Online Journal of Applied Knowledge Management

A Publication of the International Institute for Applied Knowledge Management

Volume 6, Issue 1, 2018

A welcomed reversal in computer science enrollments: Analysis of contributing factors and recommendations to sustain the growth

David T. Smith, Indiana University of Pennsylvania, PA, USA, david.smith@iup.edu

Azad I. Ali, Indiana University of Pennsylvania, PA, USA, azad.ali@iup.edu

Abstract

Periodic assessment of enrollment in programs is helpful in different ways. It reveals patterns that may not be immediately clear and it identifies contributing factors that provide insight into sustaining growth. For Information Technology (IT) programs, this kind of assessment takes another dimension - the rapid changes of IT may dictate on faculty members to continuously study market demand and to reshape their program's curriculum in response to the demand. This study assesses the recent trends of enrollment in Computer Science programs. It presents enrollment trends in the years since the turn of the century, analyzes the factors that contributed to the cycles occurring during these years, and provides recommendations for sustaining the most recent trend. This paper sheds light on a program in Computer Science at a university in Western Pennsylvania. It shows how this department continued assessing the trends regularly and how they are responding to the most recent enrollment trends in their program. The paper concludes with a summary and recommendations.

Keywords: Enrollment reversal in computer science programs, enrollment trends in computer science programs, attrition in computer science programs, and graduation rates in computer science programs.

Introduction

The enrollment in Computer Science (CS) programs has lately shown a steady and significant increase (Fisher, 2016; Guzdial, 2017; Zweben, 2011) following a protracted dip at the start of the millennium (Benokraitis, Bizot, Brown, & Martens, 2009; Zweben, 2009). While CS faculty members are happy to embrace the new trend, identifying the factors that contributed to this reversal is not totally agreed. This study conducted an analysis of several different factors and offers recommendations to sustain the upward trend.

This paper is a follow-up on a prior study by Ali and Shubra (2010) at the time of the dip. Ali and Shubra (2010) analyzed several factors that led to the downturn and suggested actions to reverse the slide of enrollment. Eight years later, the dip has been reversed. Thus, this study assesses that specific reversal. The results presented herein may be of interest to faculty members in the CS, Information Technology (IT), and related computing disciplines.

In this study, we assess the current enrollment trends in CS programs, we analyze the factors that led to the changes, as well as make our recommendations and remarks at the end. We first present the enrollment data and trends over the last two decades. We then analyze the corrective actions that effected the change. Next, we discuss market changes that helped in the latest enrollment trend. This will be followed by an explanation of a specific program and the efforts to address enrollment. Lastly, this paper presents a summary and recommendations.

Enrollment Trends – Charts and Numbers

In this section, we present enrollment trends in CS programs in two formats: first we present them in charts and graphs to give an overall view of enrollment. Second, we present them in numbers and tables to make comparison easier and to show different categories that are listed under.

Enrollment Trends - Charts to Cheer On

In contrast to the earlier study of Ali and Shubra (2010), enrollment data and graduation rates all point upward. A reputable source for data is the Computer Research Association (CRA) (2017), which issues annual reports on enrollment, graduation and other information for CS programs and other related disciplines (like Computer Engineering - CE). The reports, termed "Taulbee Reports" (CRA, 2017), are based on annual surveys and data collected from more than 200 programs in the United States (U.S.) and Canada. The latest Taulbee Report assessed data from 1995 through 2016 and projections for year 2017 (CRA, 2017). A few charts from the Taulbee Report are shown below (CRA, 2017). Figure 1 shows the Bachelor of Science (B.Sc.) degrees awarded in CS and CE majors between the years 1995 and 2016, with projection for 2017. The chart shows a peak in graduation numbers around year 2005 followed by a significant dip in graduation between 2005 and 2009, and then an upward trend ever since. Although, the chart shows the combined enrollment in two programs (CS & CE), the CS programs are identified as the bulk of those surveyed.

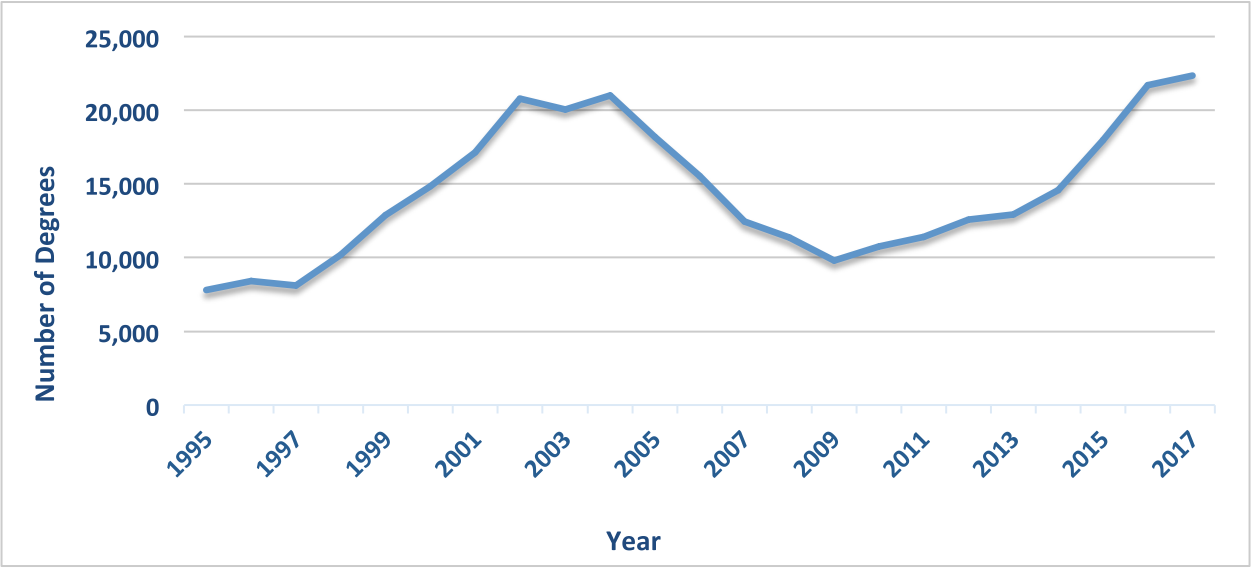


Figure 1: B.Sc. in CS and CE Production from the CRA Taulbee Survey (CRA, 2017)

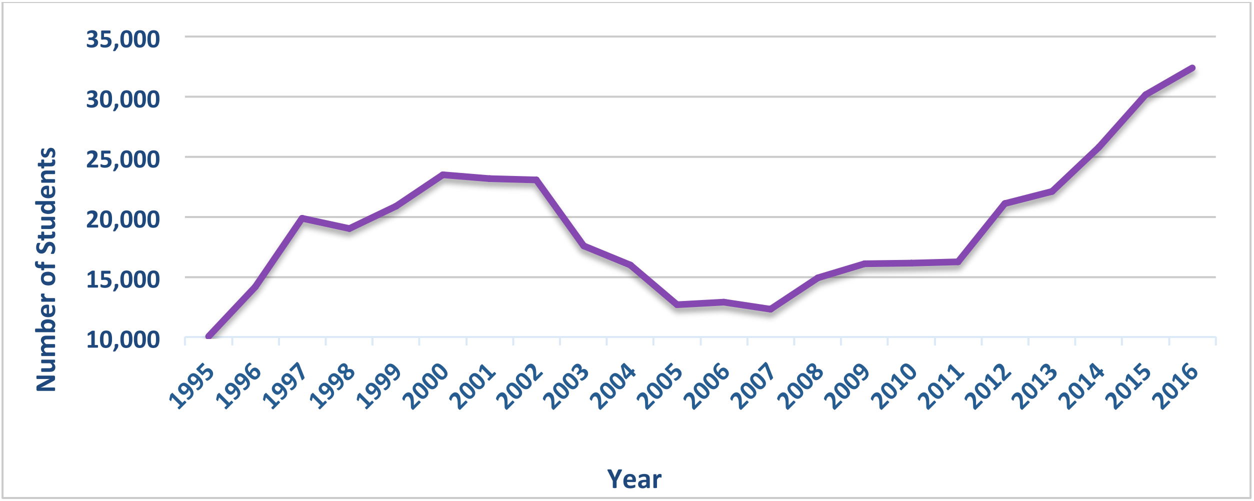


Figure 2: Newly Declared Undergraduate Majors: CS, CE, and IT (Beginning in 2008) from the Taulbee Survey (CRA, 2017)

The upward trend is not limited to degrees produced, as enrollment numbers also show a significant upward trend. Figure 2 recreated from the Taulbee report (CRA, 2017) shows the trend in newly admitted students in IT related programs (again CS programs are identified as the bulk of those surveyed). Figure 2 shows the dip down in admission preceded the slide down in graduation number by four years, as anticipated since most programs take four years to complete a degree. The slide down in admissions started in about 2002. Figure 3 shows a chart from Taulbee report (CRA, 2017) that contrasts the numbers of new declared majors with the number of students continuing their CS major. The number of continuing majors surpasses those admitted indicating a strong retention rate. It shows that majority of students who are admitted into the CS programs continue studying in their programs of study.

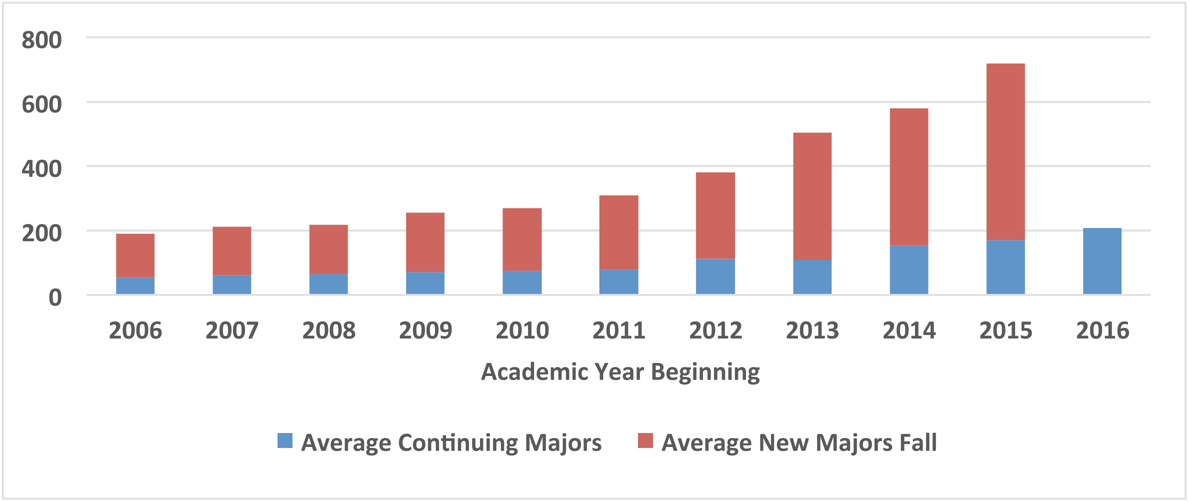


Figure 3: Average New and Continuing CS Majors per Academic Unit - Retention in CS Programs as Reported the Taulbee Survey (CRA, 2017)

Enrollment Trends – Numbers To Brag About

While the preceding charts (Figures 1 to 3) are helpful, a tabular listing of graduation and enrollment numbers in CS programs versus overall university enrollments reveal a brighter picture. To produce such listing, our search for numbers in enrollment led us to the National Center for Education Statistics (2018), which collects, analyzes, and makes available data related to education in the U.S. and other nations. Data for Table 1 were extracted from their databases. The table compares number of bachelor degrees granted in CS versus total of all other degrees. The more obvious number that shows this trend of comparison in the 3rd and 5th columns that show the percent of change. While both raw data shown in 2nd and 4th columns show a glimpse of the difference, the percent of change draws a clearer picture. For example in 2006 there was an increase of 2.7% in overall degrees granted over the previous year, but the graduation numbers in CS degrees decreased by 14.5%. This was significantly different in 2014. Here CS graduations show a substantial increase of 16.7% while overall graduation was only 3.7%.

A similar comparison can be made in the number of enrollment in CS programs versus enrollment in all programs combined, as shown in Table 2. Data for the 2nd column were obtained from The National Center for Educational Statistics (2018), however, it did not include a breakout for CS programs. Thus, to provide a comparison we reviewed all the Taulbee reports between 2005 and 2016 and extracted the number of CS enrollment for each year and placed them in the 4th column. The percentages of change (columns 3 & 5) in this table are more significant for this study. While in 2014 enrollment in overall programs dipped by 1.1%, enrollment in CS programs increase by 24.7%.

Table 1: Comparison Data Between Degrees Granted in CS and Total of all Degrees in the US 2005-2016

Year	Overall Degree Granted	% change	CS Degree granted	% change
2005	1,514,426	-	52,293	-
2006	1,554,564	2.7%	44,686	-14.5%
2007	1,593,009	2.5%	39,292	-12.1%
2008	1,627,338	2.2%	36,157	-8.0%
2009	1,665,723	2.4%	35,959	-0.5%
2010	1,716,075	3.0%	37,470	4.2%
2011	1,783,661	3.9%	41,021	9.5%
2012	1,857,579	4.1%	44,929	9.5%
2013	1,908,091	2.7%	48,713	8.4%
2014	1,978,589	3.7%	56,826	16.7%
2015	2,013,354	1.8%	61,825	8.8%
2016	2,040,669	1.4%	67,316	8.9%

The picture appears clearer; while the numbers and charts support a reversal of trends of enrollment in CS programs along with the fact that enrollment in these programs is steadily increasing. The key question that may rise from such observation is: what are the factors that led to this reversal in enrollment and graduation? In our opinion, this reversal of trend is attributed to two key of factors: (1) factors related to changes in the teaching of programming courses, and (2) factors related to the market changes. We discuss both factors in the next two sections.

Table 2: Comparison Data Between CS Programs Enrollment and Enrollment in All Programs 2005-2016

Year	Overall Enrollment	% change	CS Enrollment	% change
2005	17,487,475	-	56,450	-
2006	17,758,870	1.6%	47,961	-15.0%
2007	18,248,128	2.8%	37,337	-22.2%
2008	19,102,814	4.7%	39,004	4.5%
2009	20,313,594	6.3%	40,147	2.9%
2010	21,019,438	3.5%	43,420	8.2%
2011	21,010,590	0.0%	43,759	0.8%
2012	20,644,478	-1.7%	49,456	13.0%
2013	20,376,677	-1.3%	58,075	17.4%
2014	20,207,369	-0.8%	72,447	24.7%
2015	19,977,270	-1.1%	90,120	24.4%
2016	20,185,000	1.0%	105,148	16.7%

Changes in Programming Courses

The teaching of programming constitutes a significant portion of CS programs. A typical CS department teaches more than one programming course; an introductory programming course, a core sequence providing an intermediate understanding and skill in a selected programming language, as well as upper level electives in specialized areas such as systems programming, web programing, mobile programming, and artificial intelligence. These upper level electives may involve other programming languages. Different studies conducted for this purpose identified programming as a major contributing factor to enrollment drop in CS programs. The same studies suggest that changes need to be made to the programming methodology, the selection of first programming language, and careful selection of additional languages from a seemly increasing number of languages (Baldwin & Kuljis, 2001; Sloan & Troy, 2006). This section analyzes the factors related to programming languages. It first explains how previous practices led to the first decline in enrollment and how corrective actions led to the reversal.

Selection of Programming Methodology

In the previous stages, programming used to be taught without much planning and without giving much attention to how interesting and/or meaningful the example programs given to the students. This was exemplified by the classical "Hello World" example in the first programming course (Westfull, 2001). In this example, students are given a task to write a program to produce a text output that writes "Hello World" on the screen. Another example typically given to the students is about converting Fahrenheit to Centigrade temperatures. These and similar examples take some time and effort to produce programs of little perceived function and hence led to dissatisfaction among students. These students were often ridiculed for spending this effort to produce simple output. In other words, there were minimal considerations for making the programs more interesting to the students.

CS faculty members have recognized this problem and there is a trend that calls for more meaningful examples, along with a methodological approach to selecting examples of programming courses. Leutenegger and Edginton (2007) called for a “Games First” approach to teaching programming. They suggested introducing examples that are “visual” and shows movements of objects on the screen which captures the attention to make it more interesting to the students. Adams (2008) on the other hand, advocated giving visual programs as example to capture the attention of students. Machuca and Solarte (2016) encouraged the selection of examples that improve the performance of students in general and shifts their focus on more meaningful examples.

Parker, Ottaway, and Chao (2006) suggested selecting a methodology first before selecting a language. They discussed four methodologies to ponder on before selecting the programing language: imperative programming, object oriented programming, functional programming and logic programming. Along these lines is the debate between “objects first” vs. “late objects” (Ehlert & Schulte 2010; Uysal, 2012). Some may argue that object oriented first is best in that a learner can readily relate to objects in the real world. Nonetheless, a case can be made for each approach. As such textbook authors and publishers may offer two versions of their text. For example Pearson publish two versions of “Java How to Program”, one for “Early Objects” and the other for “Late Objects” (Deitel & Deitel 2017).

Selection of First Programming Language

> Like many things in life, first impression matters, and programming is no exception. The importance of a properly constructed first course in programming cannot be overstated. Such a course leaves students with good programming habits, the ability to learn on their own, and a favorable impression of programming as a profession. A poor experience may result in a “just get by” attitude, bad programming habits, and could lead to a change in majors (Pendergast, 2006, p. 491).

It is not new and well established among educators in the computer field that learning to program is a difficult task for the majority of students (Kelleher & Pausch, 2005). Different factors attributed to this difficulty including rigid syntax, unfamiliar structure, and length of time spent to produce a simple output (Baldwin & Kujilas, 2001; Sloan & Troy, 2008). The bulk of this difficulty shows in the first programming course and the selection of a programming language plays a significant role in addressing this difficulty.

In the distant past, selecting a programming language in the first programming course was rarely made based on methodological selection or studying the suitability for teaching. Instead, selection of a programming language was mostly based on the prevalent language of the time. For example, earlier in the teaching of programming, FORTRAN or COBOL was used in the introductory courses. Even in recent past C or Java has been used. However, these programming languages have proven that they are not educational language (Pendergest, 2006). In other words, is it argued in literature that these languages contribute to the difficulty experienced by the students in learning how to program.

The most recent approaches to selecting the first programming language attempt to address this issue differently. Such new approach considers different factors that make it easier for students to learn the language and also contributes to prepare them for the market (Xinogalos, 2016). The factors that are considered in these selections include market demand on skills of the programming language, the availability of libraries and resources for the language, the simplicity of learning, the availability of textbooks, and other considerations (Parker, Ottaway, & Chao 2006). This kind of selection that considers multiple factors was found to contribute to enhancing students' performance in first programming courses (Machuca & Solarte, 2016).

Decline of "Teaching Programming Languages"

Efforts were put forward to establish what is termed as "teaching languages". Early examples of such languages were BASIC and Pascal. In the 1980s, Pascal was often used in introductory CS programming courses. These languages also experienced a period of acceptance in industry as the personal computer revolution occurred. However, as use of the personal computers grew and evolved, these languages faded, while being replace with new languages such as Java and C#. With these, new complexities needed to meet application requirements resulted in a higher learning curve, especially for beginners. To address this issue, new brand set of languages came that is represented by the Alice programming language (Adams, 2006), Scratch programming language (Dann, Cooper, & Pausch, 2006), and the like. These fully fledged educational languages worked well in learning programming concepts. However, there is one serious problem – these programming languages don't appear to be used in the industry and, therefore, of limited value at the time students seek employment.

An Industry Adopted Language that is Easy to Learn

Fortunately, a programming language has emerged that is both suitable for beginners (Shein, 2015) and has gained significant acceptance in the industry (Parker & Davey, 2017), this being "Python". Python is considered to be easy to learn and add value to a student's resume. Being easy to learn, an increasing number of schools have adopted Python over other languages.

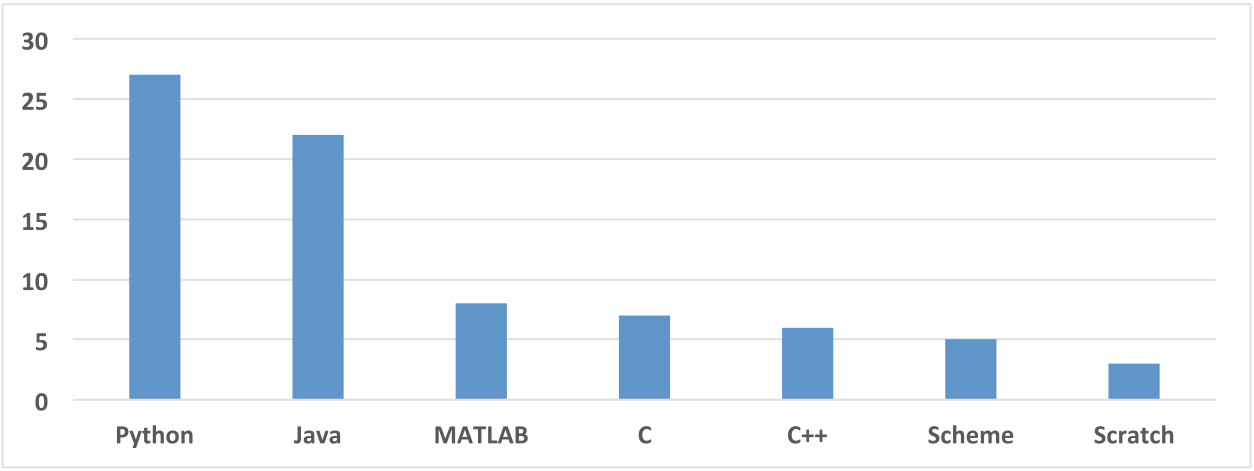


Figure 4: Number of US CS Departments (out of 39) that Use Each Language to Teach Introductory Programming Courses (Guo, 2014)

A survey of 39 CS departments conducted by Guo (2014) showed that in 2014, more than 25% were using Python as the language in the first programing courses, followed by just over 20% using Java (See Figure 4).

Market Changes

The market has shown many new developments that led to the reversal of trends in the job market. This, in turn, has led to increase in CS enrollments. We list a few factors that led to this new trend.

Outsourcing Turns to Insourcing

> Outsourcing, a term to describe the transfer of U.S. jobs to developing countries, emerged significantly over the last decade or so. Its original objective was to set up a national presence in other countries such as China, Brazil, India, and others with the potential for high revenue growth for goods and services. Today, more than 50% of revenues in many corporations come from international markets. The availability of low wage Chinese workers was another factor. What a winning strategy: big incremental global revenues, unlimited availability of resources, significant labor cost reductions, fewer government regulations, lower taxes and huge profits – right? (Burton, 2016, p. 34)

In short, outsourcing refers to shipping jobs abroad. Putting it a political lingo, it may mean 'taking American jobs and shipping them to other countries'. At the turn of the 21th century, many companies began shipping jobs to India, China, and the like to lower cost. Jobs like programming, technology support, and technology analysts took the brunt of this outsourcing (li & Shubra, 2010; Rottman, 2006; Warner & Hefetz, 2012). This resulted in a perceived lack of opportunities for students considering an IT career, thus, led to lower enrollments in CS and IT programs.

The relationship between outsourcing of IT jobs to other counties and enrollment in CS programs may not seem direct or clearly evident. But, understanding the job cycle may reveal an impact of the outsourcing on enrollment, though indirect it has a substantial impact on enrollment. Figure 5 depicts our perspective on this relationship.

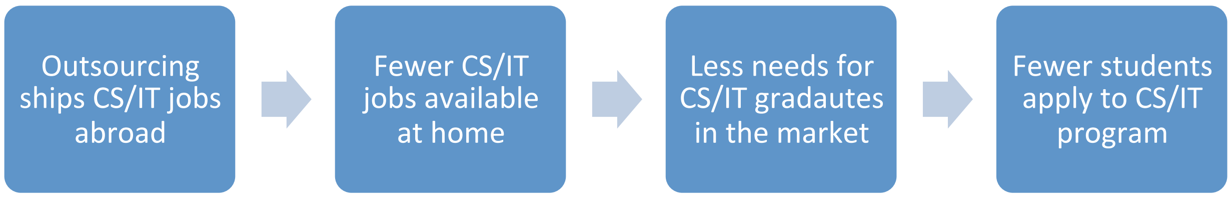

Outsourcing ships CS/IT jobs abroad
Fewer CS/IT jobs available at home
Less needs for CS/IT gradautes in the market
Fewer students apply to CS/IT program

Figure 5: Outsourcing and Effect on CS/IT Programs Enrollment

For different reasons (including economic, political, geographical, & others), outsourcing in not a major factor in the recent employment market and is no longer talked about in the media and the political arenas. Instead, we read more often about another trend called "insourcing". Burton (2013) explained about insourcing as:

> The reversal of outsourcing, transfers jobs from a domestic or foreign contractor or supplier to the internal operations of a business. Insourcing can include bringing a third-party contractor or consultant to work inside a company's facility to provide talent for missing core competencies or resource constraints. The decision to insource is made to protect proprietary technology and intellectual property, maintain control of critical production or competencies and minimize risk in the global supply chain. (p. 35)

In a larger point of view, insourcing means keeping jobs in the US instead of a shipping them abroad. Among the jobs that were insourced include many IT jobs. This, in turn, has a positive effect on applications (& enrollment) in the CS/IT programs. Figure 6 depicts our perspective on this indirect effect of insourcing on applicants to IT.

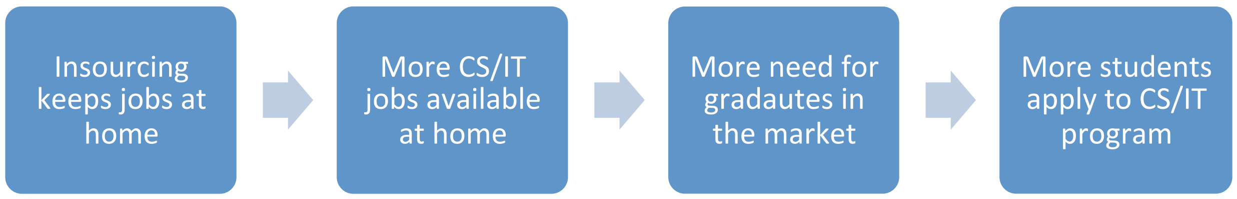

Insourcing keeps jobs at home
More CS/IT jobs available at home
More need for gradautes in the market
More students apply to CS/IT program

Figure 6: Insourcing and Effect on CS/IT Programs Enrollment

Changes in the Computer Technology Market

> The rate and expansion of the IT industry has resulted in an abundance of job titles. The causes are multi-faceted, stemming from a technology explosion, coupled with the introduction of new programming languages, tools, software development paradigms, work practices and job requirements. There may also be, however, underlying factors, particularly when treating IT as a profession. (Donohue & Power, 2012, p. 6)

Many attributed the previous increase of enrollment in IT programs to the dot com boom. They described it as increase in web platforms that led to increase jobs, and then to increase demand on programs that graduates students that teach these talents. A similar (yet sharper) increase in technology is being experienced nowadays. This increase in number of technology platforms has a considerable impact on enrollment in IT programs. This relationship can be explained by the theory of "scarcity of talents" (Bhatt, Emdad, Roberts, & Grover, 2010). In accordance to this theory, when a new technology emerges to the market, the industry shifts focus on the emerging technology. Thus, more companies switch to the new technology and they will be in need to hire new people who are trained on the new technology. In other words, scarcity of supply of talents will be created and more demand of such talents will be created.

Following this theory, one cannot help it but to look at the current market changes and get impressed by the numerous and substantial recent changes in the technology market. This includes an increase: in the use of mobile phone and other portable devices platforms (Glassman & Shen, 2014), in the use of cloud computing (Burns, 2012), in the number of cybersecurity threats (Carlton & Levy, 2017; Solander, Forman, & Glasser, 2016), and in the use of open source programs (Ali & Smith, 2014). Summing all together, we can show this contrast of the increase in technologies on enrollment in CS/IT programs in Figure 7.

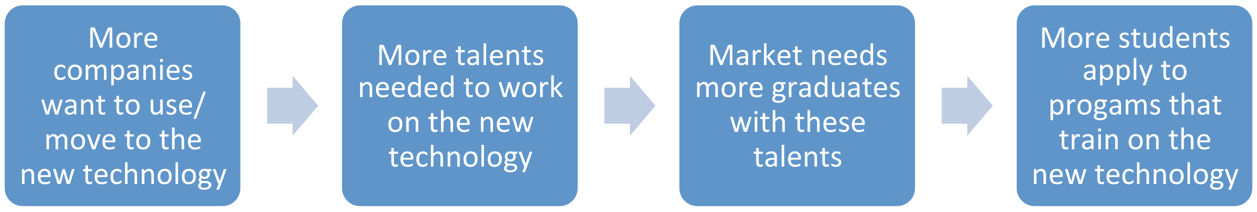

More companies want to use/ move to the new technology
More talents needed to work on the new technology
Market needs more graduates with these talents
More students apply to progams that train on the new technology

Figure 7: Increase of Technologies and CS/IT Enrollment

Abundance of Resources to Learn Programming

Today, with the proliferation of resources on the web, students nowadays can look at tutorials, tools, help videos, technology articles, help forums, developer communities, and the like that were not easily available during previous times. Videos and visual displays of programs or other resources help the students grasp concepts that were difficult to digest in prior years. Here is a small sample of online sites available for programming topics:

- YouTube.com - how to videos of virtually any topic including programing
- CodeAcademy.com, W3Schools.com and Tutorialspoint.com – wealth of tutorials on technical topics including programing
- StackOverflow.com – developer community with answers to many technical questions
- Gnu.org, Eclipse.org and Apache.org – open source languages and tools, many of which include documentation and getting started manuals
- Oracle.com – Java together with documentation, tools, tutorials, and community forum
- Python.org – Python, Python documentation, and community forum
- Wiki.Python.org – Links to download Python, tutorials, documentation, etc.

One particular development that is worth discussing is the Open Software Source Movement (OSSM) allowing many to download programming language compilers, editors, and integrated development environment (IDEs), all for free. The origin of open source software dates back to 1991 when Linux was introduced, and is considered the first tangible achievement of OSSM (Asiri, 2006). Since that time, the list of open source software has grown significantly and now covers a much wider range of applications (Dalziel, 2003).

Increase Number of Programming Languages and Libraries

Kaplan (2010) suggested that there exist more 2000 programming languages that were used in the year 2000. This number is liable to increase substantially as time progresses. With the proliferation of technology came an increasing number of programming languages. Additional factors introduced as well that deal with the introduction of new methodologies, systems, classifications, and many others. This also created many other libraries that can be shared across programming languages. If a programming language is weak in a certain application (like a graphical user interface for a business application), a library can be incorporated to handle and overcome the weakness. These libraries effectively form variations within the programming languages. A case in point is JavaScript programming language. JavaScript is the dominant programming language for client side processing in a web browser and it behooves faculty to include it somewhere in the curriculum. However, JavaScript imposes awkward syntax and lacks a structured framework. Addressing these deficiencies are a host of JavaScript libraries such as JQuery, Node.js, and Angular. The challenge for CS departments is to determine, which of these should accompany JavaScript in the curriculum.

In terms of number of programming languages, this number has grown so much that it is hard to keep an accurate count of the languages. Although a limited number of programming languages (like Java, C++, Visual Basic, & others) remained widely used in academia, a mentionable number of languages (like Scheme & Swift) are not often taught by college programs. As the number of programming languages increased, so too did the number of programming paradigms. The old list, which was basically procedural and structured, now included event driven, declarative, object oriented, functional, and many more related to the web. As a result, academia is faced with the challenge of selecting a set of languages and paradigms that can fit into a program and that provides the best benefit to its target population (Welton, 2005).

Keeping a Pulse on Market Direction

> The rate and expansion of the IT industry has resulted in an abundance of job titles. The causes are multi-faceted, stemming from a technology explosion, coupled with the introduction of new programming languages, tools, software development paradigms, work practices and job requirements. There may also be, however, underlying factors, particularly when treating IT as a profession. (Donohue & Power, 2012, p. 6)

Although recognizing patterns in job demand is important to all professions, identifying and following job trends takes special importance for IT profession. Prabhakar et al (2005) noted that “IT Professionals know they must keep their skills up to date, but knowing so requires knowing what skills on demand” (p. 91). Possessing these skills is what we termed here as “Keeping a Pulse on the Market Demand”. Smith and Ali (2014) used web-mining techniques to assess market demand by mining data from popular job search sites. Figure 8 shows the trend of data gathered from 2010 to 2017 for programming languages and web frameworks. These graphs show the numbers of job postings taken from the Dice.com employment search site by referencing a selected programming language or technology. These show that Java has been, and still is, the highest demand of the selected programing languages, however, it has passed its peak while demand for Python continues to grow. Likewise, for web frameworks jQuery has passed its peak and angular is now the leading framework to compliment JavaScript (Figure 8).

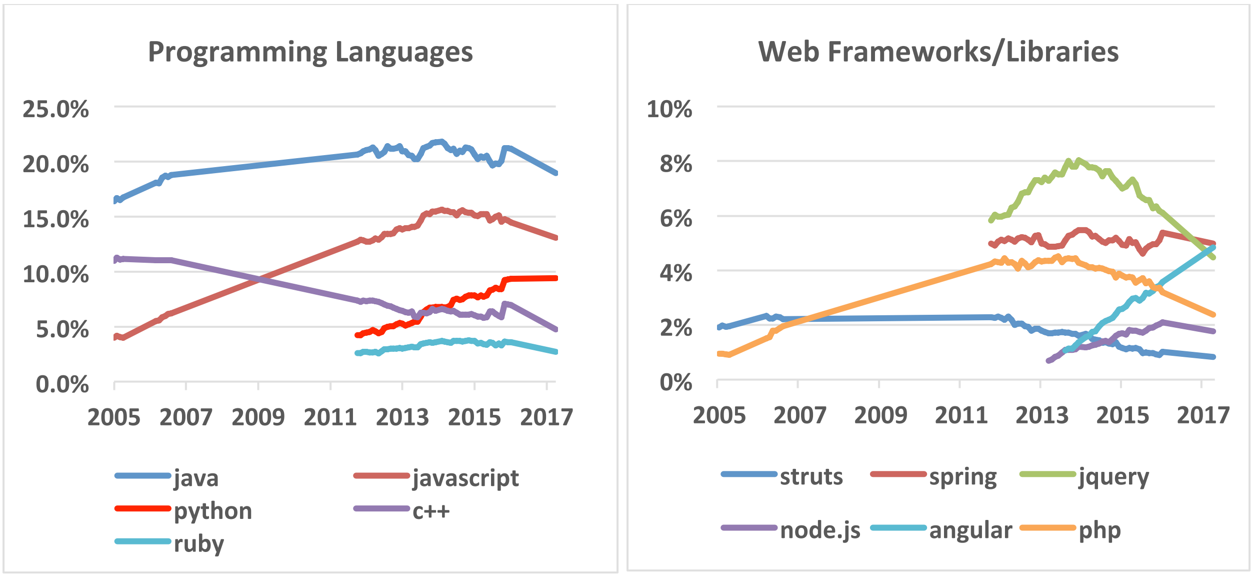


Figure 8: Percent Job Postings on Dice.com by Programming Language and by Web Frameworks/Libraries

The data for preceding charts was extracted from Dice.com. However, popularity of this site has declined in recent years. Gaining popularity is the site Indeed.com. While historical data to show trends has not been gathered from Indeed.com, the data for current demand is shown in Figure 9. Here, demand for Python has surpassed C++ and even JavaScript. This data indicates that Python is now the second highest demanded language (Figure 9).

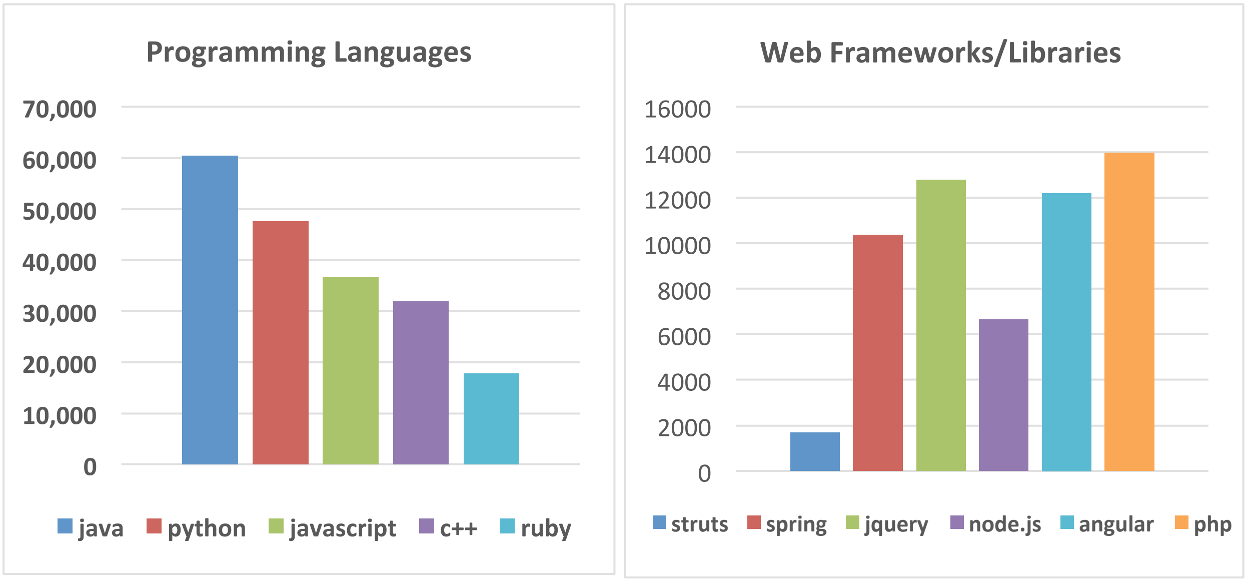


Figure 9: Number Job Postings on Indeed.com by Programming Language and by Web Frameworks/Libraries

For IT educators, this kind of knowledge plays a significant importance because they have the task of preparing the students for this fast changing environment. Reviewing trends in the job market is one way to get the educators informed of such job demand and make adjustments in course content accordingly.

The Trend Reversal of Our Programs

This section outlines the experience of the CS department at Indiana University of Pennsylvania, PA, USA, and list the efforts that helped reverse the downward enrollment trends in the program. Such efforts included partnering with other departments, introducing new programs, and making a number of changes to the programs. The remainder of this section elaborates on these efforts.

The CS department faced similar number of enrollment trend that was explained earlier in this paper. Figure 10 shows the admission trend of students into our program and Figure 11 shows the rate of graduations (right) for years 2000-2016. While a small dip occurred this past year (2017), this should be viewed in context of rural western Pennsylvania. In recent years, students graduating from area high schools have declined resulting in a decline in overall enrollment at the university. As overall enrollment improve, we anticipate enrollment in our programs to resume an upward trajectory.

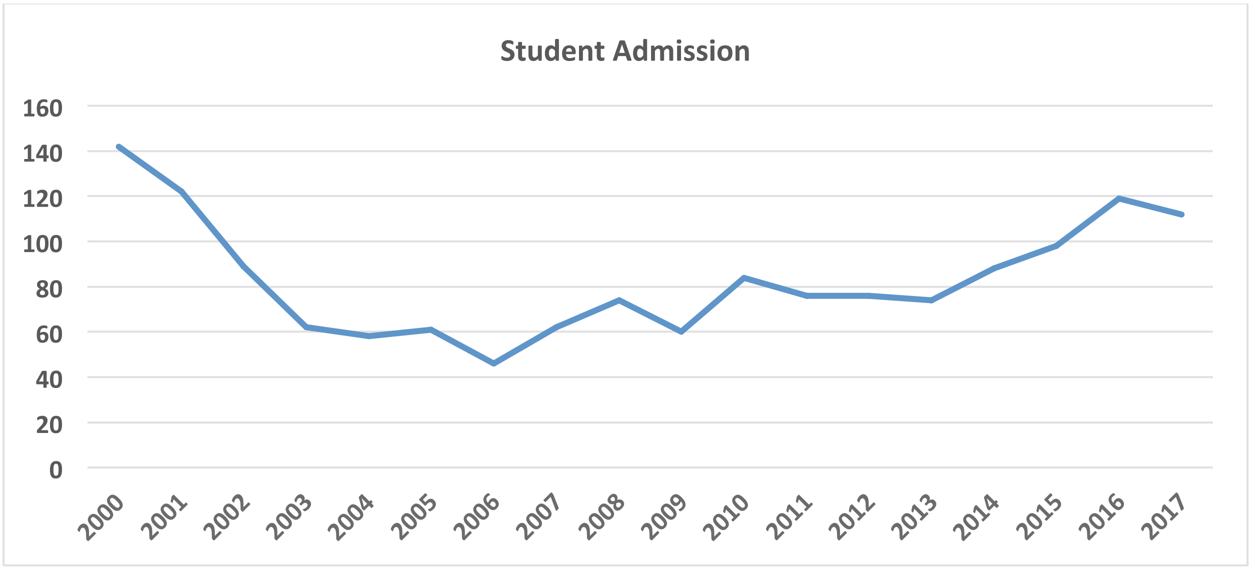


Figure 10: Student Admission in CS Programs at Indiana University of Pennsylvania between 2000-2017

We have had and remain using a late objects methods in our introductory programming course focusing on programing basics and learning fundamental algorithms. We have adopted large number of well-focused examples combined with many small assignments to lay the foundation for programming. Object oriented paradigm is introduced in the subsequent course, continues through the core and into several electives. Functional and logic programming are introduced in selected electives.

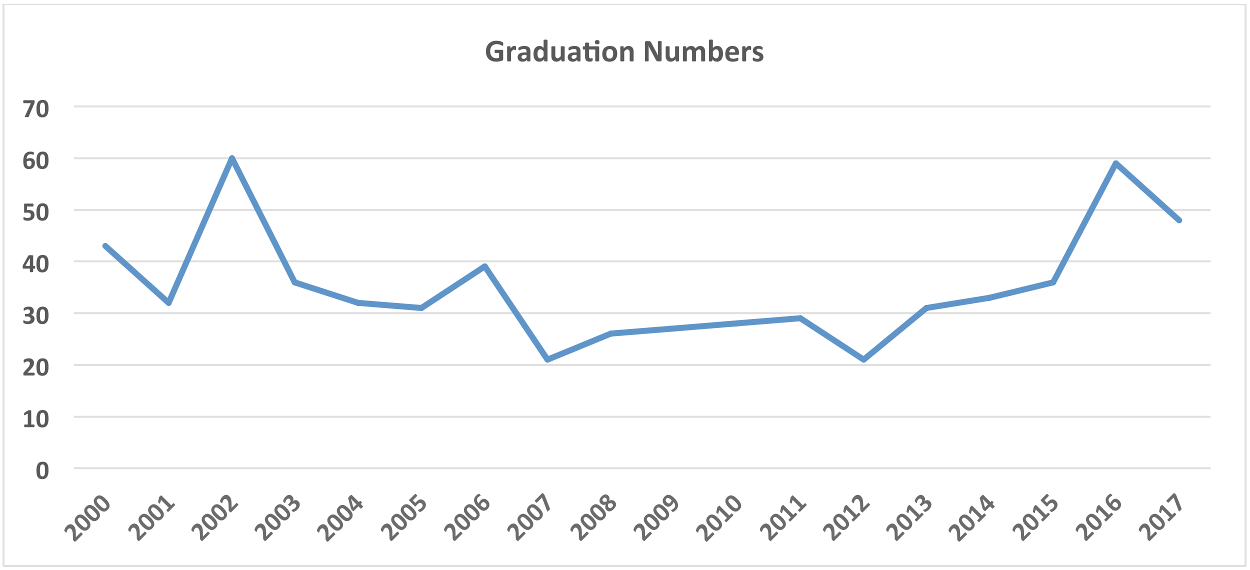


Figure 11: Graduation Numbers in CS Programs at Indiana University of Pennsylvania between 2000-2017

Our main goal from selecting the first programming language used in our introductory course is to respond to market demand and teach a programming language that is widely used in the industry. Given this, we chose Java, but limited use to a subset of constructs to support imperative methodology. Full coverage of Java is then completed in subsequent courses. This has worked well overcoming problems experience when a subset of C was used in the first course. However, with the now increasing industry demand for Python, we may soon consider adopting it in the first programming course. For now, we experimenting with selected use of Python in our overview of CS course that precedes the programming courses.

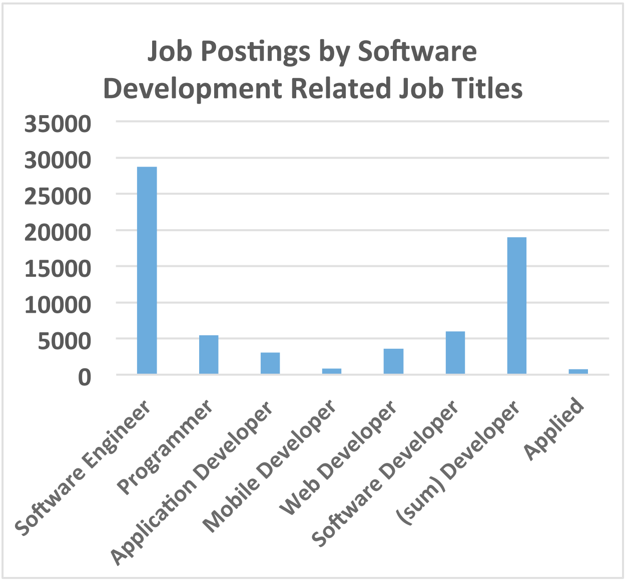
Similar to many other CS programs, we experimented with Alice programming language as an alternate first programming course with mixed success. While most students taking this alternative course did quickly grasp the basic programming concepts, the “movie paradigm” used in Alice was perhaps too compelling as a number of students would get engrossed in generating reams of code for their movie and missing the overall point of algorithms. Furthermore, the transition to Java was awkward as we consider Alice to be more object based (fixed set of object types to manipulate) than object oriented. This, and given the stagnation of Alice enhancements, led us to abandon this alternative.

The main directive for selecting additional programming languages and technologies in further courses is to respond to market demand. In other words, our goal is to prepare them to best marketable and enable them to readily find jobs on graduation. We have found the use of our web mining of employment search sites to be highly effective yet simple means to assess market demand of numerous programming languages and technologies. With this information, we continue to make adjustments in course content and our programs. For example our web programming course once placed emphasis on server side frameworks, such as struts, with nominal attention to JavaScript. Struts has since been abandoned and JavaScript along with jQuery now is the major component. However, based on the latest trend, the angular JavaScript library may soon replace jQuery. In addition, to changes within a course, new demands may lead to additional courses. We introduced a web services course that will use Amazon Web Services (ASW) and Restful as these are now in high demand.

Our attention to this market demand approach to selecting languages and technology is a major draw into our programs. While a case can be made that once someone learns a given programming language or technology, s/he can quickly learn another of similar structure and function, perspective students and families take note that we are best preparing them for the market to 'hit the ground running' upon hiring. We consider this to be a significant factor in our increased enrollment. Given the wealth of online resources, faculty members are adding references to these as part of the course materials. YouTube videos are often used both in class and after class assignments. Online tutorials are used to augment lectures. Online interactive programs are used to readily demonstrate algorithmic progressions, such as insertions into a tree structures. Given the wealth of online resources, some courses rely solely on these in place of traditional textbooks. In any event, these access to resources has contributed to student success.

Another improvement to our programs was to change the name of the program to be better aligned with the job titles in industry. The use of our web mining technique for assessing market demand for programming languages and technologies is well suited for assessing popularity of job titles. The Indeed.com site even provides a search for phrases in a job titles as of January 2018 (See Figure 12).

Based on similar numbers accessed three years ago, we renamed our CS Applied track to be CS Software Engineering track, since the number for this job title exceeds other job titles of the same nature. Software Engineer even exceeds the sum of job titles have some technology word followed by "Developer". Furthermore, we renamed our CS Information Assurance Track to be CS Cybersecurity Track. We believe these changes to be another factor in increased enrollment.



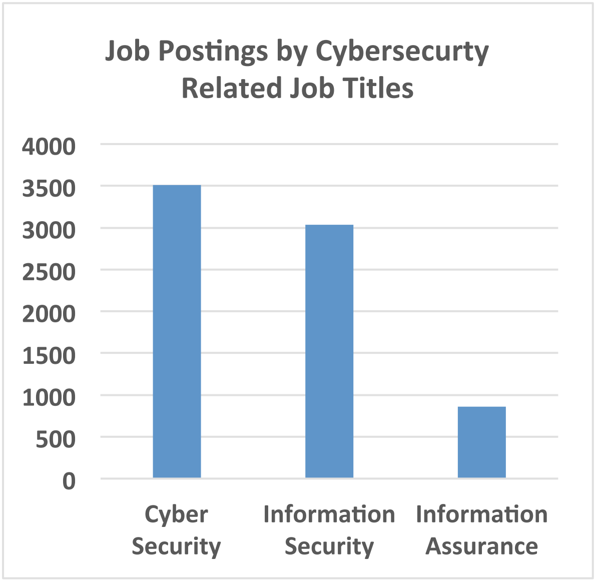


Figure 12: Number Job Postings on Indeed.com by Job Titles (as of January 2018)

One of the hallmarks of our department has been responding to market demand and preparing the students for the latest developments in the technology market. For this purpose, we offer an internship course providing experiential leaning at companies both large and small in the western Pennsylvania region and across the US. Ali and Smith (2015) explained the benefits of bringing internships for students and the benefits in keeping up with the demand on the new technologies. Students present their work after their completion of their internship with other students providing further interest by others. The general consensus that the students express is their appreciation of the abundance of technologies they learn from their internships.

Summary and Recommendations

This paper has presented the reversal of enrollment trends in CS programs. It began by providing the numbers and statistics supporting this claim followed by analysis of several factors we believe have an effect on enrollment. It then introduced steps that have been taken to deal with this issue of enrollment decline among students enrolled in technology programs. Last, this paper elaborated on the experience of the CS programs at our university (Indiana University of Pennsylvania, PA, USA), which were successful in reversing this trend of enrollment and as a result the number of students in our programs than have increased in recent years.

One thing we learned from the 'dot com' collapsed is that the changes in market demand can happen in a relatively short period. Our recommendations to academic program leaders is to routinely assess market demand and make changes within courses as well as programs in response to observable market changes, thereby, having graduates to be best prepared to enter the workforce. In our case, this resulted in promoting our program's reputation in industry and drawing in future students.

References

Adams, J. (2008). *Alice in action computing through animation*. Boston, MA: Course Technology.

Ali, A. & Shubra, C. (2010). Efforts to reverse the trend of enrollment decline in computer science programs. *Issues in Informing Science and Information Technology, 7*, 209-224

Ali, A. & Smith, D. (2015). An internship program at a computer science department – theoretical foundation and overall coordination. *Issues in Informing Science and Information Technology*, *12*(1), 1 -10.

Ali, A. & Smith, D. (2014). Teaching an introductory programming language in a general education course. *Issues in Informing Science and Information Technology*, *13*(1), 57-67.

Asiri, S. (2006). *Open source software*. Retrieved from ACM Digital Library, http://www.acm.org/dl

Baldwin, L. P., & Kuljis, J. (2001). Learning programming using program visualization techniques. *Proceedings of the 34th Hawaii International Conference on System Sciences*. Maui, HI, USA.

Benokraitis, V., Bizot, B., Brown, R., & Martens, J. (2009). Reasons for CS decline: Preliminary evidence. *Journal of Computing Sciences in Colleges*, *24*(3), 161-162.

Bhatt, G., Emdad, A., Roberts, N., & Grover, V. (2010). Building and leveraging information in dynamic environments: The role of IT infrastructure flexibility as enabler of organizational responsiveness and competitive advantage. *Information & Management*, *47*(7-8) 341-349.

Burton, T. T. (2013). Outsourcing revisited. *Industrial Engineer: IE*, *45*(5), 34-39.

Carlton, M., & Levy, Y. (2017). Cybersecurity skills: The cornerstone of advanced persistent threats (APTs) mitigation. *Online Journal of Applied Knowledge Management, 5*(2), 16-28.

Computing Research Association (2017). Generation CS: Computer science undergraduate enrollments surge since 2006. *Computing Research Association*. Retrieved from http://www.cra.org/crn/

Dalziel, J. (2003). Open standards versus open source in e-learning. *EDUCAUSE Quarterly Magazine*, *26*(4), 4-7. Retrieved from https://www.educause.edu/ir/library/pdf/eqm0340.pdf

Damicon. (2007). Damicon's list of open software. *Damincon*. Retrieved from http://www.damicon.com/resources/opensoftware.html

Dann, W., Copper, S., & Pausch, R. (2006). *Learning to program with Alice*. Upper Saddle River, NJ: Prentice Hall.

Deitel, P. D., & Deitel H. (2017). *Java how to program (early objects)* (11th Ed.). New York, NY: Pearson Education Inc.

Donohue, P., & Power, N. (2012). Legacy job titles in IT: the search for clarity. *Proceedings of the 50th annual conference on Computers and People Research* (pp. 5-10). Milwaukee, WI, USA.

Ehlert, A., & Schulte, C. (2010). Comparison of OOP first and OOP later – first results of reading the role of comfort level. *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 108-112), Bilkent, Ankara, Turkey.

Fisher, L. (2016). Booming enrollments. *Communication of the ACM*, *59*(7), 17-18.

Glassman, N. R., & Shen, P. (2014). One site fits all: Responsive web design. *Journal of Electronic Resources in Medical Libraries*, *11*(2), 78-90.

Guo, P. (2014). Python is now the most popular introductory teaching language at top U.S. universities. *Communication of the ACM Blog*, Retrieved from https://cacm.acm.org/blogs/blog-cacm/176450-Python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext

Guzdial, M. (2017). 'Generation CS' drives growth in enrollments. *Communication of the ACM*, *60*(7), 10-11.

Kaplan, R. M. (2010). Choosing a first programming language. *Proceedings of the 2010 ACM conference on Information Technology Education* (pp. 163-164), Midland, MI, USA.

Kelleher, C, & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environment and languages for novice programmers. *ACM Computing Surveys*, *37*(2), 83-137.

Leutenegger, S., & Edgington, J. (2006). A games first approach to teaching introductory programming. *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, Houston, TX, USA.

Machuca, L., & Solarte, P. (2016). Improving student performance in a first programming course. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 367-367), Arequipa, Peru.

National Center for Education Statistics (2018). Completions - postsecondary education: Bachelor's degrees. *Data-Planet™ Statistical Datasets by Conquest Systems, Inc. [Data-file]*. Dataset-ID: 017-004-002.

Parker, K., R., Ottaway, Chang, J., & Chao, J. T. (2006). A formal language selection process for introductory programming courses. *Journal of Information Technology Education*, *5*(1), 133-151.

Parker, K. R., & Davey, B. (2012). The history of computer language selection. In A., Tatnall (Ed.). *Reflections on the history of computing* (pp.166–179), Springer Berlin Heidelberg, Heidelberg, Germany.

Pendergast, M. O. (2006). Teaching introductory programming to IS students: Java problems and pitfalls. *Journal of Information Technology Education*, *5*, 491-515.

Prabhakar, B., Litecky, C. R., & Arnett, K. (2005). IT skills in a tough job market. *Communications of the ACM*, *48*(10), 91-94.

Rottman, J. W. (2006). Successfully outsourcing embedded software development. *Computer*, *39*(1), 55-61.

Shein, E. (2015). Python for beginners. *Communications of the ACM*, *58*(3), 19-21.

Sloan, R. H., & Troy, P (2008). CS 0.5: A better approach to introductory computer science for majors. *Proceedings of the 39th SIGCSE technical symposium on Computer science education* (pp. 271-275), Portland, OR, USA.

Solander, A. C., Forman, A. S., & Glasser, N. M. (2016). Ransomware--give me back my files!. *Employee Relations Law Journal*, *42(2),* 53-55.

Smith, D., & Ali, A. (2014). Analyzing computer programming job trend using web data mining. *Issues in Informing Science and Information Technology*, *11*(1), 203-214.

Uysal, M. P. (2012). The effects of objects-first and objects-late methods on achievements of OOP learners. *Journal of Software Engineering and Applications*, *5*, 816-822.

Warner, M. E., & Hefetz, A. (2012). Insourcing and outsourcing. *Journal of The American Planning Association*, *78*(3), 313-327.

Westfall, R. (2001). Technical opinion: Hello, world considered harmful. *Communications of the ACM*, *44*, 129-130.

Welton, D. (2005). The economics of programming languages. *Welton.it*. Retrieved from http://www.welton.it/articles/programming_language_economics.html

Xinogalos, S. (2016). Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. *Education and Information Technologies*, *21*(3), 559-588.

Zweben, S. (2009). Computing degree and enrollment trends. *Computing Research Association*. Washington, DC, USA.

Zweben, S. (2011). Undergraduate enrollment in computer science trends higher; doctoral production continues at peak levels. *Computing Research Association*. Washington, DC, USA.

Authors' Biographies

David T. Smith, Ph.D., Professor of Computer Science – Indiana University of Pennsylvania has 17 years' experience in academia and 21 years of industry experience in database systems, computer language development, and other systems programming. He holds a bachelor degree in Physics and Mathematics Education from Indiana University of Pennsylvania, an M.S. in Computer Science from University of Central Florida, and a Ph.D. in Computer Science from Nova Southeastern University. Dr. Smith is active in consultancy and has research interests in artificial intelligence, distributed object computing, data mining, and software engineering.

Azad Ali, D.Sc., Professor of Information Technology at Eberly College of Business – Indiana University of Pennsylvania has 34 years of combined experience in areas of financial and information systems. He holds a bachelor degree in Business Administration from the University of Baghdad, an M.B.A. from Indiana University of Pennsylvania, an M.P.A. from the University of Pittsburgh, and a Doctorate of Science in Communications and Information Systems from Robert Morris University. Dr. Ali's research interests include service learning projects, web design tools, dealing with isolation in doctoral programs, and curriculum.