Software Architects: A Different Type of Software Practitioner

Jack Downey

Lero – the Irish Software Engineering Research Centre
University of Limerick
jack.downey[at]lero.ie

Keywords: Software POP-I.A skills; POP-V.B case studies.

Abstract

This paper reports on two studies into the skills required to develop software. In the first, senior software practitioners in four companies, across seven roles were interviewed. However, the architect role proved difficult to characterize, because architects did not seem to carry out any architecture work. It was hypothesized that this was because the five architects interviewed were all working on mature products. In a second study, five senior architects in a single company were interviewed and the realisation from this study is that, unlike most other software practitioners, architects are not so much project focussed as product and market focussed. This discovery suggests that focussing the interviews on projects rather than on products is inappropriate for architects.

1. Introduction

In the beginning there were programmers and systems analysts. Then, after the NATO conferences in 1968-69 (NATO, 1969), there were software engineers. Some scholars expressed concern about this development (Baber, 1982; Parnas, 1999), arguing that people without engineering backgrounds should not call themselves engineers. Later we began to see software designers and software architects. Should these new titles be regarded with similar cynicism? Are they just systems analysts re-branded for the 21st century?

But what is architecture and how does it differ from design? According to Bass, Clements and Kazman (2003), the "software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them" (p.3). However, Gilb and Finzi provide a much clearer insight – for a computer to provide functionality, to satisfy functional requirements, programs must be designed. However, to satisfy non-functional requirements – such as performance, scalability, maintainability, usability and security – an architecture is required (Gilb & Finzi, 1988). Thus architects who are involved in systems where non-functional requirements are critical are likely to include significant architectural aspects in their work.

Fortunately, it was possible to conduct two studies that shed light on the architect role. The first was a wide-ranging study that sought to characterize seven important software roles – product manager, project manager, architect, programmer, tester, technical writer and customer support personnel – in terms of skills.

The practitioners interviewed worked in four different companies. Two were Irish subsidiaries of large North American multinationals, one was an Irish subsidiary of a smaller, European multinational, while the forth was an Irish, indigenous company. Despite their corporate differences, all the architects interviewed referred to themselves as "systems architects". Another common factor was that all of the interviewees worked on mature products.

In the second study, five software architects in a single company that had several architectural concerns were interviewed. This company was at an interesting stage in its development. It began as a start-up and developed a suite of management and monitoring tools for a niche segment of a particular market. It had, a year or two prior to the study, been taken over by a large multinational, whose goal was to establish a strong presence in this market. The approach the corporation took was to acquire a set of companies who were servicing the various niches. The result was ownership of a portfolio of products, with similar functionality, targeted at the different niches.

However, each company had approached the problems in diverse ways and the architectures they used were radically different. For the architecture team, the ultimate aim is to create a suite of products with the same look and feel and reporting mechanism.

Another concern created by the takeover is scalability. The multinational has a very respected name and a considerable sales and marketing force. This is leading to larger customer installations, where the demands made on the products have increased by orders of magnitude.

Therefore, although the products are mature, these architects have to deal with significant architectural issues – specifically merging disparate architectures and enhancing capacity and performance.

In contrast to the interviewees in the first study, those in the second used the title "software architects" and not "systems architects", despite the fact that one person in particular – the benchmarking specialist – has significant involvement in hardware platform evaluations. Interestingly, the British Office of National Statistics categorises software architects as "IT Strategy and Planning Professionals" and systems architects as "Software Professionals" (Office for National Statistics, 2000, p.77), suggesting that systems architects should be the people with more technical input. However, this is more likely to reflect the general confusion in the industry in relation to roles and job titles (Belbin, 2000; Dierdorff & Rubin, 2007; Downey, 2006a; Shaw, 2000) rather than being a significant result in itself.

The remainder of this paper will present the research methodology, the findings of the architect interviews in the two studies and will conclude by reflecting on the insights gained in these studies.

2. Research Methodology

Data for both studies were gathered using a semi-structured interview instrument. This interview instrument was informed by social cognitive theory (Bandura, 1986) and was instrumental in developing an artefact-centric framework that proved useful in classifying software skills (Downey & Power, 2007).

The first study, where five systems architects across four companies were interviewed, sought to understand the skills needed to develop software. Thus, senior people from the main software team roles were studied in order to obtain this information. A grounded theory approach was used in the study (Strauss & Corbin, 1998) which implies that the interviewer must gather information without having a pre-conceived hypothesis. Given that access to senior practitioners is difficult, how can an interview capture all relevant information, despite not knowing a priori exactly what that information is?

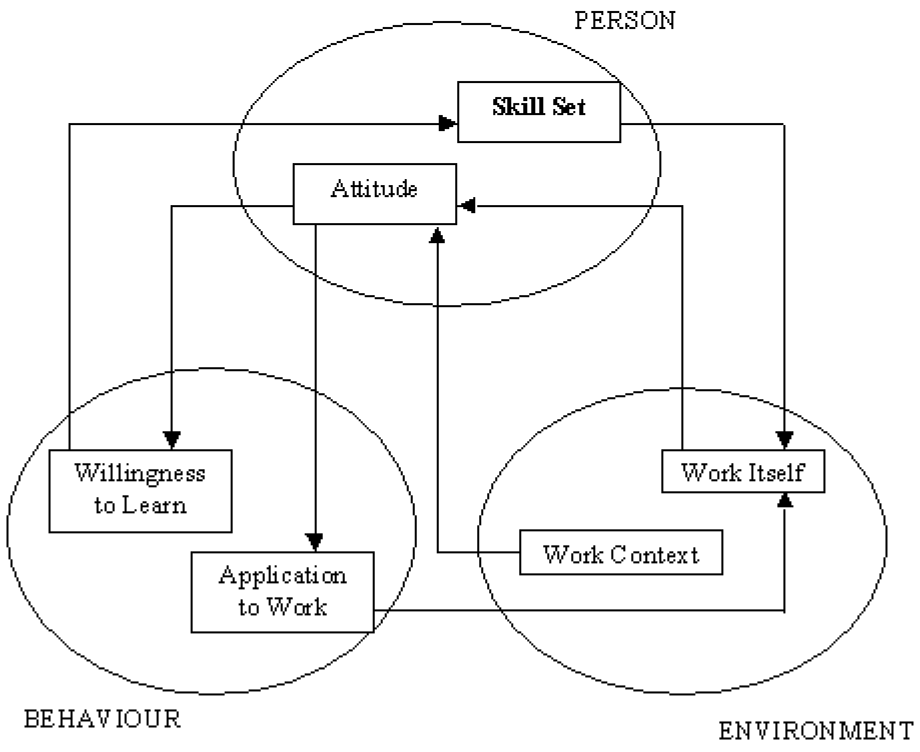

Figure 1- Social Cognitive Theory

Bandura's social cognitive theory provided the solution. According to Bandura, a person is described by their personality, their environment and their behaviours, all of which influence each other reciprocally (Bandura, 1986). Therefore, any interview instrument that seeks to elicit skills information needs to be informed by personal characteristics (such as attitude and skill set), the work environment (that is: the work itself and the context in which this work takes place) and the person's behaviour (their attitude towards work and their willingness to learn, for instance) – see Figure 1.

This triadic theory was now used to develop an interview checklist that contained three overall questions. The first: "what do you do?" asked the interviewee to consider their job in terms of a typical project – when did they get involved in the project, when did they leave and what project phases did they contribute to? The second: "what skills do you have?" was informed by existing skills research (Lee, Trauth, & Farwell, 1995; Noll & Wilkins, 2002; Sawyer, Eschenfelder, Diekema, & McClure, 1998; Shi & Bennett, 1998; Trauth, Farwell, & Lee, 1993). The interviewee was asked to consider each of the listed skills and indicate how relevant this skill was to them. The skills were broadly categorised as technical, business and inter-personal. The final question was biographical in nature: "how did you acquire these skills?" Here the person's early career motivation was explored as well as determining how extra-curricular activities – such as team sports – contributed to their skill set.

A total of thirty-five senior software practitioners – architects, project managers, product managers, programmers, testers, technical writers and customer support people – were interviewed in the first study and the outcome of the research was an artefact-centric framework. The framework is based on two observations: The first is that artefacts provide the focus for communication between the different roles and the second is that all the development artefacts – documents and code – experience a similar life-cycle.

For every artefact, someone has to commission it and provide its requirements. Then the person assigned the task of creating the artefact must analyse the problem to be solved and may need to consult other artefacts or other stakeholders to clarify understanding. In some cases, intermediary artefacts must be created – such as prototypes or scenario models – to gain the necessary insights. Then a solution must be found – that is, new knowledge must be synthesised and that knowledge disseminated to and reviewed by the team. The feedback from the review may trigger another development cycle, leading to version 2.0 of the artefact (Downey & Power, 2007).

This new understanding of artefacts informed the interview instrument which was adapted for the second study. As Eisenhardt notes (1989), grounded theory "relies on continuous comparison of data and theory beginning with data collection. It emphasizes both the emergence of theoretical categories solely from evidence and an incremental approach to case selection and data gathering." (p.534) Thus, it makes sense to review the research process at logical points during the study and to adjust the instruments on the basis of the emerging picture.

Thus, for the second study, the overall three questions still remain, but the checklists within the first two questions have changed. The interviewee is still asked to describe a typical project, but now, instead of focussing on phases of the project lifecycle, the interviewer seeks to learn more about the artefacts involved.

Similarly, the artefact-centric perspective informed the second question and the skills taxonomy probed by "what skills do you have?" now is divided under the headings: communication, collaboration, decision-support and technical. Essentially, every artefact draws on the first three types of skills, whereas the technical skills are very specific to particular artefacts.

3. Findings

Miles and Huberman (1994) describe the first stage of qualitative analysis as 'data reduction', where the large volume of data is simplified and focused. Thus in both studies, each interview was studied in detail and analysed sentence by sentence. Short, descriptive terms called codes were created and related sections of the interviews were filed under these codes. This basic level of analysis is called 'open coding' (Strauss & Corbin, 1998). The interview data were entered into the QSR Nvivo software package. This kept track of the assigned codes and allowed all interview data associated with a given code to be viewed.

3.1 The First Study: Five Architects, Four Companies

The five architect interviews in the first study yielded a total of sixty-four codes. Of these, all of the architects only agree in six. That is: all have bachelor's degrees in engineering; none of them manages a budget; all have interviewed candidates, which can be attributed to their having had some type of leadership experience; all have to give presentations and have received training in this; all review the preliminary, or marketing, requirements documents from the product management / marketing function and, finally, all have reservations about the benefits of short professional training courses.

From these codes, we get a very limited picture of the systems architect. They are all experienced people, whose careers are based on an engineering foundation. They all take part in the requirements definition phase. Since they do not manage budgets, it is unlikely that they carry out a formal management function.

In order to build up a better picture of what a systems architect is, a further level of coding was needed. It was decided to examine the remaining codes using the method of triples. This works by analysing a set of elements (for instance, interview data classified by a particular open code), three at a time, and identifying the odd one out. Having identified the odd one out, we need to know, not only why this is unique, but why the others are similar. These contrasting statements together make up what is called a 'construct'. For instance, all the architects had to communicate with customers – but some met them face-to-face while others had to work through the project management function. This form of analysis is also used in the repertory grid technique (Stewart, 1997).

This analysis formed the basis for a definition of the systems architect (Downey, 2006b, pp.218-219):

> "The systems architect is a technical expert with experience of the company's products and the industry in general. S/he works with product management to refine customer requests into a set of engineering requirements that are possible to implement. S/he will identify possible implementations that satisfy the requirements and assess their feasibility in terms of time

> schedules and headcount. Architects sometimes remain with a project through construction, writing the functional specification and contributing to the design, code and test activities as a technical lead, an advisor or in a hands-on development capacity. After product delivery, architects may provide additional consultancy as well as training to end-users."

The definition is also informed by the significant overlap between the different roles discovered in the first study. For instance, some of the skills required by the architects relate to the background of the product managers they work with. If a product manager comes from an engineering background, s/he normally could answer technical customer questions. However, if the product manager is from a business or sales background, then the architect would have to attend customer meetings to address technical concerns.

Another finding of the first study comes from the architect interviews themselves. None of the architects seems to carry out much in the way of architectural work. Indeed, the only tangible mention of architecture relates to the assignment of software processes to computer platforms in a multi-host system.

> "If you have new services and functionality to introduce, you put them on different boxes. So the architecture wouldn't really change. Architecture is how the boxes are organised – what's running on the different boxes. How you manage whether they're highly available, or whether they're fault tolerant or whatever. … For software architecture, you'd identify where different processes would sit."

Indeed the role seemed to have more in common with Misic's definition of a systems analyst than with an architect.

> "A systems analyst is a problem-solving specialist who works with users and management to gather and analyze information on current and/or future computer-based systems. With this information, the systems analyst, working with other MIS personnel, defines the requirements which are used to modify an existing system, or to develop a new system. The systems analyst identifies and evaluates alternative solutions, makes formal presentations, and assists in directing the coding, testing, training, conversion, and maintenance of the proposed system." (Misic, 1996, p.35)

It was hypothesized that this situation prevails because the interviewees were working on mature products, where the architecture has been developed already.

3.2 The Second Study: Five Architects, One Company

The first insight gleaned from the second study is that architects can specialize in different areas. In this architectural team, architects cover four different areas: The company offers two distinct products and each one has an architecture group assigned to it; a third group is concerned with integrating the products into the parent corporation's portfolio and the fourth deals with benchmarking and performance issues. The architecture group's manager agreed to be interviewed and selected one member from each of the four teams as well.

New product releases in this company are driven by the need to provide new features. Suggestions for new developments come from customers, marketing people, product managers and also from the architects. It is a tribute to the marketing research being done by the corporation that the customer-facing roles seem to have advance notice of most customer features.

> "Most of the time the customer and marketing come up with exactly the same requirements and features. All the customer-raised enhancements were actually raised by marketing or product management about a year ago, or more than that. So we know what the customers want usually."

Besides having a good idea of what functionality will be expected, the product is also constrained by the corporate marketing strategy. The plan is to merge products from the acquired companies into a single, coherent architecture. It is also hoped that these products will cope with loads far in excess of their original requirements. Thus, the architecture team has a good idea of what direction the product has to take. In order to move the product gradually in the right direction, the architects will propose certain “features” for the next release. Such architectural features may not add any new functionality but might remove some legacy code or restructure the product to make it more scalable, for instance.

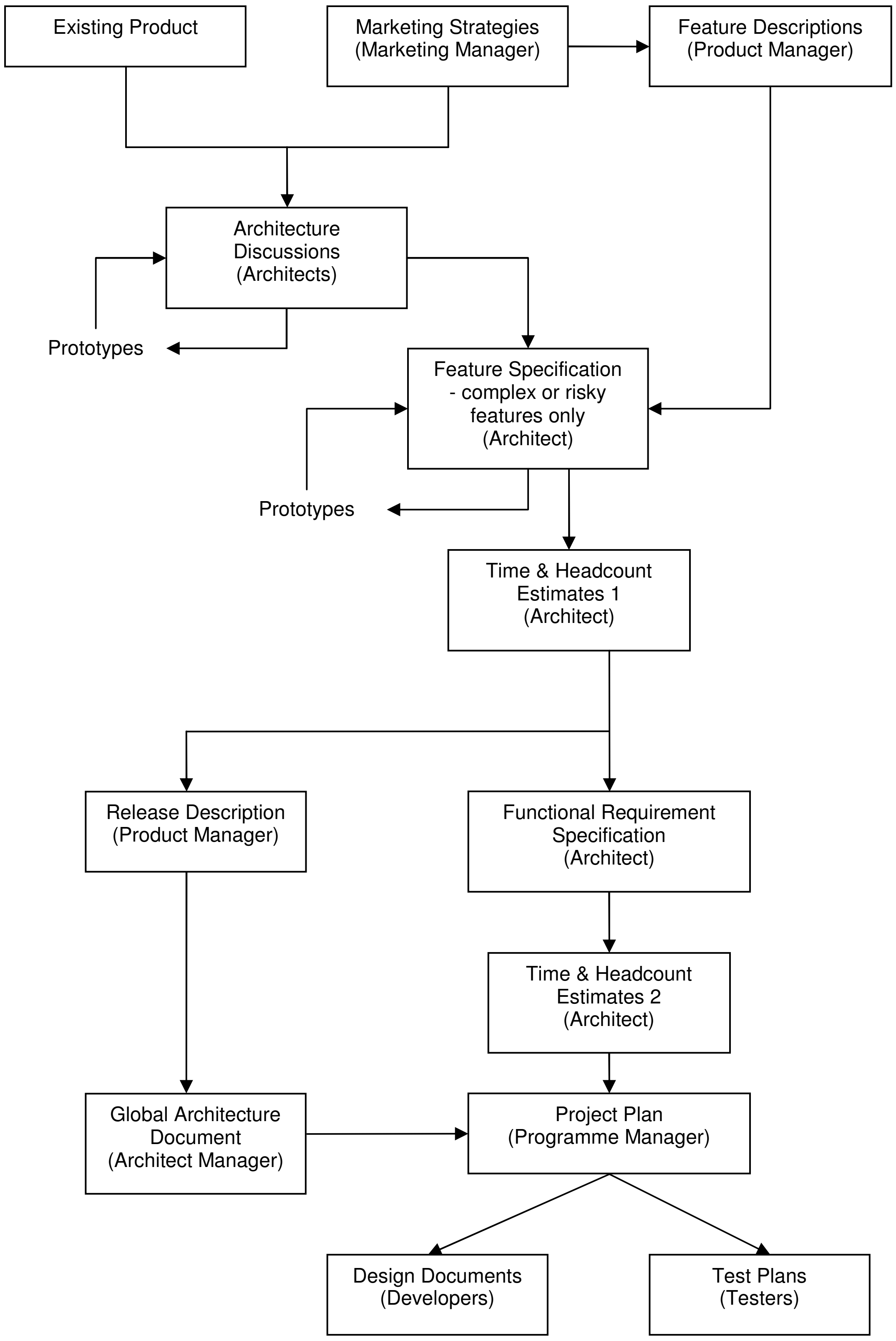
Existing Product
Marketing Strategies (Marketing Manager)
Feature Descriptions (Product Manager)
Architecture Discussions (Architects)
Prototypes
Feature Specification - complex or risky features only (Architect)
Prototypes
Time & Headcount Estimates 1 (Architect)
Release Description (Product Manager)
Functional Requirement Specification (Architect)
Time & Headcount Estimates 2 (Architect)
Global Architecture Document (Architect Manager)
Project Plan (Programme Manager)
Design Documents (Developers)
Test Plans (Testers)

Figure 2 - The Architects’ Artefacts

> "You influence the product manager and you get him to put features – you make him aware of the fact that he needs to put a feature on the roadmap for the next release, to cope with such-and-such."

Interestingly, the driving artefacts for the next release might not be tangible documents. The marketing strategies are often disseminated through presentations and the feature descriptions are discovered in conversation with the product manager, who works in the same building. This informality is unlikely to continue as the company adopts the new parent corporation's processes.

Whenever a new feature is proposed, a feasibility study needs to take place. The architects are called to provide evidence of technical feasibility. If this is a complex feature, or one that involves new technology, then a prototype must be developed. Some features suggest several competing solutions and architects might prototype each option before deciding on one. This work will be documented in what is called a feature specification (see Figure 2). If the feature proves to be technically feasible, then the architect will be called upon to produce initial time and headcount estimates.

The architecture group manager and the product manager will work together to define the next release. Two documents are produced: the product manager will create a release description, while the architecture manager produces a global architecture document. Features that are chosen for the next release are assigned a requirements number and studied further in order to produce a functional requirements document (FRS). This is a key document, containing a set of numbered sub-requirements. This specification will offer one solution, but will also contain the decision rationale for choosing that particular solution. The FRS will also explain the future directions this feature could take. The architects are constrained by delivery deadlines, so the most elegant solution might lose out to the one that is easiest to implement. Because it forms the basis for development and test, the FRS is closely reviewed.

> "Then we go through the formal approval process, which could be as many as fifteen people, or more than that. The reason being that there are many groups involved: There's obviously architecture, there's development and there's testing, documentation; there's different development groups that might be impacted; there's professional services or customer-facing people as well."

The extra information in the FRS contributes to a second round of estimations, before a programme manager is assigned to the release and work begins on the project plan. At this point, the architects in the product groups act as technical leads or consultants during the development of the features. However, the architects meet regularly to discuss the overall shape of the product. In this forum, the architects can raise their concerns informally.

> "People would actually just suggest that bit of the code here or this bit of the architecture, I just don't like. That won't cope with a bigger network or a bigger configuration and that's how we actually replaced two of the components."

> "For example, we know that we have the current loading process. It's not good and maybe not as fast as it could be. There are a few architectural points that we don't really like and I know they could be causing some performance issues."

If the concerns raised resonate with the rest of the group, the architect may be asked to develop a proof of concept prototype.

> "When a new feature is proposed, like a big feature and it requires something like a quite complicated architecture, we do a prototype and someone – usually one of my architects or myself – would go and decide to do a prototype."

Prototyping was highlighted by most of the architects as their key contribution. It requires architects to be able to learn new technologies quickly and produce a prototype that contains some 80% of the required functionality in 5% of the development cycle. This quick learning skill is particularly true for those architects involved in the overall integration effort. They must learn how the other products in the corporation work.

> "Because there are initiatives that are across a broad set of products in the portfolio, they will have downloadable software to use and maybe even worked examples. You're there downloading stuff and looking at it, understanding it, installing it, figuring out how it works."

For the benchmarking people, the key research they need to carry out is to investigate the latest hardware platform developments. The benchmarking architect interviewed for this study enjoys good relationships with the hardware vendors and gets access to the latest products. He has been impressed with the latest CMT (Chip Multi-Threading) architectures, where the dispatcher has been rendered in hardware - performance is now approaching mainframe levels. Each machine has its strengths and weaknesses, so the best systems feature a mix of machines – each running applications it is best suited to. Benchmarkers also carry out prototyping tasks, where they demonstrate proofs of concept. This work has been helpful in establishing price/performance ratios.

The benchmarking architects seem like specialised testers as first glance. Based on the FRS documents included in the next release, these architects prepare a benchmark requirements document. This draws from the feature descriptions and also from customer usage scenarios – different customers use the product in different ways. This leads to a benchmark test plan. The plan seeks to exercise the "pain points" through load (increasing demand on the system until it can no longer cope) and soak (running the system under load for several days to reveal cumulative effects, such as memory leaks) tests.

However, it seems a bit late in the day to determine if the performance and scalability of the product is adequate.

> "There is a certain inherent complexity about it though, because you honestly don't know whether assumptions that are valid for small things – it's a classic computer science problem: how does the computational needs of a system vary with the size of a data set and is that a constant factor? In many systems there's a shear point, beyond which the characteristics of the system change completely. If you jump, for example, from a million objects and scale up to 100 million objects, do you need 100 times as much hardware or do you need a million times as much hardware or, in fact, do you reach a point where you can no longer do the computations and you have to go back to the drawing board?"

It would be good if these architects could predict performance at an earlier stage. For this reason, the group is investigating the use of mathematics.

> "One of the things we're looking at at the moment is using mathematical models to try and predict complexity. Essentially, when you're looking at a network – at least the way it's usually envisaged – you're looking at an extremely large tree and I'm looking, at the moment, at seeing whether we can get any ideas from graph theory and predicting the complexity of something."

Having studied the system under load, the benchmarking team feeds back its findings to the sales people and the product managers. These recommendations will influence the next release of the product. This team also produces tools (and their associated user manuals) that can assist the sales effort. It has sizing tools, for instance, that can provide dimensioning information for customers – given a particular load, what amount of hardware is needed to support it?

The group is already seeing benefits. The architecture is changing to be more distributed and scalable. It has also fed back into the process so that document templates now contain prompts relating to benchmarking concerns.

In the earlier study, the architects essentially converted customer (or marketing) requirements into testable and measurable ones that could be implemented in software. They also determined the technical feasibility of an initiative, as well as providing time and headcount estimates.

While the product architects in this study carried out a similar role, the integration and benchmarking architects revealed a completely different set of concerns. For these roles, the ability to gather intelligence – how the other products in the portfolio work; what new hardware platforms are coming on stream – and to prove concepts through prototypes are key skills. Most significantly these architects deal with genuine, non-functional requirements – performance and scalability – and architecture concerns, such as distribution of state and merging of products.

Interestingly, a solid background in computer science does not seem to be required in these roles. Three of the interviewees have engineering degrees, while the others have backgrounds in geology and English literature. All were good at mathematics at school, with two of the interviewees having access to computers as children. Given that Graham (2003) argues that the best business analysts are never scientists or engineers and advises people concerned with requirements to broaden their interests, this eclectic mix of architects should not be a problem. Also, great emphasis was placed, particularly by the group manager, on the collegiate nature of the group, suggesting that the company is making best use of their diversity (DeMarco & Lister, 1999).

4. Conclusions and Future Work

These two studies have produced several insights into the systems/software architect role. The first study contributed two interesting observations: First of all, the architect works in a team setting and a full picture of the role emerges when placed in context within the team. The second finding – that architects carry out very little architecture work was unexpected. It was only discovered during the second study that this was because of a weakness in the interview instrument rather than the fact that the practitioners were developing mature products. Because the instrument focussed on project work, rather than product-related work, the interviewees in the first study were not prompted to explain the full scope of their role.

The second study, because of its emphasis on artefacts, allowed more time to be spent exploring the entire architecture role and two interesting findings have emerged as a result: Firstly, there are different types of architects – the second study introducing product, integration and benchmarking architects. Secondly, the architect role cannot be described in terms of project work. These practitioners have a product- and market-centric perspective. Their goals tend to be strategic and achieved over a sequence of software releases. Architectural changes to the product tend not to have immediate payback – they rarely enhance the product's functionality – but are vital to ensuring that the product is class-competitive and is able to support future demands.

Obviously, the second study is limited in that it reports on a single company. As Yin (1994) advises, multiple sources of evidence are important in establishing construct validity. As seen in the second study, benchmarking and integration architects provide a different perspective to product architects. However, are such divisions unique to this company? Each organisation has its own culture which is the result of shared experiences since its foundation (Schein, 1999). Therefore the study needs to include other organisations to see if an overall picture of the architect emerges. However, it must be remembered that case study research provides purposeful samples as opposed to statistically representative ones. "The power of purposeful sampling lies in selecting information-rich cases for study in depth. Information-rich cases are those from which one can learn a great deal about issues of central importance to the purpose of the evaluation, thus the term 'purposeful' sampling" (Patton, 1987, p.51).

Furthermore, the lessons from the first study should not be forgotten and the architects' co-workers should also be canvassed. Indeed, the theory underlying the interview instrument – Bandura's Social Cognitive Theory – stresses the importance of a person's environment, including the other team members.

However, the approach to further interviews needs to be changed to reflect lessons learned in the second study:

1 As mentioned in section 3, some of the insights found in the first study are due to the interplay between the architects and their co-workers. Indeed, many of the insights gained on the architect role were determined by interviewing other practitioners. Thus, to get the complete picture of the architects (and the artefacts they work on) the study must also include the architects' main collaborators – product managers, project managers, sales and marketing people and senior managers.

2 Focusing the first question – "what do you do?" – on a typical project means that the most critical of the architects' contributions are neglected. Traditional textbooks (e.g. Norris & Rigby, 1992) describe a development project as beginning with requirements and finishing with test. As shown in figure 2, the existing product and the corporate strategy for the product are also essential drivers for the architects' work. Instead of asking an architect what s/he does in terms of a typical project, a better opening would be to ask: "what is the architecture of the product (or products) you work on and how are they expected to evolve?" Then the architect can be asked how they get from the status quo to the evolved product – which is, essentially, the question: "what do you do?"

5. Acknowledgement

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

6. References

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

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

Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice* (2nd ed.): Addison-Wesley.

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

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

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

Downey, J. (2006a). *The Knowledge, Skills and Abilities Required for Telecommunications Software Development: An Artefact-centric Framework (PhD Thesis).* University of Limerick, Limerick.

Downey, J. (2006b). *Systems Architect and Systems Analyst: Are These Comparable Roles?* Paper presented at the 2006 ACM SIGMIS CPR Conference, Claremont, California.

Downey, J., & Power, N. (2007, 19-20 April, 2007). *An Artifact-centric Framework for Software Development Skills.* Paper presented at the ACM SIGMIS CPR, St. Louis, Missouri.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review, 14*(4), 532-550.

Gilb, T., & Finzi, S. (1988). *Principles of Software Engineering Management*: Addison-Wesley.

Graham, I. (2003). The Compleat Requirements Analyste. *IEEE Software, 20*(6), 99-101.

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

Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis* (2nd ed.). Thousand Oaks, California: Sage Publications.

Misic, M. M. (1996). The Skills Needed by Today's Systems Analysts. *Journal of Systems Management, 47*(3), 34-40.

NATO. (1969, 7-11 October 1968). *Software Engineering: Report of a conference sponsored by the NATO Science Committee.* Paper presented at the Software Engineering, Garmisch, Germany.

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

Norris, M., & Rigby, P. (1992). *Software Engineering Explained*. Chichester, England: John Wiley & Sons.

Office for National Statistics. (2000). *Standard Occupational Classification - Volume 2* (v6 ed.). London: The Stationary Office.

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

Patton, M. Q. (1987). *How to Use Qualitative Methods in Evaluation*. Newbury Park, California: SAGE Publications, Inc.

Sawyer, S., Eschenfelder, K. R., Diekema, A., & McClure, C. R. (1998). Corporate IT Skill Needs: a Case Study of BigCo. *ACM SIGCPR Computer Personnel, 19*(2), 27-41.

Schein, E. H. (1999). *The Corporate Culture Survival Guide*. San Francisco: Jossey-Bass.

Shaw, M. (2000). *Software Engineering Education: A Roadmap.* Paper presented at the 22nd International Conference on Software Engineering, Limerick, Ireland.

Shi, N., & Bennett, D. (1998). Requisite IS Management Knowledge and Skills Construct: A Survey. *ACM SIGCPR Computer Personnel, 19*(1), 3-19.

Stewart, V. (1997). *Business Application of the Repertory Grid Index*: Enquire Within Developments Ltd.

Strauss, A., & Corbin, J. (1998). *Basics of Qualitative Research* (2nd ed.). Thousand Oaks, California: Sage Publications, Inc.

Trauth, E. M., Farwell, D. W., & Lee, D. (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* (2nd ed.). Thousand Oaks, California: Sage Publications, Inc.