Careers in Software: Is there Life after Programming?

Abstract

Students interested in information systems and information technology careers can see a clear career path – from programmer to chief information officer (CIO). However, those interested in computer science or software engineering are attracted to the technology and seem not to consider what happens at later stages of their careers. This paper draws on a three-year study where senior software practitioners across a variety of roles were interviewed about their work. The findings show profound difficulties in making the transition from hands-on technical work to a role where they achieve through guiding the efforts of others.

1 Introduction

In the information technology/information systems world, there is a very distinct career path: programmer, analyst, IT manager and eventually CIO (Lee 2005). The skills needed to support such a career were identified as far back as the 1990's – categorized as technical, business, inter-personal and the application of technology to business (Trauth, Farwell and Lee 1993; Lee, Trauth and Farwell 1995).

Essentially, the successful IS/IT person is one that understands IT but is also expert in business processes and their requirements. Recent research into IT skills also supports this breakdown (Huang, Kvasny, Joshi, Trauth and Maher 2009; Taylor and Woelfer 2009). Despite the changing face of technology – mainly the emergence of web-based services – the need to apply technical

solutions to business problems remains. As Lee and Mirchandani (2009) put it, “IS managers at higher-level positions should understand the relationships between business strategies and their related technologies” (p.118). In other words, school-leavers contemplating a future career can see how a qualification in IS/IT can equip them for productive employment until retirement age. Significantly, “IS programs include programming, but focus on problem solving, design, development, organizational applications and implementation” (Clark, Walz and Wynekoop 2003, p.141). They also include business modules and promote working in groups.

In contrast, a career in software engineering is much less obvious. Because computers are now employed in all application domains, it is difficult to pin down in what industry software engineering graduates will find themselves working. Essentially, the expertise they acquire during their under-graduate studies makes them adept in all aspects of the computer, which is a general-purpose tool – not a sound basis for any profession (Holmes 1999). However, it is unlikely that they will have any knowledge about the industry where they are expected to apply this expertise.

This lack of domain knowledge makes it difficult for software engineers to understand their customers’ requirements. Each domain has its own set of ‘common sense’ constructs that are very difficult to convert into a deterministic formula that a computer understands. For instance, we are all familiar with the concept of a car crash. However, how does your airbag controller know that you are crashing and not simply driving over a rutted surface? It is little wonder that Jackson (1998) proposed establishing a discipline around descriptions like these.

Another worrying feature of the software engineering discipline is the tendency not to value technical experience. McGovern (1998) describes a rapid promotion path for graduate software people up to a certain point. Then the company would prefer that person either to leave or to move into a non-technical role, such as

management or sales. Unfortunately, management does not appear to appeal to software people in general. An Australian study shows that software practitioners "liked what they do and did not want the responsibility and headaches of managing and developing people." (Gallagher, Kaiser, Frampton and Gallagher 2007, p.17)

Further confusion for people considering a software career relates to roles. There are no clear role definitions in the software world (Belbin 2000; Shaw 2000; Gallagher et al. 2007; Downey 2009). Two people with the same job title could have widely differing responsibilities. Various factors come into play, including history with the company and co-workers' skill sets.

Finally, the spectacle of hundreds of software engineers losing their jobs as companies decide to off-shore or outsource their software development to lower-wage economies has certainly affected the number of students opting for software-related university courses.

2 Project background and methodology

This paper uses data gathered during a three-year study of senior software practitioners to gain insight into the question of software engineering career paths. Including a pilot study, a total of thirty-eight senior software practitioners were interviewed. These people were in their thirties or forties, at or approaching what Dalton, Thompson and Price (1977) term Stage III of their careers. Dalton et al's career stage model describes four distinct career stages: Stage I being the apprenticeship stage, where supervision is required. Stage II is where you work competently on your own, with minimal supervision. Stage III is when you move from accomplishing through your own, hands-on efforts to leading and mentoring others. Finally, Stage IV is when you have an executive role and are concerned with guiding the organization as a whole and grooming others for high office.

Lethbridge, Sim and Singer (2005) note that many “of the field studies in software engineering tend to be exploratory in nature, because we are still gathering basic knowledge about the human factors surrounding software development and maintenance” (p.314). For this reason, a grounded theory approach was chosen, where data would be gathered without any a priori influence in order to develop a theory (Strauss and Corbin 1998).

The plan was to interview senior practitioners across seven different roles. These were product managers, project managers, architects, programmers, testers, technical writers and customer support personnel. To provide construct validity, project teams were identified in four different companies and people carrying out the various roles were chosen in each of the four companies. This is what Yin (1994) describes as a multiple-case embedded case-study.

It was decided to adopt a semi-structured interview approach for a variety of reasons. Firstly, senior practitioners are difficult to access and it is important that all relevant data is gathered in one session – it is unlikely that another one will be possible. Secondly, there is a real possibility in the software world that the interviewee will be shy and inarticulate, so an interview checklist will help to keep the conversation going. Finally, at the end of the day, the data gathered from the individual interviews will have to be compared with each other. Having a structure to the interview facilitates this.

While grounded theory is an excellent way of removing pre-conceived biases from the data gathering exercise, it also serves up the problem of obtaining all relevant information without knowing in advance exactly what information will be required. It was decided to use Bandura’s “social cognitive theory” (Bandura 1986) to inform the interview instrument. This theory states that a person is defined by personality, behaviour and environment and that these three factors all influence each other reciprocally.

The resulting interview instrument has three overall questions as follows (Downey 2005):

1. **What do you do?** A checklist of software development activities informed by the software engineering (SWEBOK - IEEE Computer Society 2004) and the project management (PMBOK - IEEE Computer Society 2003) bodies of knowledge was added under this question to structure and guide the conversation.
2. **What skills do you have?** This study's original goal was to learn about a person's skill set, so the checklist included here was derived from skills research carried out in the IS/IT arena (Trauth et al. 1993; Lee et al. 1995; Sawyer, Eschenfelder, Diekema and McClure 1998; Shi and Bennett 1998; Noll and Wilkins 2002).
3. **Where did you acquire these skills?** This is a biographical section where early career motivation, academic history, work experience and outside interests were explored in order to determine how the interviewees obtained their skill sets.

The interviews themselves were given a maximum time limit of two hours. However, most were completed within ninety minutes. Although the interview was recorded, extensive notes were taken during the session. This activity seemed to put the interviewees at ease and it also allowed the researcher to write up a brief report based on these notes immediately after the interview. When the first draft of the report (ranging between three and six pages, depending on the interview) was complete, the researcher listened to the recording again, to clarify certain points and to ensure that important items had not been overlooked.

The revised report was then e-mailed to the interviewee for what Miles and Huberman (1994) call "member checking". This allows the informant to correct factual errors, clarify misunderstandings and remove points that they are

uncomfortable with. It is this last aspect that prompts Miles and Huberman's warning that such edits "can improve both the quality of data and the quality of final conclusions, but they also can result in truncated or distorted conclusions if someone has been given, and exercises, the right of censorship" (p.48).

Each report, having been approved by the interviewee, was entered into the NVivio software package and a process called "open coding" was carried out. Every sentence of the report was classified under a specific code. If the concept expressed in that sentence was new, a new code was created; otherwise, the sentence was filed under an existing code. It was noticed, as the interviews proceeded, that fewer new codes were needed, until the study reached the point where interviews could be successfully classified using existing codes. This situation is called 'saturation' (Strauss and Corbin 1998).

Throughout the study, efforts were made to classify the growing list of open codes into higher-order groupings. This process is called 'axial coding' and this particular study required several attempts before useful orderings emerged. Full details of the entire study and its findings can be found in Downey (2009).

For this paper, the open codes were revisited and those relating to careers extracted. These are presented in the next section.

3 Findings

As Figure 1 shows, there are many ways to become a software practitioner. This picture has been built up from answers to the third question: Where did you acquire these skills? While the checklist asked about university education before exploring work experience, it was often the case that the interviewees would describe their careers to date in chronological order. This showed several interviewees alternating between formal education and the workplace – with some having worked in areas as diverse as catering, marketing and retail outlets.

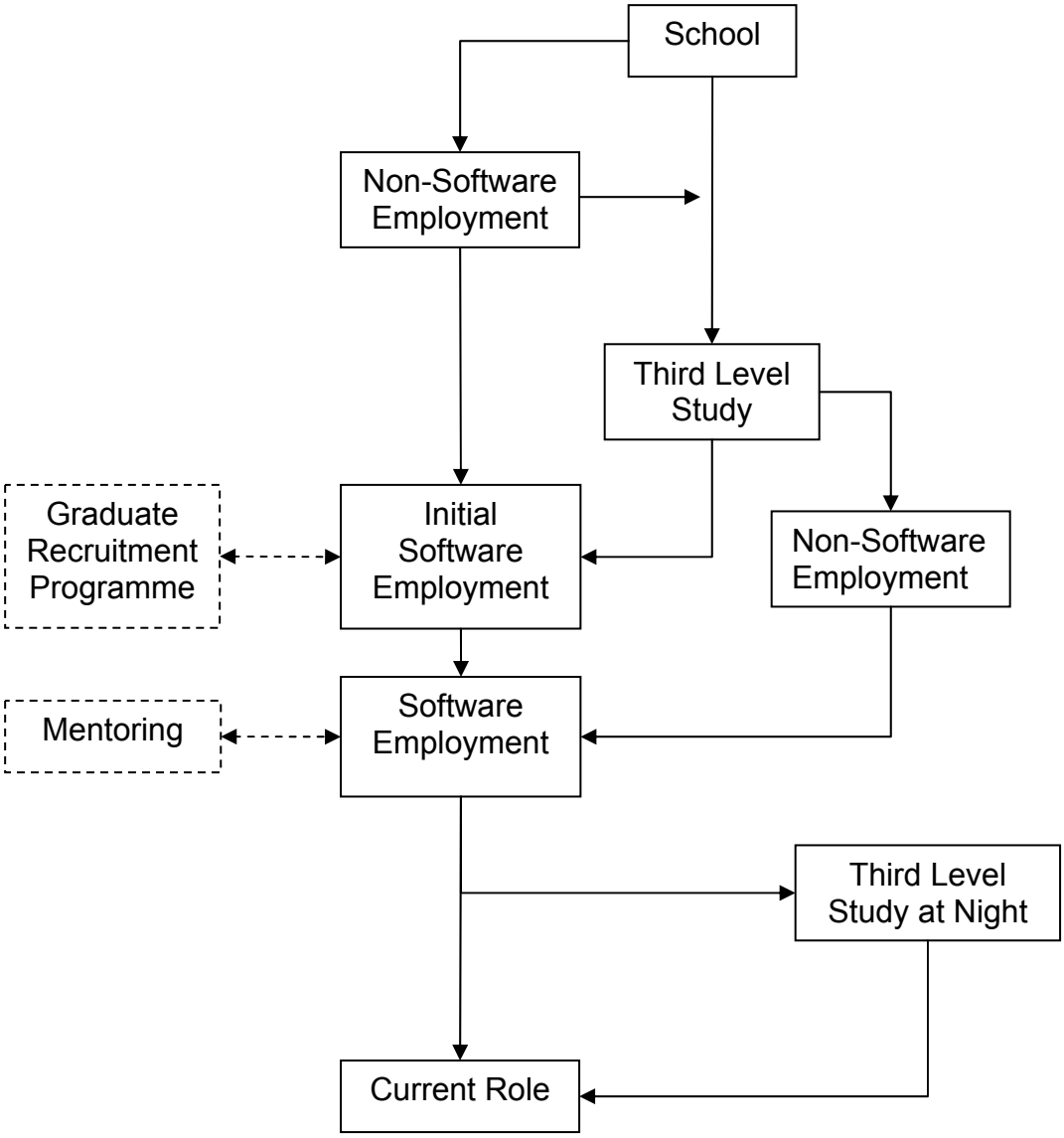

Figure 1 The Origins of a Software Practitioner

What is striking about the resulting diagram is the fact that the lack of a university-level qualification is not a barrier to entering the profession. According to Joseph, Ang and Slaughter (2005), there is evidence of a 'protean' career path in IT. This is where someone from outside the IT world studies an emerging technology and works in the area for a while. "The very existence of a protean career path suggests that the IT profession has low barriers for entry and exit. This ease of entry and exit suggests a lack of professionalization that may arise

from constantly experiencing a rapid rate of technological advancement and slow conformance to technological standards" (p.96). Although this particular study has no evidence relating to exit paths from software development, it does offer support for the idea that the barriers to entry into software development are also low.

Table 1 also illustrates the diverse career paths followed by thirty of the thirty-eight interviewees- those who participated in the main study. What is noticeable is that twenty-five of these thirty practitioners have had some sort of managerial experience and several of these (noticeably those in the "systems architect" role) have returned from a managerial role to a technical one.

Role	Career Path
Product Manager	
1	Programmer ► trainer ► product manager
2	Programmer ► product engineering manager
3	Programmer ► test ► customer support ► product mgr.
4	H/W design ► programmer ► technical sales manager
Project Manager	
1	Programmer ► group leader ► R&D mgr ► group mgr
2	H/W engineer ► programmer ► project manager
3	H/W test ► group leader ► R&D manager ► section mgr
4	Programmer ► group leader ► development manager
Systems Architect	
1	Programmer ► group leader ► systems architect
2	Programmer ► project mgr ► systems architect
3	Programmer ► group leader ► member of technical staff
4	Programmer ► member of technical staff
5	Programmer ► customer support ► chief technical officer
Programmer	
1	Programmer ► group leader ► senior S/W engineer
2	Network S/W engineer (informal team lead)
3	Senior S/W engineer (informal team lead)
4	S/W developer (informal team lead)
Tester	
1	H/W test ► S/W test engineer
2	H/W test ► programmer ► S/W test ► test group manager
3	S/W test ► system test engineer (group leader)
4	Customer support ► lead test engineer
5	Programmer ► test engineer
6	H/W technician ► programmer ► S/W test engineer
Technical Writer	
1	Programmer ► technical writer (group leader)
2	Marketing ► senior technical author (informal team lead)
3	Technical writer
Customer Support	
1	Customer support ► product support manager
2	Programmer ► customer support ► technical marketing
3	Programmer ► customer support ► support resource mgr
4	H/W test ► network operator ► customer support engineer

Table 1. Partial Career Paths

However, the overriding impression given by the interviewees is that they do not like managerial responsibilities. This distaste can be attributed to a range of

factors. The most obvious being the personality profile of the interviewees. Several mentioned that they tended to be introverted in character. For instance, one informant made this observation explicitly when asked about presentation skills:

> "It's not my favourite part of my job. Personality-wise I tend to be, and I guess a lot of engineers tend to be, more introverted than extroverted."

Another, related observation was that communication was a problem for some people:

> "We weren't trained as engineers to be communicators. Some people are very good at it – it's a skill. It's more a natural one. You could improve it with learning I think or practice, but you don't tend to get a lot of practice."

This indictment of engineering education suggests that the people interviewed in this study were not well prepared for a managerial role. There also seems to be little or no support for managers in the workplace, particularly when they take up their first front-line position. One of the interviewees pointed out that, while mentoring was very useful in her early career, she found very little support later on. DeMarco (2001) concurs that mentoring is lacking. Managers rarely get groomed. They might be sent on management courses that teach abstractions but they rarely have a 'facilitator' and they don't get encouraged to work with their 'co-learners'.

A noticeable feature of this industry is the tendency to assign senior programmers to informal team leading responsibilities. However, these programmers are still expected to carry out sizeable amounts of coding work as well as manage the team. The conflict between the attention to detail needed for the technical work and the broad focus required for the leadership role is particularly difficult for someone attempting both roles at once.

Added to this difficulty is the informality of the arrangement – the nominated team leader is given the responsibility to manage the team or the project, but not the official authority.

> "That can be very difficult and sometimes, when you don't own the resources – it's very frustrating. We're given all the responsibility of delivering something without being given the full authority to do it."

Support for the new manager differs across the organizations. In one multinational, a group leader claimed that it is "hard to be a bad manager here".

> "All managers have to do professional training provided by an external company. This would be on management skills – the soft skills – and it's quite detailed. You get a whole psychoanalysis done on you and you identify your weaknesses and your strong points. People are interviewed about you and it's all presented and you're rated. Then they focus on the areas: 'This is your weakness, this is your strength'. It's quite intense."

However, in another company, a newly appointed team leader decided to do a management diploma at night to learn the skills he felt he needed.

Given this general lack of preparation for management work, it is not surprising that many software practitioners would prefer to remain on a technical career ladder. It is widely agreed in the literature that it is the work itself that is the primary motivator (Herzberg, Mausner and Snyderman 1959; Hackman and Oldham 1980; Thatcher, Liu and Stepina 2002). However Allport (1937) provides a useful caveat, one that is very applicable to this study: "When skills are mastered, they have no longer any motivational character" (p.247). One of the programmers interviewed had moved through several groups within his company to gain skills in different technologies, pursuing what Schein (1990) calls a circumferential career path. Gallagher et al (2007) also find support for this phenomenon: "Responding to why they might leave or what caused turnover in

these positions, the lack of a challenge was the overwhelming response. Several stated that if they had to do the same work for an extended period, they would rather change jobs or even work for another firm" (p.17).

So, while the managerial career path is not attractive and the circumference one lacks financial reward, is the technical career ladder the right progression for the technically-minded practitioner? The first step on this ladder is the software or systems architect role.

It is noticeable in Table 1 that those who are currently in the architect role have earlier been front-line managers, suggesting that companies who offer dual-track career paths want to encourage their senior people into management first. Promotion is considered a way to retain staff in some companies and, as DeMarco and Lister (1999) point out, this results in people being promoted too soon. They recommend that at least ten years' experience is needed before promotion to front-line management. Combined with the lack of preparation and support for novice managers, this might explain why disillusioned managers return to technical roles.

A further interesting observation is the nature of the systems architect role. Instead of being a wholly technical role, the informants in this study seemed to carry out a similar function to systems analysts in IS/IT (Downey 2006). Their main concerns are the refinement of user requirements and the development of time and headcount estimates. The former task puts the architect close to the customer, either interfacing directly or going through a product management function. The latter work feeds into feasibility studies and later to the project plans. For people who do not want managerial responsibilities, it is ironic that many architects are expected to be the technical leads for the projects they specify.

Another related factor here is the way “contemporary work is becoming increasingly complex with shifts toward team- and project-based work resulting in a blurring of role boundaries” (Dierdorff and Rubin 2007, p.598). This is clearly seen in the senior programmer role. This is probably the most pervasive of all the roles, with programmers being involved from the requirements elicitation stage all the way through to assisting the customer support personnel installing the product on site. They are expected to display customer-facing skills as well as being involved as informal team leaders. It would appear from the interview data that the role of the programmer encompasses all the other roles and suggests that a programmer really needs to be able to assist in any aspect of the development life-cycle.

Exploring the practitioners’ early career motivation shows an aptitude for science and engineering, with many of the informants having access to computers, either in school or at home. There is an overwhelming love for technical work, so much so that none of the managers expressed any difficulty in motivating their staff. However, enthusiasm for managerial roles was not seen in this study. From the architect who dismissed the role as being that of a “paper pusher” to the programmer who only applied for the team leader job because he “was afraid of who would get it” if he did not, the consensus is clear: management is not attractive to these technical people.

Unfortunately, at some stage in the software practitioner’s career, s/he will have to manage and mentor other people. It would be helpful if students were prepared for this transition when entering the profession and if those making the transition were given more support by their employers.

4 Conclusion and Recommendations

Computer science and software engineering courses tend to have a largely technical content. Similarly, job adverts for software positions invariably look for people skilled in particular technologies (Litecky, Arnett and Prabhakar 2004). So it is not surprising that people who love computer technology are drawn to careers in software development.

However, as we have seen, the personality profile of those recruited into this industry tends to be introverted and they need constant technical stimulation. It would appear from the interviews that the best way to keep technical workers happy is to rotate them regularly onto different technologies. Certainly, the practice noted in McGovern (1998) and criticized by DeMarco and Lister (1999) of promoting technical people rapidly seems to be problematic for many practitioners.

While the circumferential career path mentioned earlier seems an ideal way to maintain a practitioner's interest as well as avoiding managerial responsibilities, it must be remembered that "[c]ontinuous re-skilling is a costly investment for the individual and employer" (Calitz, Watson and Kock 1997, p.31). Also, many companies are cutting back on training, especially when staff turnover is high (Sumner 2001; Conn 2002). However, the main problem with this strategy is the perceived worth to the company of the person. As one of the interviewed architects points out:

> "With all these engineers coming out of college, computer programmers coming out of college, you're in competition with them. You're probably on twice their salary and you're going to have to keep pushing yourself to keep ahead."

The problem appears to be moving from Stage II of Dalton et al's (1977) career model to Stage III. In other words, software developers need to make the

transition from dependence to independence. Instead of waiting to be told what to do, the practitioners now should be able to develop their own ideas of what needs to be done in a given situation. While many of the interviewees are comfortable with deciding on the technical issues of their work, they are less comfortable with the other facets of this stage – mentoring junior personnel, broadening their interests beyond the technical sphere and dealing with people outside the group. This last area is particularly difficult as those in other departments – sales and marketing personnel, finance and legal professionals - as well as customers and end-users generally have different concerns and, for all intents and purposes, speak a different language.

Furthermore, while Stage II saw the workers build their own confidence, now confidence must be instilled in others. Also, the Stage III worker must not be afraid of subordinates – s/he needs to give them the freedom to develop. While in Stage II, the workers have responsibility for their own work, now they must be willing to take responsibility for others'. Some find this difficult. Also there is a fear that moving away from a technical role will make them less employable if they lose touch with the technical aspects.

Unfortunately remaining at Stage II not only places the practitioner at risk from recent graduates, but there is also the very real danger that the technical work will be off-shored or outsourced to a lower-wage economy (Herbsleb and Morita 2001).

Finally, many programmers in the software development world have the title 'software engineer', a practice that has been criticized in the past (Baber 1982; Parnas 1999). Yet an engineer has been defined as: "the manager and facilitator of a technical process. Professional engineers do not themselves do the hack-work of their projects, though it is an important part of their apprenticeship to do this" (Holmes 1999, p.82).

This attachment to “the unambiguous and immediate gratifications obtainable from solving concrete technical problems” (Bailyn and Lynch 1983, p.281) is seen in other engineering disciplines as well, but people who wish to pursue careers in software engineering must be prepared to move away from hands-on work. Computer science and software engineering courses must prepare their students for this transition and their companies must offer a great deal more support and encouragement to facilitate the new stage in their employees’ careers. However, Smits, McLean and Tanner (1993) argue that “the ultimate responsibility for career management lies with the person” and employees should take steps to develop the necessary Stage III skills by “volunteering for committee assignments, speaking to community groups, coaching new employees, participating in academic and staff development programs designed to build specific ‘people skills’, and so forth” (p.117).

5 Acknowledgement

This work is partially supported by Science Foundation Ireland (SFI) under grant number 03/CE2/I303-1.

6 References

Allport, G. W. (1937). Pattern and Growth in Personality. Cambridge, Massachusetts, Holt, Rinehart and Winston.

Baber, R. L. (1982). Software Reflected: The Socially Responsible Programming of our Computers. Amsterdam, North Holland.

Bailyn, L. and J. T. Lynch (1983). "Engineering as a life-long career: its meaning, its satisfactions, its difficulties." Journal of Occupational Behaviour **4**(4): 263-283.

Bandura, A. (1986). Social Foundations of Thought & Action: A Social Cognitive Theory. Englewood Cliffs, New Jersey, Prentice Hall.

Belbin, R. M. (2000). Beyond the Team. Oxford, Elsevier Butterworth-Heinemann.

Calitz, A. P., M. B. Watson, et al. (1997). Identification and Selection of Successful Future IT Personnel in a Changing Technological and Business Environment. Proceedings of the 1997 SIGCPR Conference on Computer Personnel Research, San Francisco, ACM Press.

Clark, J. G., D. B. Walz, et al. (2003). "Identifying Exceptional Application Software Developers: A Comparison of Students and Professionals." Communications of the Association for Information Systems **11**(8): 137-154.

Conn, R. L. (2002). "Developing Software Engineers at the C-130J Software Factory." IEEE Software **19**(5): 25-29.

Dalton, G. W., P. H. Thompson, et al. (1977). "The Four Stages of Professional Careers - A New Look at Performance by Professionals." Organizational Dynamics **6**(1): 19-44.

DeMarco, T. (2001). Slack - Getting Past Burnout, Busywork and the Myth of Total Efficiency, Dorset House.

DeMarco, T. and T. Lister (1999). Peopleware: Productive Projects and Teams, Dorset House.

Dierdorff, E. C. and R. S. Rubin (2007). "Carelessness and Discriminability in Work Role Requirement Judgments: Influences of Role Ambiguity and Cognitive Complexity." Personnel Psychology **60**(3): 597-625.

Downey, J. (2005). A Framework to Elicit the Skills Needed for Software Development. 2005 ACM SIGMIS CPR Conference, Atlanta, Georgia, ACM Press.

Downey, J. (2006). Systems Architect and Systems Analyst: Are These Comparable Roles? 2006 ACM SIGMIS CPR Conference, Claremont, California, ACM Press.

Downey, J. (2009). Career Paths for Programmers: Skills in Senior Software Roles. Newcastle, England, Cambridge Scholars Publishing.

Gallagher, K. P., K. Kaiser, et al. (2007). Best Practice for Grooming Critical Mid-Level Roles. ACM SIGMIS CPR, St. Louis, Missouri, ACM Press.

Hackman, J. R. and G. R. Oldham (1980). Work Redesign. Reading Massachusetts, Adison-Wesley.

Herbsleb, J. D. and D. Morita (2001). "Global Software Development." IEEE Software **18**(2): 16-20.

Herzberg, F., B. Mausner, et al. (1959). The Motivation To Work, John Wiley & Sons Inc.

Holmes, N. (1999). "Software Engineering: To Be or What To Be?" Software Engineering Notes **24**(3): 81-83.

Huang, H., L. Kvasny, et al. (2009). Synthesizing IT Job Skills Identified in Academic Studies, Practitioner Publications and Job Ads. 2009 ACM SIGMIS CPR, Limerick, Ireland, ACM Press.

IEEE Computer Society (2003). IEEE Guide Adoption of PMI Standard: A Guide to the Project Management Body of Knowledge. New York, IEEE Inc.

IEEE Computer Society (2004). Guide to the Software Engineering Body Of Knowledge. Los Alamitos, California, IEEE Computer Society, Los Alamitos, California.

Jackson, M. (1998). "Defining a Discipline of Description." IEEE Software **15**(5): 14-17.

Joseph, D., S. Ang, et al. (2005). Identifying the Prototypical Career Paths of IT Professionals: A Sequence and Cluster Analysis. 2005 ACM SIGMIS CPR Conference, Atlanta, Georgia, ACM Press.

Lee, C. K. (2005). Transferability of Skills over the IT Career Path. 2005 ACM SIGMIS CPR Conference, Atlanta, Georgia, ACM Press.

Lee, D. M. S., E. M. Trauth, et al. (1995). "Critical Skills and Knowledge Requirements of IS Professionals: A Joint Academic/Industry Investigation." MIS Quarterly **17**(3): 313-340.

Lee, K. and D. Mirchandani (2009). Analyzing the Dynamics of Skill Sets for the U.S. Information Systems Workforce Using Latent Growth Curve Modeling. 2009 ACM SIGMIS CPR, Limerick, Ireland, ACM Press.

Lethbridge, T. C., S. E. Sim, et al. (2005). "Studying Software Engineers: Data Collection Techniques for Software Field Studies." Empirical Software Engineering **10**(3): 311-341.

Litecky, C. R., K. P. Arnett, et al. (2004). "The Paradox of Soft Skills versus Technical Skills in IS Hiring." Journal of Computer Information Systems **45**(1): 69-76.

McGovern, P. (1998). HRM, Technical Workers and the Multinational Corporation, Routledge.

Miles, M. B. and A. M. Huberman (1994). Qualitative Data Analysis. Thousand Oaks, California, Sage Publications.

Noll, C. L. and M. Wilkins (2002). "Critical Skills of IS Professionals: A Model for Curriculum Development." Journal of Information Technology Education **1**(3): 143-154.

Parnas, D. L. (1999). "Software Engineering Programs Are Not Computer Science Programs." IEEE Software **16**(6): 19-30.

Sawyer, S., K. R. Eschenfelder, et al. (1998). "Corporate IT Skill Needs: a Case Study of BigCo." ACM SIGCPR Computer Personnel **19**(2): 27-41.

Schein, E. H. (1990). Career Anchors: Discovering Your Real Values. San Diego, Pfeiffer & Co.

Shaw, M. (2000). Software Engineering Education: A Roadmap. 22nd International Conference on Software Engineering, Limerick, Ireland, ACM Press.

Shi, N. and D. Bennett (1998). "Critical Success Factors for IS Executive Careers - Evidence from Case Studies." ACM SIGCPR Computer Personnel **19**(3): 34-54.

Smits, S. J., E. R. McLean, et al. (1993). "Managing High-Achieving Information Systems Professionals." Journal of Information Management Systems **9**(4): 103-120.

Strauss, A. and J. Corbin (1998). Basics of Qualitative Research. Thousand Oaks, California, Sage Publications, Inc.

Sumner, M. (2001). A Report on Industry-University Roundtable Discussions on Recruitment and Retention of High-Tech Professionals. ACM SIGCPR Conference on Computer Personnel Research.

Taylor, H. and J. P. Woelfer (2009). Critical Skills for IT Project Management and How They are Learned. 2009 ACM SIGMIS CPR Conference, Limerick, Ireland, ACM Press.

Thatcher, J. B., Y. Liu, et al. (2002). The Role of the Work Itself: An Empirical Examination of Intrinsic Motivation's Influence on IT Workers' Attitudes and Intentions. SIGCPR '02, Kristiansand, Norway, ACM.

Trauth, E. M., D. W. Farwell, et al. (1993). "The IS Expectation Gap: Industry Expectations Versus Academic Preparation." MIS Quarterly **17**(3): 293-307.

Yin, R. K. (1994). Case Study Research: Design and Methods. Thousand Oaks, California, Sage Publications, Inc.