# Career Paths for Programmers

# Career Paths for Programmers: Skills in Senior Software Roles

By

Jack Downey

CAMBRIDGE SCHOLARS

PUBLISHING

Career Paths for Programmers: Skills in Senior Software Roles, by Jack Downey

This book first published 2009

Cambridge Scholars Publishing

12 Back Chapman Street, Newcastle upon Tyne, NE6 2XX, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

This research has been supported by the Science Foundation Ireland Investigator Programme, B4-STEP (Building a Bidirectional Bridge Between Software ThEory & Practice).

ISBN (10): 1-4438-0242-5, ISBN (13): 978-1-4438-0242-0

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# List of Appendices

# LIST OF ACRONYMS

| | |
|---|---|
| 3GPP | Third Generation Partnership Project |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| B.A. | Bachelor of Arts |
| B.Comm. | Bachelor of Commerce |
| B.E. | Bachelor of Engineering |
| B.Eng. | Bachelor of Engineering |
| B.Ind.Eng. | Bachelor of Industrial Engineering |
| B.Sc. | Bachelor of Science |
| BSC | British Computer Society |
| B.Tech. | Bachelor of Technology |
| CASE | Computer Aided Software Engineering |
| CATP | Customer Acceptance Test Plan |
| CCITT | Commité Consultatif International de la Télèfonique et de la Télègraphique |
| CD | Compact Disc |
| CM | Configuration Management |
| CMM | Capability Maturity Model |
| CNRC | Cellular Network Resolution Centre |
| COTS | Commercial Off The Shelf |
| CSR | Customer Support Request |
| CTO | Chief Technical Officer |
| CV | Curriculum Vitae |
| DSS | Decision Support Systems |
| EC | European Community |
| ETSI | European Telecommunications Standards Institute |
| EU | European Union |
| EVA | Earned Value Analysis |
| FDI | Foreign Direct Investment |
| FERB | Front End Review Board |
| FRDB | Feature Request Database |
| GAA | Gaelic Athletic Association |
| GPRS | General Packet Radio Service |
| GSS | Group Support Systems |
| HR | Human Resources |

| | |
|---|---|
| HRM | Human Resource Management |
| H/W | Hardware |
| ICT | Information and Communications Technology |
| IEEE | Institute of Electrical and Electronic Engineers |
| IITD | Irish Institute of Training and Development |
| IMI | Irish Management Institute |
| IP | Internet Protocol |
| IS | Information Systems |
| IT | Information Technology or Institute of Technology |
| ITU | International Telecommunications Union |
| J2EE | Java 2 platform, Enterprise Edition |
| JDBC | Java Data Base Connectivity |
| JFC | Java Foundation Classes |
| JMS | Java Messaging System |
| JNDI | Java Naming and Directory Interface |
| KSA | Knowledge and Skills/Abilities |
| LDAP | Lightweight Directory Access Protocol |
| MAP | Mobile Application Part |
| M.Eng. | Master of Engineering |
| M.Eng.Sc. | Master of Engineering Science |
| MIS | Management Information Systems |
| MMS | Multi-media Message Service |
| M.Phil. | Master of Philosophy |
| MS | Microsoft® |
| M.Sc. | Master of Science |
| NPV | Net Present Value |
| O&M | Operations and Maintenance |
| OEM | Original Equipment Manufacturer |
| OO | Object Orientation or Object Oriented |
| PC | Personal Computer |
| PERL | Practical Extraction and Reporting Language |
| PERT | Project Evaluation and Review Technique |
| PMBOK | Project Management Body Of Knowledge |
| PMI | Project Management Institute |
| QA | Quality Assurance |
| R&D | Research and Development |
| RADIUS | Remote Authentication Dial In User Service |
| RF | Radio Frequency |
| RFID | Radio Frequency Identification |
| RFI | Request For Information |
| RFP | Request For Proposal |

| | |
|---|---|
| RFQ | Request For Quotation |
| SCM | Software Configuration Management |
| SDL | Specification and Description Language |
| SEI | Software Engineering Institute |
| SFIA | Skills Framework for an Information Age |
| SIP | Session Initiation Protocol |
| SMS | Short Message Service |
| SOC | Standard Occupational Classification |
| SWEBOK | Software Engineering Body Of Knowledge |
| S/W | Software |
| TCL | Tool Command Language |
| UML | Unified Modelling Language |
| UMPS | Universal Mobile Telephony Service |
| VB | Visual Basic |
| XML | eXtensible Markup Language |

# FOREWORD

EILEEN M. TRAUTH,
PENNSYLVANIA STATE UNIVERSITY

In these difficult economic times it is more important than ever to strengthen the connection between academic preparation and workplace skills. This is particularly the case in the globally competitive field of software development where the stakes have never been higher. The ability to educate and retain a software development workforce is no longer just an issue for educational institutions and the software industry. For countries such as Ireland, whose economies depend heavily on a robust information economy, it is a matter of national importance. In this well written and highly accessible book, Jack Downey makes a unique contribution to the software development literature. He does this by addressing a clear gap in the IT skills and knowledge literature. Whereas I and others have been writing for some time about the skills and knowledge needed in the information systems profession, little systematic research has been carried out in the software development field. In response to this need his book details the research that was conducted in order to better understand the connection between the various software development roles and the skills needed to successfully carry them out.

There are several important audiences for this work. Clearly, members of the software industry would benefit from reading this book. In particular, managers who have responsibility for staffing projects or recruiting new personnel would want to read it. The artefact-centric framework that he developed can be of use not only in the recruitment process. It can also facilitate an overall understanding of the various roles carried out by members of a software development group in the course of enacting a project. In doing so, this framework provides a useful mechanism for determining the skills that are needed to develop software. It also serves to highlight the skills that every team member needs – to communicate, collaborate and to support the decision-making process. Contrary to many stereotypes about the nature of work in the information technology professions, this study reinforces the need for software developers –

particularly those in management roles – to posses both the ‘back end’ technical skills and the ‘front end’ interpersonal skills needed for customer interaction, requirements analysis and domain understanding.

But in one of the most distinctive characteristics of this book, Jack Downey has reached out to an audience that is atypical of software development writing: national policy makers. While some may find this unusual, those of us who have studied the connection between the development of an information economy and the economic development of a country know precisely why he addresses this audience. And there is no better country in which to do this than Ireland. As I explained in my 2000 book, *The Culture of an Information Economy: Influences and Impacts in the Republic of Ireland,* the case of Ireland was one of the first to document the connection between the growth of an IT sector and the increase in economic prosperity. While my book traced the policies and processes put in place over a thirty year period to bring about Ireland’s Celtic Tiger, Downey’s book addresses the issue of sustainability in the current economic climate. Whereas Ireland had little competition in the 1970s when it first began to attract multinational IT firms, that is not the case at present. In this highly competitive and global information economy of the twenty-first century, Ireland must constantly strive to stay ahead of the trend that moves software development work to lower wage countries.

For these reasons Downey’s critical assessment of the skills needed by the software development profession is not just a matter for academics to consider. It should be taken to heart by policy makers as well. His interviews with senior software developers reveal a dilemma for educators and policy makers in Ireland. The informants indicated that little or no business skills are required, and -- because they work mainly on mature products – neither are high-end research skills. But if Ireland’s software sector is to be sustainable, it needs to provide more value, Downey argues. Irish software facilities need to engage with *research* as well as development. By identifying the lack of roles for people with business/entrepreneurial and research skills, this study introduces a factor that may have escaped notice by policy makers: how to gain the skills needed for the next generation of software development when such environments are few and far between. Thus the implications of Downey’s critical assessment of the roles and skills of software developers are significant beyond industry and academe. Perhaps the best reason to read this book is for this connection. One of the factors that accounts for Ireland’s success in creating an information economy was the way the university curricula adapted to the skills and knowledge required of those who sought employment in the multinational IT sector. Jack Downey’s findings argue that continued cooperation among the multiple stakeholders – academe, industry and policy makers – is as important now as it ever was.

# CHAPTER ONE

# INTRODUCTION

Commercial software development is significantly more complex than programming as a hobby (Weinberg 1971). When writing a program for personal use, the amateur programmer understands the requirements and will not have to explain to third parties how the program works. Similarly, if the program contains faults, the author decides if they are worth correcting or not. In contrast, commercial software development is dominated by the need for communication. For instance, during his studies of TRW, Barry Boehm observed that roughly two-thirds of the time spent in a development project leads to documentation, while only the remaining third produces code (Boehm 1981; Boehm, Penedo, Stuckle, Williams and Pyster 1984). Similarly, during his days in IBM Fred Brooks (1972) noted that only about one-sixth of the development activity is coding. He also discovered that half of the development time is taken up in various levels of testing. A factor that Brooks does not consider in his breakdown is maintenance, which Boehm (1981) estimates takes up around half the development lifecycle but, according to Hatton (1998), makes up four-fifths. So, comparatively speaking, the low-level design, coding and unit testing activities – what McConnell (1993) calls software construction - is a small fraction of a commercial software development project.

This suggests that the other activities, such as software testing (Roper 1994; Myers 2004) and maintenance (Swanson 1976) or evolution (Lehman 1998) should command more attention than they do. Even so, bearing in mind Dijkstra's point that: "Testing shows the presence, not the absence of bugs" (Dijkstra 1969, p.66), a comprehensive testing strategy will not address the fundamental goal of computing science: "How not to make a mess of it" (Dijkstra 2001, p.92).

Because software is inherently complex (Brooks 1986), often needs to satisfy disparate customers (Birk, Heller, John, Schmid, Masser and Muller 2003) and is regularly expected to meet 'hard' deadlines (Miranda 2002), not making a mess of it - i.e. developing software on time and without error - is easier said than done. Moreover, because computers are becoming more pervasive, flawed software can have tremendous consequences (Colwell 2002). Also, many projects never get to completion (DeMarco and Lister 1999) and some should never even have been started because they were either technically or commercially infeasible (Boehm and Huang 2003).

Why is software development so error-prone and time-consuming? Boehm (1981) makes the point that the cost in terms of time and effort of repairing an error increases by a factor of ten from one phase of development to the next. This suggests that attention to the earliest phase, requirements, would pay dividends. However, as noted in Norris and Rigby (1992), “the time and effort devoted to [requirements definition] is usually minimal” (p.47). This leads to the situation where, “in any development effort, the requirements make explicit some – but only some – of the desired properties of the final system” (Bass, Clements and Kazman 2003, p.6). Turski (1986) describes the essential difficulty of software as having to work with one formal system – the hardware – and one informal one – the application domain. “Strictly speaking, a description of non-formal domain properties cannot be proven” (p.1078), therefore software developers can never guarantee that their solutions totally address the given problems.

However, the software industry seems to prefer devoting its time to managing the software development process than improving its requirements elicitation process. For instance, despite the fact that each project is unique (Turner and Muller 2003) efforts are being made to standardise the software development process, an example being the Capability Maturity Model (Carnegie Mellon Software Engineering Institute 2008). Alternatively, it is argued that reducing the process (and documentation) overhead will lead to shorter development cycles and quicker responses to changing requirements. Thus, various ‘agile’ management techniques have emerged, such as Extreme Programming (Beck 2000) and SCRUM (Schwaber 1997). Although regarded as a new step in project management, Larman and Basili (2003) argue that these are simply forms of ‘iterative and incremental’ developments that trace their roots back to the 1970’s. While these are very popular with developers, they appear to be most effective on small projects (Boehm and Turner 2003) or non safety-critical ones that have volatile requirements and which are developed by small, skilful co-located teams (Williams and Cockburn 2003).

No matter what management approach is used, the development of commercial software products requires the skills of a multi-disciplinary team. Between them, the team members must determine what the customer wants, decide on the most cost-effective way to do it, plan the steps needed and produce the end product - a deliverable comprising tested software, user manuals and training materials. For larger systems, the software must be installed on site and the end-users trained in its operation and maintenance.

Therefore, in addition to the detailed technical skills needed to develop computer programs, members of the team must have customer-facing skills, business skills, management skills and general communications skills that allow the team members interact effectively.

With the emergence of China and the Andhra Pradesh region of India as software development forces, as well as the accession of Eastern European states to the European Union (EU), Ireland's software industry is under threat from lower-wage economies. The Irish government is concerned about this and has commissioned one of its advisory bodies (Forfás) to investigate the problem. Their report recommends that the various industrial sectors, including information and communications technology, add more value to whatever they do by developing new skills (Forfas 2003).

Although the Irish government must ensure that the workforce has sufficient skills to attract investment and, by doing so, create employment, the skills needed to develop software are of concern to many parties: Universities want to make certain that their graduates are equipped to obtain employment in industry. Practitioners need to have the necessary skills to pursue careers in the software industry. Professional bodies strive to transform software practitioners into true professionals having the characteristics identified by Ford and Gibbs (1996); one of these characteristics being the continuous development of professional skills. Finally, industry requires a wide range of skills in order to develop its products; it would like to acquire these skills as economically as possible.

Extensive skills research has been carried out in the information systems (IS) arena, although such research has not been done in the software engineering discourse. While it is useful to learn from the IS research, it must be noted at the outset that there is considerable confusion over the definition of IS (Kaarst-Brown and Guzman 2005) and the work done in the IS area may not have direct relevance for commercial software development.

Therefore, while the literature cannot list the skills needed to develop commercial software, IS skills research may provide useful guidance, as shown in the following examples:

For educational institutions, charged with developing relevant curricula in the changing world of information systems, efforts have to be made to bridge the gap between what skills the institutions are equipping their graduates with and what industry needs from its staff (Trauth, Farwell and Lee 1993). A problem for the institutions is that recruitment adverts emphasise the required technologies, but final selection is based on the person's communications or 'soft' skills (Litecky, Arnett and Prabhakar 2004).

Trauth et al (1993) consulted IS practitioners in industry to determine what skills employers are looking for. Respondents recommended that university programmes should place greater emphasis on 'real world' experience, communication skills, analytical ability and problem solving. They went on to note that companies need high quality people with general intellectual depth, solid inter-personal skills and some functional business knowledge. In a later report from this study, these researchers identified the importance of business functional knowledge and inter-personal and management skills  (Lee, Trauth and Farwell 1995):

Another study motivated by IS curriculum design was carried out by Bailey and Stefaniak (1999). They divide their skills findings into three headings: 'soft', business and technical. The authors place particular emphasis on listening skills, whose importance is emphasised in the literature. For instance, Nichols and Stevens (1957) note that "business is tied together by its systems of communication. This communication, businessmen are discovering, depends more on the spoken word than it does on the written word; and the effectiveness of the spoken word hinges not so much on how people talk as on how they listen" (p.85). The general impression from Bailey and Stefaniak's (1999) work is that technical skills are easier to teach than non-technical ones. Indeed some IT companies would prefer to hire people with minimal technical skills so long as they can demonstrate solid soft and business skills.

In Singapore, Shi and Bennett (1998) draw on earlier work and use four knowledge (general IS, organisational overview, organisational unit and IS product) and two skills (organisational and technical) categories in their IS skills survey:

One IS study, (Misic 1996), focuses on the skills needed by systems analysts. Misic reaches the conclusion that strong technical skills are not necessarily the most useful ones for analysts to have. He predicts that tomorrow's analysts may come from business disciplines rather than from the programming pool, citing how some end-users are proving very adept at designing and developing their own PC-based systems.

In the Irish context, Forfas (2003) recommends that courses teaching industry knowledge, technology entrepreneurship, team working, team leading, project management and understanding the customer perspective be added to software-related curricula.

To summarise, the key skills required in the IS arena are: technical, business, interpersonal and the application of technology to business problems - which leads me to ask: are similar skills needed by commercial software practitioners?

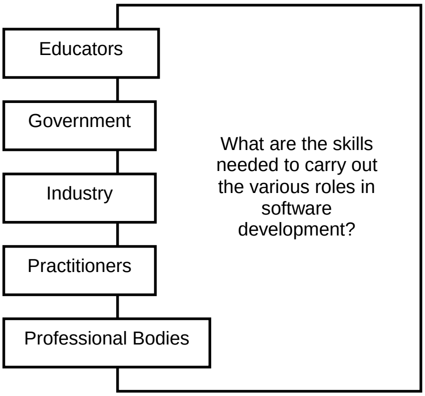
Judging by the wide range of activities involved in commercial software development, it is expected that an eclectic range of skills is also required in this field. Because software is developed by multi-disciplinary teams, then the necessary skills are likely to be distributed among the team members. This conjecture is supported by Weinberg (1971) who observes that not everyone on the team needs to have all the skills. If this is the case, how are the skills allocated across the different roles seen in software development teams?

When the literature considers roles, it tends to focus on a single role at a time and, generally, its concern is with what makes a good exponent of that role. So, for instance, while Misic (1996) determines the skills employed by systems analysts, other studies look for the characteristics of a good systems analyst (Vitalari and Dickson 1983; Schenk, Vitalari and Davis 1998; Wynekoop and Walz 2000), a good project manager (Cheng and Dainty 2005) or a good software engineer (Turley and Beiman 1995).

While this is significant research - highlighted by DeMarco and Lister's (1999) assertion that the best software people outperform the worst by a factor of ten – the literature does not seem to consider other roles in the software development team. Customer-facing roles, such as product management and customer support, are not studied; neither are the training and documentation functions.

So, what are the skills needed to carry out the various roles in software development? This is an important question to ask as it affects all the stakeholders listed in Figure 1. The educators need to know what skills their graduates will need. The government will want to draw attention to the high-value skills that the country boasts in order to attract investment. Practitioners are interested in positioning themselves to best advantage. Professional bodies will want to develop their members' careers so that they tailor their skills to the areas in most demand. Finally, industry will only continue to trade in a country if it offers the skills a company needs. Considering the increasing competition for foreign direct investment, this question is of particular importance in an Irish context.

Figure 1: Skills Stakeholders


Educators
Government
Industry
Practitioners
Professional Bodies
What are the skills needed to carry out the various roles in software development?

Research has been done on the skills required in the software development sector, but the tendency has been to group all the roles under one heading. For instance, the British Office for National Statistics places team leaders, systems architects, software developers and testers under the same heading – standard occupational classification code 2132 (Office for National Statistics 2000b). Similarly, Irish studies (that make use of the British codes) offer recommendations for the Information and Communications

Technology (ICT) sector as a whole (Forfas 2003; Enterprise Strategy Group 2004).

Professional bodies have also avoided role distinctions. The Irish Computer Society (2008) is promoting the use of the Skills Framework for an Information Age (SFIA Foundation 2008). This framework provides a list of skills and asks practitioners to assess themselves on each one. They can then plan their development by improving existing skills or by gaining new ones. However, what constitutes an ideal skills profile for a particular role is not discussed.

In summary, there are noticeable gaps in the literature. Firstly, skills surveys have not been carried out in the commercial software development industry. The skills surveys carried out by IS researchers have revealed the need for technical, business, interpersonal and the application of technology to business skills. Because the IS environment mainly concerns the development of software solutions for in-house use, the skills identified for this sector may not be appropriate in the commercial software industry. Secondly, there are no satisfactory role definitions for software development practitioners.

Being able to identify the skills needed for particular roles will allow stakeholders to target their efforts effectively. By gaining insights into what the key roles in the software development process are, academic courses can be targeted to produce graduates capable of these roles. For instance, how do testers and customer support people currently prepare for their positions? Indeed, for any given role in a software development project, how do people advance into senior grades – what sort of career progression can a new graduate look forward to?

The remainder of this book will describe a research project, based in Ireland, that set out to answer the question: what are the skills needed to carry out the various roles in software development? The insights gained through this research will contribute to the software engineering literature by addressing skills needs and role definitions in the software development industry. It is hoped that the research will also be of benefit to Irish policy makers.

# CHAPTER TWO

# REFINING THE QUESTION

## 2.1 Introduction

What exactly does the research question mean? It seeks to equate skills with various roles. But what roles need to be considered and what precisely is a skill? The next sections will explore these ideas.

## 2.2 Selecting Roles

A first step is to decide what software development roles to concentrate on. The roles chosen for study should reflect the software development lifecycle, so that the people involved in bringing a product from initial requirements elicitation through to final product installation and maintenance are considered.

Capers Jones (2003) notes that, the bigger the organisation, the more distinct and specialised the roles. In his survey, he identifies a total of thirty roles in software development. However, many of these roles seem like specific tasks within a larger role, suggesting that a smaller organisation would merge some of these responsibilities. Table 1 shows a possible merging, one where the roles reflect the entire development lifecycle.

Table 1: Combining Roles

| **Role** | **Activities** |
|---|---|
| Architect | Cost analysis<br>Cost estimation<br>Function point analysis<br><br>$\left.\begin{array}{l}\text{Human factors}\\\text{Performance analysis}\\\text{Reliability}\\\text{Reusability}\\\text{Security}\end{array}\right\}$ Non-functional requirements<br><br>Software engineering<br>Systems analysis<br>Education & training (preparing material for customers) |
| Programmer | Configuration control<br>Networks<br>Integration<br>Librarians<br>Data administration<br>Database design<br>Data quality |
| Project manager | Metrics and measures<br>Project planning<br>Quality assurance<br>Standards |
| Technical writer | |
| Tester | |
| Customer support | |
| Product manager | Assists in cost analysis and estimation<br>Assists with systems analysis (domain knowledge)<br>Globalisation |

These roles can also be seen in the Standard Occupational Classification (Office for National Statistics 2000b):

Table 2: Work Areas

| **Role** | **SOC Code** | **Sample Job Titles** |
|---|---|---|
| Architect | 2132 | Architect, systems (computing) |
| | 2131 | Architect, software |
| | 2132 | Designer, software |
| | 2132 | Engineer, design, software |
| | 2123 | Engineer, staff, telecommunications |
| | 2132 | Programmer, analyst |
| Programmer | 2132 | Developer, software |
| | 2132 | Developer, systems |
| | 2132 | Engineer, development, software |
| | 2132 | Engineer, software |
| | 2132 | Engineer, software, senior |
| | 2132 | Engineer (professional, software) |
| | 2132 | Programmer, computer |
| | 2132 | Programmer, systems |
| Project manager | 1136 | Manager, development, software |
| | 1136 | Manager, development, systems |
| | 1136 | Manager, project, development, software |
| | 1136 | Manager, software |
| | 1137 | Manager, development (*research and development*) |
| | 2132 | Leader, project, software |
| | 2132 | Leader, project (software design) |
| | 2132 | Leader, team, software (computing) |
| Technical writer | 3412 | Author, technical |
| | 3412 | Writer, technical |
| | 3412 | Writer, technical, senior |
| Tester | 2132 | Tester, software, computer |
| Customer support | 2131 | Consultant, support, software |
| | 2131 | Consultant, support, technical |
| | 2131 | Engineer, support, software |
| Product manager | 1137 | Manager, product (*research and development*) |

As noted in chapter 1, most of the research into roles concentrates on a single role at a time and the characteristics needed to be effective in that role. As DeMarco and Lister (1999) point out, the best developers outperform the worst by 10:1; they outperform the median by 2.5:1. These studies find that skill is only one factor in determining performance.

For instance, Turley and Bieman (1995) attempted to determine what differentiates good and bad software engineers. Their findings show that experience is an important factor and "that personal attributes and interpersonal skills are most closely linked with performance differences. Skills associated with task or situation did not generally emerge as differential." (p.37) High performing engineers tend to have an appreciation of the 'big picture'. The researchers' work results in a list of thirty-eight competencies, which are classified as task accomplishment, personal attributes, situational and interpersonal skills.

Research in the information systems (IS) area also suggests that non-technical skills and personal attributes relate to performance. For instance, the systems analyst is a key player in IS developments. Several studies have been carried out to determine what characteristics and skills are needed to succeed in this role. In their early study, Vitalari and Dixon (1983) identify two problems with this task:

1. There are no well accepted job performance criteria
2. There is no understanding of how different skills relate to job performance

They interviewed a total of eighteen systems analysts, nine of whom had been identified by their managers as effective and the remaining nine as poor. The hypothesis they sought to test was: that effective analysts will have different problem solving behaviours to less effective ones. To a large extent their argument is substantiated and they conclude that highly-rated analysts demonstrate the following types of behaviour:

- Analogical reasoning – trying to fit the problem into a previously experienced scenario.
- Planning and goal setting.
- Hypotheses management.
- Application of heuristics (rules of thumb).
- Good inter-personal relationship with the customer

Vitalari returned to this area in an effort to understand how problem-solving approaches differed between novice and expert systems analysts (Schenk et al. 1998). This research results in five potential reasons for error-proneness in novice analysts:

1. Less domain knowledge utilisation.
2. Superficial concern with user involvement.
3. Use of general versus specific triggers.
4. Less hypothesis testing and rejection.
5. Less goal generation.

A later study by Wynekoop and Walz (2000) uses Delphi analysis to identify twelve traits of high-performing systems analysts:

Table 3: Traits Associated with High-performing Systems Analysts

| **Knowledge or Skill** | **Personality Trait** |
|---|---|
| Can abstract business problems and translate into computer systems. | Is analytical and logical. |
| Is a creative problem solver. | Prefers challenging work. |
| Has good interpersonal and communications skills. | Is organised. |
| Has a high level of technical knowledge. | Is both dependable and reliable. |
| Has leadership skills. | Is team oriented. |
| Has a high level of business knowledge. | Is self-motivated. |

Those traits containing a knowledge or skill component can be taught. However, Hackman and Oldham (1980), warn us that management training has not been wholly satisfactory and it is unlikely that someone can be taught to be creative.

Systems analysts are not the only people to be assessed in this way. Although Cheng and Dainty's (2005) study concentrates on building construction project managers, when they categorise projects as being: "inherently unique, tend to be awarded at short notice, are reliant on a transient workforce and exist within a complex multidisciplinary team-oriented environment" (p.26), it can be seen that their findings should be relevant to software development project managers also. Like Vitalari and Dixon, these researchers compared good and average project managers