit it.

- First, there is only so much time, had I additional time to spend at St. Kate's I probably would have taken more CS classes.
- I completed my degree within a timeline that only allowed required courses
- I was also, unfortunately, introduced to computer science later in my academic efforts…so I didn't have time time/money to fit more non-required courses into my credit load.
- I didn't realize how much I enjoyed computer science until my second to last year at St. Kate's. I took as many computer science courses as I could before graduating.

Minor Categories

Four minor but noteworthy categories were also identified. At least 35% of the respondents gave voice to these categories:

1. ***Required*** (9/20 or 45%)

A good number of respondents indicated that they took the first computer science course because it was required. They admit that they might not have taken it if it were an elective.

- I had been interested in taking computer science courses in high school, but because of scheduling conflicts was not able to do so. ... [C]ertain CS courses were required for a mathematics major at St. Kate's, and because I knew I was majoring in math, I began taking the required courses right away in my first semester.
- I took more courses than was required because I really enjoyed the one that I had to take.
- I wouldn't have thought to take a [CS] class unless [it] was required for me, and if I had taken it earlier I might have been able to take even more classes.

2. ***Career Development*** (8/20 or 40%)

Once students identified themselves as able to do computer science, they appreciated the coaching that would help them present themselves to potential employers as computer professionals. This included resume and portfolio building, individual 360° performance evaluations, and connections with and introductions to professionals in industry.

- I would suggest keeping the professional development section. It really helped when I applied for both of my internships these past two years. It was nice to be able to have a portfolio already developed and critiqued, so I wouldn't have to create one on my own without any direction.
- I was also able to display my skills in project management and time management (working with deadlines) through the portfolio we were asked to create during these classes.
- The regular self and team evaluations with my professor were helpful to me in working on my strengths and weaknesses. The projects were a great way to use the concepts from class in a fun way, to have a tangible result of work, and to learn professional skills and group work. The presentations, reports, portfolios and application to job skills was very valuable to me. It was very nice having the conceptual knowledge as well as the practical kind, and the blend we got through our classes was interesting and helpful. I am obviously grateful for the networking opportunities and professional aspect, because these are skills that are not as straightforward to me, but help reinforce the others.
- I would encourage that students are exposed/coached about how taking CS classes can help them. When I was in college, I thought that by taking a CS major, I would have no choice than becoming a programmer. For students for whom programming is not an appealing career, it is good to know how they can use the CS classes in other careers.

3. ***Interested in Minor*** (7/20 or 35%)

Some of the participants surveyed expressed an interest in a minor. Five of the seven participants who expressed a desire to have a minor actually completed one. Reasons varied from employment opportunities, to the number of CS options at the College of St. Catherine, to a late-blooming interest.

- Math as a major is a very broad field. I knew that in order to market myself in the workforce, I needed a minor that would complement my major, hence my decision to get a minor in CS.
- I was required to take 111 and 112 for the math major, and found that I liked it well enough to continue and get a minor.

4. ***Limited Offerings*** (7/20 or 35%)

A few participants complained about the limited offerings at the College. Since there is no major program at the time of the study, only one full-time CS professor was employed, who taught 4 regular courses and advised for independent studies. Students are able to take courses at nearby colleges through the ACTC consortium agreement, but participants felt that more offerings would have been helpful for them and the college.

- Although the number and type of CS courses offered grew significantly while I was there, compared to many other colleges and universities, it was still a limited offering.

- I believe the program should be broadened… more classes should be offered in more ways (ex. Weekend College), and students should be able to major in computer science at St. Kate's
- I'd rather stay at my home college to take more classes. I love the ACTC colleges, but it's so much easier not to have to go between campuses to study the courses I want.
- I had already taken all the computer science courses that were offered at St. Kate's, fulfilling my minor, and preferred not to pursue further studies off-campus.