

Introduction

Over a decade ago, Studs Terkel wrote a now famous book called *Working*. It has recently been reprinted (Terkel, 1997). In the book, dozens of people take a chapter each to talk about their daily experiences of their jobs . . . as cab drivers, engineers, nurses, secretaries, call-girls . . . a wide canvas! Studs allowed people to talk into his tape recorder . . . no steering, no prompting, no interruptions . . . at least that's how it seems to the reader. Almost pure stream-of-consciousness stuff. When I first read *Working*, I thought "That's a neat idea. Write a book by getting other people to do all the work! No analysis, no interpretation, no conclusion . . . just get people talking, and write it all down."

This book is a bit like *Working*. Most of it consists of people talking about their work . . . but in a structured way. And I do have some analysis, and a lot of interpretation. Like Studs, I don't try to come to firm conclusions.

The people I get talking are not cab drivers or nurses. They are men and women who manage information systems development projects for external clients. They work in software houses, IT consultancies, or whatever. They act as problem diagnosers, system designers, system implementers for their clients. In the words of the trade, they are *IT solution providers*.

What do they talk about in this book? They talk about their "recipes" for managing projects . . . particularly their ways of handling potentially serious problems like unrealistic client expectations, lack of real project ownership, and the like. In fancier language, they talk about their *theories-of-action*.

Argyris and Schön (1978) invented the term *theory-of-action*. They define a theory-of-action thus: "A full schema for a theory-of-action would be as follows: in situation S, if you want to achieve consequence C, under assumptions a, . . . n, do A." (10). So, this book is about situations, actions, consequences and assumptions in managing IS/IT projects . . . as seen by experienced IS/IT project managers (PMs).

Another way to describe this book is in terms of Gluch's notion of a *project-risk-element* (Gluch, 1994). A project-risk-element consists of three components: a risk condition, a default project transition, and a default

project consequence. A risk condition is an aspect of the current state of the project that gives cause for concern. A default transition is the likely sequence of states the project will move through if no action is taken to deal with the risk condition. A default consequence is the likely ultimate adverse outcome resulting from the default transition. This book is about IS/IT project-risk-elements, and the strategies that experienced PMs claim to use to cope with these.

The task of eliciting peoples' theories-of-action is not at all straightforward. Schön discusses this problem in the context of how professionals such as consultants, engineers and managers think in action: "When a practitioner displays artistry, his intuitive knowing is always richer in information than any (external) description of it. Further, the internal representation strategy, embodied in the practitioner's feel for artistic performance, is frequently incongruent with the strategies he/she uses to construct external descriptions of it. Because of this incongruity, for example, people who do things well often give what appear to be good descriptions of their procedures which others cannot follow. Everyone who has tried to learn from a book how to ski or write a story knows how difficult it can be to act from such a description" (Schön, 1983: 276).

Even if the expert is willing and able to explain "how he/she does it", there is the question of the validity of the information being volunteered. "... activity is context bound and to study any human activity divorced from its context is to study it divorced of its meaning" (Bell and Hardiman, 1988: 57). In other words, the "algorithm" elicited from an expert in an artificial setting, or while the expert is working on a contrived task, may not be the algorithm used "in the real heat of battle".

Not encouraging! So, how did I set about the task?

I broke the study into three phases. In Phase 1, I tried to answer the question: What factors do experienced IS/IT project managers take into account when deciding on how to "shape" and manage new projects for new clients? By factors, I mean things like the characteristics of the system to be built, the nature of the client organization, the sorts of people involved, the technology to be used, the politics of the situation . . . whatever. In Argyris and Schön's terms, the variety of (*S*)ituations to be dealt with. In Gluch's terms, the variety of *risk-conditions* to be removed or managed around. To answer this question, I interviewed fourteen experienced project managers. I asked each PM to choose some recent projects that he/she had managed. Then I asked the PM to compare and contrast these projects with one another, in terms of factors that had differed across the projects, and that had led to differences in the ways in which he/she had "shaped" and run these projects. In Part 1 of this book, I explain how I did all of this. I show examples of some

of the interview transcripts, to bring the process I used to life. Then I show the results I obtained.

In Phase 2 of the study, I used the factors I identified in Phase 1 to create a set of hypothetical project profiles. I then talked through these profiles with a further set of twenty IS/IT project managers. I explored with each of these PMs how he/she would handle a project with the particular profile I had chosen (randomly) for him/her. In Argyris and Schön's terms, I defined a set of hypothetical project (S)ituations, and I had the PMs describe how they would handle these situations. In other words, to describe the (A)ctions they would take to handle these situations. In Part 2 of the book, I explain how I did this, and I show the transcripts of all twenty conversations in full. These conversations make up the larger part of the book. This is the Studs Terkel-like piece!

In Phase 3 of the study, I analyzed the twenty conversations. My aim was to distill out the project managers' theories-of-action. I explain how I did this, and I show the results I obtained, in Part 3.

The project managers who helped me in this study are all very experienced and successful people. So, their opinions on how to do things carry a lot of weight. Nevertheless, I felt that I should make an independent effort to demonstrate the "rationality" (or otherwise!) of their theories. This I tried to do in Phase 4 of the study, in which I looked at their theories through a series of theoretical "lenses" that I had selected from the management and organizational literature. These efforts form the bulk of Part 4 of the book.

Who are my target readers? My primary intended readers are IS/IT/software project managers. But I think the book is of relevance to **any** professional who makes his/her living by bringing specialist skills to bear on solving problems for client organizations. Particularly if the problems to be solved are ill-defined, and the solutions entail organizational change. I also hope the book will be read by researchers and students of information systems and software engineering project management.

PART 1

**“What Makes
Different
Projects
Different?”**

How Project Managers Construe Projects

2

2.1 Introduction and Method

In this part of the book, I try to answer the question: What factors do experienced IS/IT project managers take into account when deciding on how to “shape” and manage new projects for new clients? In Argyris and Schön’s terms, the variety of *(S)ituations* that PMs have to deal with in the early, strategic phase of new projects. In Gluch’s terms, the potential *risk-conditions* that PMs have to manage.

A group of fourteen experienced IS/IT project managers (PMs) helped me to answer this question. All fourteen PMs had at least eight years’ experience of running projects for external clients. All but one were men. All were in the range 35–55 years of age. Ten of the fourteen PMs were owner-directors of their companies. The remaining four PMs reported to an owner-director. The numbers of developers employed in their companies ranged from two to ten persons. The scale of the projects they typically managed fell within the ranges:

Project duration: 2–18 months;

Project effort: 2–36 man-months;

Project team size: 1–6 people.

All worked with mainstream, current technology. All were in the business of providing information systems “solutions” to commercial clients.

I used the technique of *personal construct elicitation* (see Stewart and Stewart, 1981 or Bannister and Fransella, 1989). A personal construct is a bipolar distinction, or scale, which a person uses when contrasting different people, objects, situations, and so on. For example, for me, an important distinction between dogs is the likelihood that a dog will bite me! So, when comparing dogs, or thinking about a particular dog, I am likely to think in

terms of “will he/won’t he bite me?” People tend to have multiple sets of many interacting constructs to help them to make sense of the world. The task of identifying the set of constructs used by a person in a particular context is called *personal construct elicitation*. The technique I describe below is one of the standard approaches to construct elicitation.

I began the session with each PM by gathering some basic information about the PM and his company. I then asked the PM to make a list of the systems development projects he/she had worked on as project manager. I explained that the projects need not be very recent, nor need they yet have been completed, nor need they have been undertaken in the PM’s current organization. If he/she could identify more than nine projects, I asked him/her to choose the three that were the most successful, the three that were the least successful, and three “in-between.”

I then said: *Projects can differ from one another in terms of situational factors that developers must take into account when planning and running them. These factors could relate to the client, the “deliverable”, the resources available, or whatever.*

I then randomly chose three projects from the list and said: *Tell me in what important ways two of these three projects were the same, but different from the third, in terms of important situational factors you had to deal with. Particularly factors that created risk.*

I asked the PM to repeat this task with different triads of projects, until no new situational constructs were being elicited (or boredom/exhaustion had set in!).

Finally I asked: *Can you think of any other situational factors that you had to take into account when planning these projects?*

A typical construct elicitation session lasted for about one hour. The shortest session was about thirty minutes. The longest session was about ninety minutes. The minimum number of projects chosen by a PM was four. The maximum number of projects chosen was six. Most PMs chose five projects. Most sessions went very well. Most respondents said they found the exercise intriguing. Boredom was not apparent!

In the next section, I show the verbatim interview teamscripts for five of the PMs: Doug, John, Martin, Rene and Dick. My questions, interjections, and so on, are shown in bold type. I hope these transcripts will help to bring to life the interview process I used.

2.2 “So, What Are the Differences?”

DOUG

I’m going to call my projects A, B, C, D and E.

OK. Now compare C, D and E with one another. What are the important differences?

In D and E we were able to build complete prototypes. We couldn’t do this with C because of the nature of the system. In D and E the clients saw the prototype and understood the limitations, so expectations were met 100 per cent. In C, we had to carry the can if we didn’t analyze the requirements properly. I really don’t like not being able to show the client a prototype. If we are not able to do even the simplest prototype or mock-up, we feel very uncomfortable. In fact, if this happened to day, we’d bail out of the project.

Also, in C we weren’t implementing on our standard platform, because of client insistence. We were led to believe that everything was OK on the platform to be used. But there was too much learning involved for us. Not a happy experience!

In C there was a change of personnel also. The person who “bought” the system, the accountant, left the company. So it became a “reselling” job again.

B, D and E?

In all three cases we were dealing with the accounts department. D and E didn’t have a separate IT department. B was a bank and did. That’s both good and bad. The good point of having an IT department is that users’ expectations are more realistic. They have a much better understanding of timescales and what’s “doable.” The existence of an IT department gives more structure to the whole process. Regular meetings and so on. The bad point about IT departments is that they can slow you down, and they can be wedded to old technology.

B, C and E?

C and E stand out because they were much smaller companies with just one or two accounts staff. B had about twenty accounts staff. In B the scope of what we were doing was very narrow – just creditors’ accounting. In C and E we were looking at everything – receivables, payables and stock control. We

like doing narrow-scope jobs in departments of big companies, not broad-scope jobs in small companies. The expectation of small companies these days, with all the hype about fancy PCs and networks, is that they should be able to get everything of the shelf on a very tight budget. But in larger companies, where you are dealing with only one module within a large central system, they expect to spend much more on it.

B, C and D?

In B and D we were dealing with people who knew what they wanted. In C we were dealing with a marketing guy who didn't seem to know what he wanted, and who didn't appreciate the full scope of the business. B and D involved upgrading existing systems, so we had a framework to work with. C was a green-field job.

A, D and E?

In A and D we were dealing with good accountants. E was a half-baked accountant. We had to spend a lot of time educating people in E to show them what they should be getting from a good system. I find that in smaller companies a major part of the job is educating people to show them what they could be getting, or should be getting from a system. This is true even in the pre-sale phase . . . telling them what they should be looking for in a good system. It can help kill the competition!

D and E were more self-sufficient from a technical point of view . . . running networks, servers, housekeeping and so on. A needed a lot of technical hand-holding at implementation stage. We regularly had to wheel in our engineers to sort things out.

A, C and E?

A and C were totally dependent on us. They just sat back and waited for us to initiate everything. E had a lot more "go" and energy. They were much more self-sufficient and were much more receptive to training. Once the people in E were shown what had to be done, that was the end of the story, whereas the other two were constantly having to be shown what had to be done, and to be held by the hand. Partially, this had to do with staff volatility. Staff were moving all the time in both these companies. New people constantly having to be trained on the system. It ended up that anything that had to be done, they asked us to do it. This business of too much dependency is a hard one to suss out before you get to know a company.

Looking at the whole set of projects, any other big differences?

One of the first things I ask myself at the very start of a new project is, are we dealing with the real decision makers, or are the people we're dealing with just runners for the real decision makers. Then I look to see if there are the real users involved also. The successful projects are those that involve the real decision makers and the real users getting together in one room at the very start of the project.

Then I go through the standard things like: realistic timescales and budget, any signs of hidden agendas, can they afford to pay for the system, what expectations do they have? That's just sussing out your client. Then you pick your players for the game: will they be able to handle this job? Will they be able to handle the client? If the client is going to be very awkward, you want somebody strong to head up your team. Kicking off the project is the next most vital thing. As I said, getting your team together with the client and the users in the one room at the one table. Both key decision makers and the real users. If you don't get this right, your project will go all over the place. Over the years, our bad projects have always been the ones that had bad kick-offs. With a good kick-off meeting, you are well on your way.

JOHN

My projects are Dermot, Fred, Billy, Heather, Ann and Emer.

What about Dermot, Fred and Billy?

Fred stands out here. It was the only project where we suffered a bad debt in all the years I've been running the company. With the benefit of hindsight, there were a number of things about this client which should have set the alarm bells off. The premises were in a not particularly nice place and weren't well maintained. Then I discovered that their product was somewhat unethical. As a company, they did not have the ethics that I would insist on now if I was to do business with them. I suppose I learnt this one the hard way! You need to deal with companies that you feel comfortable to work with. Dermot and Billy were fine. That's the big difference between these three projects.

Billy, Heather and Dermot?

Billy was a project done through their IT department, which meant that the communications structure was much cleaner. The other two were done directly for end-users, which meant that, not only had we to take into account the whole development and implementation process, but we also

had to educate people about designing systems, costing them, and how to manage their own participation in the project. Ben was one of the systems that I really cut my teeth on, and on which we very nearly lost our shirt. It was the first big system that I ever managed. And it was in the area of production monitoring. Heather and Dermot were just database applications, and as such you could really scope out the potential downside risks. Because they are fairly straightforward. The production monitoring system was heavily based on doing calculations which were quite complex. Production engineers would be very au fait with the sorts of sums involved, but not computer people. For example, one of the things that needed to be calculated by the system was the percentage diamond usage on a grinding wheel. This sounds very straightforward, but to do it you had to take various factors into account. And it meant that to get the formula right took two or three visits to the plant. Then we had the problem of testing our “recipe”. If you are in a situation where someone says “I want that report sorted by surname”, you can do it and test it yourself. But we couldn’t do this in Ben. So we had to depend on users and engineers for huge input and testing.

Any other differences here?

Heather was done for a foreign national who didn’t speak good English. That added a few complexities to the problem. But you just have to deal with these kinds of things on a personal basis.

Heather, Ann and Fred?

I want to separate Ann from the other two. One of the mistakes we made with Ann was that we allowed ourselves to be sucked in to a wide-scope turnkey solution. We were to provide the network, software and services, and support on the network and the software and the hardware. In hindsight I would fight tooth and nail in the future to insist on splitting the software, the network and the hardware into separate projects because these three need different skill-sets. Unless you are specializing in hardware sales you can’t really afford to look after it and maintain it, so that’s when you need the likes of XXX and YYY to just come in with you and supply the hardware and maintain it and have a separate contract with the client. We found that sending out a programmer to deal with networking solutions took twice the time of a real network guy. The skill-sets are very different. You just can’t take a programmer and say he is an all-round animal. It’s horses for courses.

Another thing about Ann. Ann didn’t have a real project sponsor on the client’s side. I have often said jokingly, but very much mean it, that “The best projects we have delivered have had the best clients.” When the client knows

what they want and how to go about getting it, and are insistent that we give it to them, typically produces the best results. It sounds so obvious! Sometimes clients are only too willing to delegate the entire project to you. You could argue that, if that's what they want – fine. And to a certain extent in the sales process I would agree with you. But when it comes to actually implementing the project you need the client to buy in with you and get really involved. Strong clients good projects, weak clients – fifty-fifty whether you will have a good project.

Fred, Dermot and Emer?

This brings me back to Fred! The one blot on my financial notebook as I see it. It was one of the projects that I had the least technical input into as there was a project leader looking after it and I left him running on his own. In hindsight there were a number of things which should have flagged to me that things weren't going right. The project initially was fixed-price. Then we had two or three extensions to it which they agreed to pay retrospectively. So the . . . anyway, that one went badly wrong.

What about Ann, Fred and Dermot?

In two of these projects we were working directly with the client, not an IT department. And these two projects had ultimately far more problems. The final solutions were neither as elegant or as well done as they could have been. Working directly with the client, you're less likely to hit the right solution with the first design. I say to people "It's not just about moving the mountain to where the client says he wants it, you must put the mountain on wheels so you can move it again when he changes his mind!" The concept of putting the mountain on wheels is the kind of design that I try to get people to do when working directly with clients. But it doesn't yield elegant solutions.

Any other big differences? Between any of the projects?

Yes. Some of these are single-person projects. In my experience, one person can do 100 per cent, two people can do about 80 per cent each. It slides enormously as you have to break a project up. Version control gets harder, communication between people gets harder . . . anyway, we all know that.

I'm trying to think through my mental process of what I do when I scope out a new bespoke project. I suppose I look at the organization itself to see if it can afford the type of solution they are looking for. If someone is asking me to rewrite an accounts system because they want particular features specific

to them, in all conscience I have to ask myself “should I not just tell them to buy . . . (off-the-shelf)”. I also look at the way they are organized. If I walk into a guy’s office and it’s in a total mess, and he will be our main point of contact, I would add a couple of grand to the project. Then I look at anything we might have to do for the first time. If we’re trying to sell a system that we don’t know we can do, I defer things, go home and do some prototyping. I come in the next morning and say “Yes – we can do it.”

MARTIN

OK, I’ve picked five projects. they’re all pretty different from each other. I’ll call them A, B, C, D and E.

Let’s start with projects A, B and C. The big differences you had to take into account?

A and B were somewhat similar in so far as what was to be done was well known from the word go. It became a technical delivery job at an early stage. In C, the requirements were very fuzzy from the start. The biggest part of the job was to define the requirements.

Any other big differences between any two?

No.

What about A, B and D?

Again requirements for A and B were reasonably well defined. D was a bit like C in that the requirements were not too obvious at the start. But also the choice of the technology component of the solution for D was not as obvious as in the other projects, and the most likely technology to be used was pretty much new.

Any other differences?

In D, as well as being unable to foresee requirements and being into new technology, we also used new methods for the analysis and design phase of the project. This project brought a lot of difficult things together. While A was one we knew we could do from the outset. It was old technology, well-known requirements and used a well-known methodology.

Any other differences before we move on?

No.

Let's try C, D and A.

There was one big difference here. In C the part of the system we were working on was a relatively small part of a long-term plan which involved five other future systems. So we had to build a system which was doable in a reasonable timescale, but which would mesh in with these future systems. So we had to design all the systems together. D and A were really stand-alone systems.

B, D and E?

B was a very well-defined project as regards what had to be done, so even though it was bigger than the other two, it was simple to manage. E was different from the other two because it was extremely complex both mathematically and scope-wise. I'd almost say it was probably too complex to be run as one project. It went in as a single system but I really believe that you should keep deliverables to more manageable proportions. We just got away with it.

What do you mean by complex?

The large number of different screens to support a function was incredible. There was no obvious hierarchy in these because there were so many combinations needed. There were hundreds and hundreds of screens, all interacting together. The real hard bit was to get the users to agree on what they wanted because the whole system was almost too much for any one person to understand.

B, C and D?

C and B technology-wise had nothing hugely new about them. D had a lot of new technology coming into it. Also, I hadn't thought of this one before, D had a huge impact at the user end – on workflow and work procedures. Much more than B and C. In B and C it was changing something that already existed and not much change at that. In D, what we designed was going to totally change the way the company operated. And in B and C, if things went wrong during conversion, the old system could carry on. D had to go right first time. That's it with these three.

A, D and E?

D and E have a lot in common. They are both to do with multi-currency transaction processing. A was much more a database application. A did not

have a major user-impact. Once the database was designed properly, we were a fair bit down the line. With D and E we had to go heavily into how users use information and the work procedures that went with this. Maybe the real difference between D and E and A is that D and E were both real-time systems that affected how the company operated, so getting the user-interfaces correct and usable was vitally important.

A, C and E?

E was a complex online system that the organization involved really wanted very badly. And all the people working on it really wanted it. Their level of input to the project was incredible and they wanted to be involved at all stages. With A, the system was being imposed on the company because of legislation changes. That definitely made managing the project different. With A, you had to bring the users with you in so far as you weren't giving them any perceived added value. In E everyone came with you and wanted to come with you. In A the project was seen primarily as a nuisance. They knew it had to happen, but it wasn't going to affect their day-to-day productivity. This meant that you had to be sure that, when people said they agreed with you, that they actually meant it, and didn't just want to get rid of you.

E, D and C?

D was going to really change how the company was going to operate. People at the top really wanted it. But the people directly affected by it had very opposite feelings because of uncertainty about changes to their jobs. So the politics were heavy going.

How about standing back and looking at the WHOLE set of projects. Any other big differences?

Even though A was not addressing a complex system, it had a very tight timescale to it because of the planned legislation changes. It could easily have missed the deadline if things hadn't been done right. It went right mainly because there was one very strong person on the client's side who knew what he wanted, and why he wanted it. I wouldn't say that it couldn't have happened without him, but it was a major help having him there. He stuck with it all the way. So it was very much a hands-on collaboration, rather than his just saying "Get on with it and I hope it works out."

In B, we had to do the pushing. The client was very passive and left things to us to shape up and move along. But this didn't matter too much because we knew what was needed from the word go.

There's one difference that hasn't come up so far. I've already said that E was very complex as regards calculations and screens. But it also needed a number of analysts working on it. For all the other projects we've looked at, you are talking about teams of less than five analysts. E had more than that and I always find that if the analysis level gets big, project management gets very tough. It's very easy to get five or six different sub-systems analyzed . . . but getting them "meshed" together properly is one of the most complex parts of managing a project. E certainly stands out on that one.

Another difference that I see across the projects is that, for some projects, we were able to break the project up into easily "measurable" and manageable modules. This lets you measure where you are. With project A we had very large modules. People on the project were always saying "we are three-quarters through program X . . .". But one could have no level of confidence in this statement. If I was doing that project again, I would break it into more modules. This would have taken extra manpower, but from a risk point of view it would have been worth it.

RENE

I'm calling mine A, B, C, D, E and F.

A, B and C?

B and C were totally custom-built solutions . . . built from scratch. A used one of our existing products, with a custom-built user-interface added. So, in A we were more restricted in what we could offer, in that we had to compromise between what the client wanted and what we could offer. Whereas in B and C, which were built from the ground up, we could give them more or less anything they wanted.

B is somewhat different from the other two in that it was replacing an existing system written by another company. Basically, they wanted a lot of the functionality of the original system, but they weren't happy with its stability . . . its reliability. Performance was very slow, and parts weren't well written.

A is different also in that the specification we drew up was changed several times by the client during the process of development. Their organization was still in its start-up phase, and they kept making internal changes to the organization that affected the product we were building. Even before we gave them the first release, it had deviated from the original specification. Happily, our code was well structured and because of this it was easier to change. Still, we had to cope with this.

Let's try A, B and D.

D is somewhat similar to A in that it was a customization of one of our existing products. D was basically computerizing some existing manual forms. So there were very few changes to the initial specification. Whereas in A they were “growing” their procedures as we went along because they were a new start-up. In D, the manual forms were very well established and had been refined internally before we got involved . . . So they were very clear about what they wanted.

F, C and E?

All of these were built from scratch. E would be different in that it had to link to software that someone else wrote. Also, there was a hardware aspect to E. We had to communicate directly with a bit of custom-built hardware. We made up the hardware ourselves with the help of an engineer. We had to work out our own communications protocol. It was no-way an off-the-shelf solution. It was interesting and it was fun.

C presented different problems in terms of the software. It involved real-time reporting on a stream of incoming information. It needed graphics and was quite tricky in terms of processing. For example, our software had to interpret gaps in the incoming data stream and take appropriate action. We didn't have to buy in any skills for this one, nor did we have to learn new tricks. I had covered the mathematical ideas needed in my computing degree course. F was a very straightforward database system.

C is different from E and F in that C had their own computer department and had a heavy investment in computers. They had a mainframe on site and they had people knowledgeable about computers. Whereas E and F were much less technically-minded, end-user types, especially E who were very naïve on what computers could possibly do for them. In the early discussions with E, we had to work hard to find out exactly what they were looking for. We would ask questions like “Give me your ideal solution?” and “What exactly happens when . . .?” Then we'd say “The computer can do this for you”, and they'd say “great!”. Then a little later on they'd keep coming up with changes. With C, they had more set ideas. They knew what computers could do, and said what they wanted. Another difference . . . C dictated the pace of the project to us. Whereas in E we had to dictate the pace, and take the responsibility for telling them what computers could do for them.

Can I try A, C, F?

Sure.

A were quite specific on deadlines and dates. We had to meet their deadlines for the various stages of the project. They wanted beta versions on time for them to test. They had listed their own test schedules and the acceptance tests the software had to pass. C and F took the software when WE said it was ready and began running it immediately. If there were subsequent problems, they would give us a phone call. Another thing . . . C and F were single-developer projects, whereas A involved more developers.

I want to bring in D. D is different from A, C and F in that the system in D was specified by someone who didn't end up actually writing the system. It was "spec'd" by one of our people and developed by another of our people. C and F were "spec'd" and developed by the same person. I think the person who actually speaks directly with the customer has a much clearer understanding of exactly what's required. No matter how good the spec you pass the developer is, there will be understandings that just aren't conveyed properly. I insist that the person who does the spec checks the work as it progresses . . . and would hopefully spot problems.

Do you prefer that the person who does the spec also does the development?

Yes. But on larger projects it can't always be like this. So, I always set up a direct interface between the programmers and the client. Sometimes even the person who "spec'd" the project can't answer programmers' questions. So we let them talk to the client directly.

Looking across the whole set of projects, any other big differences?

They are all happy customers!

DICK

I'll call the projects A, B, C, D and E.

D, E and C?

D and E were larger projects. They involved innovative systems and software, at least in terms of our experience. C was really a "commodity" job. D and E were absolutely tailored, one-off situations. In C we half hoped we could turn the solution into a product that could be used in several other situations. D and E were done for big organizations . . . a government body

and a large manufacturer/exporter. C was done for a small general medical practice.

Did the scale difference make a difference to the way you had to think about the projects?

With the benefit of hindsight, I would have handled all the projects differently. C proved to be the most troublesome, in so far as we took a very simplistic approach to it. We didn't realize at the start that the client was totally unsophisticated. Even though we produced an early spec and got agreement on it, when it came towards implementation the goalposts kept moving and moving. Their ideas on requirements got more and more refined as they thought more about it . . . so we were chasing a moving target. The users in D and E were much more sophisticated . . . they were much more articulate in defining what they wanted. They were used to formalizing the steps in a project.

We had a meeting this morning about our projects generally and our approach for next year. We feel that dealing with the "lower end" of the market . . . with computer illiterates . . . has enormous costs for us and that clients here have an over-optimistic . . . over-simplistic . . . view . . . some of them can hardly cope if they have to press more than one button! We have now more or less made a rule that we won't get involved in that lower end again because it is too troublesome. There is too much client education involved in bringing a client who is in no way computer literate to being computer-literate . . . and we have to carry that overhead.

We are also learning that, because of the powerful development software available now, we can skip the classical stages like doing a written system specification. We can do a prototype . . . the client can come in, he can see and feel it . . . tell us if he likes it or not . . . we can say "make any modifications you want because it's easy to do . . . you'll be happier in the end". From a selling point of view this is very attractive to the client. But it means we must follow a much more formal route . . . or else we could get into an infinite loop.

B, D and E?

D and E were big database-type projects. We did ground-up systems for these which were totally written by us. In E we had to interface with some other software. There was a big overhead on this because we couldn't get at the source code . . . we literally had to experiment with the interface. We could put pure database people on B and D. On E we needed someone who could get down to almost machine-code level.

B, C and E?

Performance was a big factor in B. It wasn't so important in C and E. We had to change the implementation language in B because we just couldn't get the performance out of it. This was made even worse because our hands were tied initially on the platform that we were writing for. B was very troublesome for us, because they behaved like a trade union. The management knew nothing about computers . . . but they were brilliant negotiators, so every little extra was squeezed out of us. We eventually drew the line and this was the best thing we ever did. The politics there were fierce. There were no politics in E. They were pleasant to work for . . . the people we dealt with were very bright and computer-literate. They understood where we were coming from, and if we had problems they helped us rather than berate us.

B, C and D?

The key thing about D is that they are a multinational company. So they are quite prepared to spend money to get the best, and had a good understanding of what they wanted done. B were penny-pinchers. Always conscious of the price . . . moaning that they weren't getting value for money. They had the attitude "We can go out and buy EXCEL for \$300 and it does a whole lot of things . . . with big screens and a big manual. Your little piece of software does about one tenth of what EXCEL does, and it costs ten-times as much."

A, D and E?

A was an oddball accounting application. They were a charity that had to carefully account for income and expenditure. There were no such notions as profits, value added tax, nominal ledgers, or things like that. We had to invent a new type of accounting system for them.

A, C and E?

A and C were one-man projects. E was much bigger. The client there was more sophisticated and more articulate about requirements. In C, it was the opposite so we had a much bigger overhead and a big cost overrun.

A, B and E?

A was a single-person project. B and E were multi-person projects.

It's interesting . . . putting all these projects in front of me and asking me about differences. It's giving me a framework of the factors that I worry

about in projects. Things like . . . computer literacy . . . knowledge of the user about requirements . . . the type of system . . . I feel I'm getting repetitious now . . .

2.3 Analysing the Transcripts

The fourteen interview sessions were tape-recorded and then transcribed onto "hard copy". From the transcripts, I extracted the constructs used by the PMs. In most cases, it was possible to document the constructs verbatim from the transcript. In some cases, it was necessary to do some minor editing of the transcript to remove hesitations, etc. In all cases, the respondents were shown the final wording of their constructs, for approval or amendment. In almost all cases, the respondents said they were happy with the wording, so few amendments were required.

I collected a total of 201 constructs from the 14 PMs. The mean number of constructs identified by the PMs was 14.4. The minimum number identified was 10. The maximum was 19. The modal number of constructs identified was 13.

There was a considerable level of apparent overlap of constructs across the PMs. In the light of this, I collected together constructs that, on the surface at least, seemed to be "saying the same thing". In this way, I formed sets of constructs. If I was at all unsure about including a construct in a set, I allowed it to constitute a set of one. For each set of constructs, I attempted to construct a single, surrogate construct which I felt captured the common meaning underlying the constructs in the set. When I had completed this exercise, I gave each PM a list of his/her original constructs, which he/she had already approved, together with the corresponding surrogate constructs. I asked the PM to confirm, or otherwise, that the wording of the surrogate constructs captured precisely the intended meaning of his/her own constructs. In the great majority of cases, respondents confirmed their happiness with the surrogate constructs. Where the PM felt that a surrogate construct did not accurately capture the intended meaning of one of his/her constructs, I removed his/her construct from its set, and allowed it to stand alone with the original wording. This exercise of replacing sets of seemingly identical constructs with single surrogate constructs reduced the total number of constructs from 201 to 113.

In Table 2.1, I show the reduced set of 113 different constructs, together with the frequency of mention of each construct. The surrogate constructs are those with a frequency of mention greater than one. For the purpose of

Table 2.1 The constructs

Need to integrate/interface with other systems			19
#	With existing systems:		
1	We will be developing the application from scratch	The application will be building on a current system (not ours)	3
2	This will be a 'green-field' exercise	We will be inheriting existing technology and systems	1
3	We don't have to change other developer's code	We have to change other developer's code	3
4	We won't have to do any tricky interfacing with existing applications written by others	We will have to do some tricky interfacing with existing applications written by others	4
5	Our system need not integrate closely with any unfamiliar third-party system or product	Our system will have to integrate closely with an unfamiliar third-party system or product	2
6	We won't have to link/communicate directly with unfamiliar hardware devices	Our system will have to communicate directly with an unfamiliar hardware device	1
	With future systems:		
7	The new system doesn't have to be particularly adaptable to future needs	The system must be adaptable enough to cope with unknown future needs	3
8	The system probably won't need to interface with as yet unspecified future systems	The system must be able to interface with as yet unspecified future systems	2
The existence/competence/seniority/commitment of the project 'patron'			17
9	The client/sponsor is a clearly identifiable person	The client/sponsor is diffuse (e.g. a committee)	3
10	We are dealing directly with the key decision-maker	We are dealing with people who can't make the big decisions	2
11	Someone in the client organisation has taken clear, committed ownership of the project	Nobody seems to want to 'own' the project	3
12	The project 'owner' is high enough up the hierarchy to give cover and help if needed	There is no powerful person we can look to for cover and help if needed	1
13	The project 'owner' has a big stake in the success of the project	The project 'owner' has nothing much to lose if the project fails	1
14	The client/sponsor seems to be well able to manage any politics around the project	Our client could be ineffectual in handling any politics around the project	2

15	The person handling the project on the client's side has the time, skill and authority needed	The primary person on the client's side lacks the time, skill or authority to do the job	4
16	Our main point of contact seems to be an organized, competent guy	Our main point of contact seems to be "all over the place"	1
Scale/coordination complexity of the project Scale:			16
17	This will be a one-person project	This will be a multi-person project	2
18	We will need only one or two people on the project	We will need three or more people on the project	2
19	We will need fewer than five analyst/designers on the project	We will need five or more analyst/designers on the project	1
20	The project will take three months or less	The project will take more than three months	2
21	The project is less than one man-year	The project is more than one man-year	2
Coordination complexity:			
22	I will have the people/skills needed working full time on the project	The people/skills I need will have to be shared across other projects	1
23	The project will involve managing just a few disciplines	The project will involve managing a lot of different disciplines	2
24	The people who "spec" the requirements with the client will also write the system	The people who "spec" the requirements with the client won't be the writers of the system	1
25	We can do it all ourselves	We will have to do significant subcontracting	3
Level of change for the client			15
26	We are just computerizing existing procedures/systems	The procedures/system we design will be new to the client	3
27	We are just upgrading an existing system	We are replacing an existing system with a very different one	2
28	The functionality will not be new to the client	The functionality will be radically new to the client	2
29	There will be no major changes to workflow or procedures	There will be major changes to the customer's workflow/procedures	2
30	The system won't mean a lot of change to the customer's organization structure	The new system will mean a lot of change to the customer's organization structure	1
31	The system won't involve integrating functions now done in different offices	The system will integrate functions now done in different offices	1
32	The system involves just a single functional area	The system will span a number of different functional areas	3
33	The project won't require major skills transfer to the customer	The project will require major skills transfer to the customer	1

Client's knowledge/understanding of requirements		14
34 They seem to have thought out their requirements	They haven't thought out their requirements	3
35 We are dealing with skilled people who know what they want	We are dealing with people who don't know what they want	2
36 They have a good understanding of the problems they want solved	We will have to work with them to identify the problems	2
37 The client knows what he wants and it makes sense	We will have to "educate" the client to show them what they really need	2
38 They know what they want and how to go about getting it	They seem to want to "delegate" the whole project to us	2
39 The client is able to define the problem in IT addressable terms	We will have to define the problem	2
40 It will become clear early on what's needed	This will have to be a "prototype a bit" and then move on from there sort of project	1
IT competence and experience of customer/users		14
41 We are dealing with computer-literate users	The users are computer-illiterate	2
42 We are dealing with experienced computer users	We are dealing with people with little or no computer experience	2
43 We are dealing with skilled people who know what is possible/impossible	We are dealing with people who don't know what is possible/impossible	2
44 The client has realistic expectations about time, cost and what's "doable"	The client has unrealistic expectations	4
45 The customer is experienced in running computer projects	The customer is not experienced in running computer projects	2
46 We have an educated customer who knows what's involved in IT projects	We will have to educate the customer about running IT projects	1
47 The people involved know enough to manage their participation in the project	We will have to educate people about how to manage their participation in the project	1
Main source of control over the project		14
48 "Structuring" of the project will be driven by the client	"Structuring" will have to be driven by us	1
49 The client has the skills so we can safely share control with them	The client lacks the skills so we will have to take control	1
50 We will have enough control to ensure that what has to be done is done	We will be dependent of what the client's people are willingly prepared to do	1
51 We will have some influence over the requirements	The requirements seem to be already set in stone	1

52 If I feel uncomfortable, I'll be able to say "stop" while the problem is addressed	If I become uncomfortable during the project, I won't be able to say "stop"	1
53 We will be able to juggle a bit with timescales	We are working to a tight client-imposed timescale	3
54 The client is setting firm checkpoints and deadlines	The client is leaving it up to us to set the pace	2
55 We won't have to share control of the project with a third party	We will have to share control of the project with a third party (e.g. other consultants)	2
56 We won't be relying on third parties for critical, non-standard hardware or software	We will be relying on a third party for critical, non-standard hardware or software	1
57 We won't be in the hands of any subcontractors	We will be depending heavily on subcontractors	1
Enthusiasm/support/energy for the project		12
58 There seems to be genuine support for the project at all levels	There is support for the project at some levels but not at others	2
59 The people we are dealing with will stick with the project through thick and thin	If things go wrong, we are on our own!	1
60 The users have a real commitment to getting it right	There are no specific users who have a vested interest in getting it right	1
61 The client is willing to put a lot of commitment into the project	The client is not really very committed to the project	3
62 The client is very passive and will leave things to us to push along	The client is very active and will help pull the project along	2
63 The people most affected seem to genuinely welcome the system	The people most affected don't seem to want the system	2
64 There is no reason to think people will be hostile to the project	The people most affected are hostile to the project	1
Developer's experience of the application		8
65 We have experience of this application	The application is new to us	5
66 We have all the needed skills in-house	The project needs skills we don't have in-house	2
67 We've done it all before	We will be delivering functionality which is new to us	1
Number of disparate groups to be satisfied		7
68 We only have to satisfy a single group of similar users	Our solution has to satisfy multiple groups of users with different needs	4
69 There seems to be no serious internal conflict about what's needed	There seems to be serious internal conflict about what's needed	1

70	The project involves only a single interest group	The system involves conflicting interest groups	1
71	There seems to be no hidden agenda	The “real” agenda is hidden from us	1
Developer’s familiarity with the platform/environment/methods to be used			7
72	The platform/environment is new to us	The platform/environment is familiar	3
73	The development methods to be used are new to us	We will be using familiar development methods	4
Who we will be mainly dealing with . . .			7
74	We will be dealing mainly with a single individual	We will be working mainly through a group or committee	2
75	We will be dealing directly with the users	We won’t be dealing directly with the users	3
76	We will be working through the client’s DP department	The project will be done directly for users	2
Logical complexity of the application			6
77	The application is straightforward	The application is complex	3
78	The processing logic/algorithms are not complex	The processing logic/algorithms are complex	2
79	The system is small/simple enough for one person to get his head around it	The system is too complex for one person to get his head around it	1
Criticality/reversibility of roll-out of new system			5
80	We can pilot the new system until we get it right	The new system has to go right first time	2
81	If our system fails, they can fall back onto the old system	There is no fall-back if our system fails	1
82	The project has a phased implementation	The whole system must go “live” on a single date	2
Developer’s understanding of the business sector			4
83	The challenges will be mainly technical ones	The challenge will mainly be to understand the client’s business	2
84	We know this business sector	This is a business sector we are not familiar with	2
Maturity of the technology			4
85	We will be implementing on technology which is not very new	We will be implementing on technology which is pretty much new	3
86	The technology to be used is less than leading-edge	We will be using some “frothy” leading-edge technology	1
Ease of validation of the solution			4
87	We will be able to show the client an early prototype/mock-up	We won’t be able to show the client an early prototype/mock-up	3

88	We can validly test our solution before showing it to the client	We need a lot of input from the client to test our solution	1
Client's willingness/capability to handle implementation			4
89	The client has the willingness and skills to manage implementation issues	We will have to sort out/drive implementation	3
90	The client is competent from an implementation point of view (e.g. managing networks, PCs)	The client needs a lot of technical hand-holding and training for implementation	1
Freedom of choice of platform/development environment			3
91	We can choose the platform/environment	The platform/environment is a "given"	2
92	We will be defining the technical architecture	The technical architecture is pre-defined for us	1
Developer's knowledge of country/culture/language			2
93	We know the country/culture	We will be working in an unfamiliar country/culture	1
94	No language problems	We are dealing with people who don't speak good English	1
Stability of client's business environment			2
95	The client's organization is stable	The client's organization is going through a period of rapid change	1
96	The client's business environment is relatively stable	The client's business environment is changing rapidly	1
Other constructs			17
97	The decision makers are also the users	The decision makers are not the users	1
98	The users will have a lot of input into the design	The system will be "thrust" on the users without their input	1
99	They are quite prepared for innovative solutions	The solution will have to be a safe, cautious one	1
100	The client's goal seems to be to get the best possible deal out of us	The client seems to view us as a partner in solving the problem	1
101	We can safely use this project to try new things and learn new skills	We must be careful to stick with the tried and tested	1
102	The client will do their own acceptance testing	The client will leave it up to us to test the system and say when it's OK	1
103	It's a batch system	It's a real-time system	1
104	The primary objective is to improve a business process	It's a DSS/EIS/MIS-type system	1
105	The system is mission-critical	The system is not mission-critical	1

106 The client has had good experiences with IT in the past	The client has had big disappointments with IT in the past	1
107 The client's premises are in good shape and in a nice district	The client's premises are in bad shape or in a bad district	1
108 They look like they can afford to pay	It could be hard to get money out of them!	1
109 The client has a forward-looking IT department	The client's IT department acts as a brake	1
110 The client is a single-person or very small company	The client is a larger company	1
111 We are dealing with straightforward, factual, down-to-earth people	The client has a lot of the "aul Irish" about them	1
112 They are cautious people who want everything checked and cross-checked	They will let us get on with it without any unnecessary double-checking	1
113 They insist on everything being "spiced out" to the last detail	We aren't tormented with unnecessary bureaucracy	1

providing structure in Table 2.1, I have grouped together constructs which seem to closely share a common theme. I show the themes I identified in decreasing order of the frequency of mention of the constructs under the theme. This categorization is obviously a subjective one. In my defence, I tried to stay as close to the constructs as possible in compiling the categories.

Table 2.2 shows, for each theme, the number of managers mentioning at least one construct under that theme, and the total number of constructs under that theme (many managers had multiple constructs under the same theme.)

2.4 Some Obvious Questions . . .

At this point, a question may have occurred to the reader . . . Are fourteen PMs enough? Would I have identified more constructs and more themes had I conducted elicitation sessions with more PMs? Table 2.3 suggests that this is unlikely. No new themes seemed to emerge after the tenth elicitation session. Would more constructs have been elicited? The answer must surely be "yes", but one's intuition is that any new constructs would overlap a lot with those already elicited.

Here's a second question . . . One would expect that the PMs' theories-of-action would include a lot of "received wisdom" from the IS and software project management literature. Is this the case?

Table 2.2 Number of managers having at least one construct under each theme and the total number of constructs under each theme

#	Theme	Number of managers having at least one construct under the theme	Total number of constructs under the theme
1	The client's knowledge/understanding/clarity regarding the requirements/problem to be solved	12	14
2	The existence/competence/seniority/commitment of the project "patron"/"owner"	9	17
3	Level of IT competence and experience of the customer/users	9	14
4	Need to integrate/interface with other systems	9	19
5	Scale/coordination complexity of the project (numbers of disciplines, need to share resources, need to subcontract, etc.)	8	16
6	Main source of control over the project (developer versus client versus third parties)	8	14
7	Level of change to be experienced by the client (to procedures, work flow, structures, etc.)	7	15
8	The need to satisfy multiple groups of disparate users versus the need to satisfy one group of similar users	7	7
9	"Who we will be working through . . . (through users versus the IT department; through individuals versus committees)"	7	7
10	Developer's familiarity with platform/environment/methods	7	7
11	Developer's previous experience of the application	6	8
12	Level of enthusiasm/support/"energy" for the project in the client's organization	5	12
13	Logical complexity of the application	5	6
14	Ease of validation of the solution (e.g. possibility of prototyping)	4	4
15	Client's willingness/capability to handle implementation	3	4
16	Freedom of choice of platform/development environment	3	3
17	Criticality/reversibility of the roll-out of the new system	2	5
18	Maturity of the technology to be used	2	4

19	Developer's knowledge of country/culture/language	2	2
20	Stability of the client's business environment	2	2
21	Developer's knowledge of client's business sector	2	4
22	Other constructs (hard to classify)	8	17
Total		201	

Table 2.3 Cumulative number of themes mentioned by number of elicitation sessions completed

Number of sessions	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Cumulative number of themes	9	13	17	17	17	19	20	20	20	21	21	21	21	21

And a third one . . . Since all the PMs work at much the same business, one would expect to find a lot of overlap across their theories-of-action, and hence the situational factors they see to be important. Is this the case? In other words, do different project managers characterize project contexts in much the same way? Or do different project managers tend to use different “lenses” when “viewing” projects?

I will try to answer both these questions in the next chapter.

Coping with IS/IT Risk Management
The Recipes of Experienced Project Managers
Moynihan, T.
2002, X, 328 p. 175 illus., Softcover
ISBN: 978-1-85233-555-7