'It's lots of bits of paper and ticks and post-it notes and things . . .': a case study of a rapid application development project

Paul Beynon-Davies, Hugh Mackay* & Douglas Tudhope

School of Computing, University of Glamorgan, UK,
email: pbeynon@glamac.uk and dstudhope@glamac. UK, and *Faculty of Social Sciences, Open University, UK email: ahmackay@open.ac.uk

Abstract. *This paper reports an in-depth case study of a rapid application development (RAD) project. RAD is a recent information systems development method noted for its high levels of user involvement and use of iterative prototyping. The paper develops a model of the core features of RAD gleaned from literature such as that published on the dynamic systems development method (DSDM). We report an ethnographic study of a RAD project and use this case material to contrast the theory with the practice of RAD. We conclude the paper with a consideration of a number of possible revisions to the prescriptions of RAD and also discuss the role of RAD within the broader context of IS development.*

> *'It's lots of bits of paper and ticks and post-it notes and things . . . you'll get to understand it, it's not that bad really'. A RAD team member*

INTRODUCTION

This paper reports an in-depth case study of a development project in the UK, which involved the use of an information systems development method (ISDM) known as rapid applications development (RAD). The study lies in the context of interpretive approaches to IS research and the organizational context of development projects. RAD is an important contemporary ISDM in that it uses techniques such as mixed development teams, product-based project management and incremental prototyping. It is also a development method that is particularly reliant on rapid development tools. Such key features are described in this paper.

Our case material is analysed in terms of the contrast between canonical ideas of RAD practice expressed in methodologies such as dynamic systems development method (DSDM) and the RAD practices we observed in our study. The case study is significant as one of the few descriptions of RAD and, indeed, of Intranet development extant in the IS development literature. As such, it raises the question as to whether there is a need for new canons for development work of this nature. The paper also discusses a number of issues concerning the appropriate place of RAD within IS development practice.

CASE DESCRIPTION

In this section, we describe our research method, aspects of the context of the IS development project and critical elements of the development work important to our explication of RAD practice.

Research method

Methodologically, our study is in the style of the qualitative/interpretive tradition in IS research (Walsham & Han, 1991). There have been few empirical studies of IS development in this tradition, particularly of the front-end stages (analysis, design and construction) of development work (Button & Sharrock, 1993) compared with a developed literature addressing the issues of IS implementation (Keen, 1981; Kling & Iacono, 1984) and evaluation (Kumar, 1990).

The case reported here is part of a 3-year study that was concerned to map the diversity of prototyping in information systems development and to specify good practice, particularly in relation to developer–user relations. In the early stages of our project (Beynon-Davies *et al*., 1996), we found the issue of prototyping almost invariably associated in UK developers' perceptions with the issue of RAD (Martin, 1992).

In early 1996, we undertook an ethnography of an intensive RAD project, which involved a high degree of user involvement. This paper presents a case study of this project and draws conclusions from this case.

The most appropriate means for interpretive research is generally acknowledged to be the in-depth case study (Yin, 1994; Cavaye, 1996). Yin (1994) cites a number of reasons for using a single case study:

- it represents a critical case in testing a well-formulated theory;
- it represents an extreme or unique case;
- it is a revelatory case, involving a domain previously inaccessible.

We believe that the case is revelatory in highlighting features of a development domain not well covered in existing IS research. The case also has some unique features in representing an approach that breaks new ground in particular areas of development work.

The primary research approach used was that of an ethnography (Beynon-Davies, 1997). Ethnography is a research orientation that emphasizes a 'soft', interpretive approach to social reality. Ethnography is not, however, a single research method; rather, it is generally seen as a distinctive 'style' of social science enquiry. It is therefore useful to think of ethnography not as an integrated method, but as a series of partially unified methodological strategies, tried-and-tested protocols for accomplishing ethnography.

Over a 3-week period, one of our research team acted in the capacity of an observer on the project and was present in the setting. Detailed field notes were kept, some 27 hours of video recordings were made, and a substantial proportion of the development documentation was collected. The researcher also conducted interviews with team members throughout the life of the project, many of which were tape-recorded.

Context

Walsham (1993) developed a strategy for interpretive explanation in which a crucial element is the description of IS work within a framework of multilevel contexts. Using this approach, a number of levels of context are important for understanding our particular case.

Organizational context

Two levels of organizational context seem to be important in explaining the case study: the user organization and the development organization.

Employees of BT, a large private telecommunications utility company, undertook the project in the UK. The project was conducted for the corporate relations department (CRD) within this company by members of the IS division of BT. CRD is the department within BT that deals with all press and public relations (PR) throughout the organization. The department is distributed among a number of regions throughout the UK.

BT has a large, internal IS division, which again is distributed throughout the company. Until comparatively recently, all development work was conducted under the auspices of an in-house methodology similar to structured systems analysis and design method (SSADM) (Weaver *et al*., 1998). Over the last few years, a number of 'champions' within the IS division have been promoting RAD approaches. Before the project described in this paper, one RAD project had been conducted for the CRD, about a year earlier. This project, and the one described here, became identified by these champions as 'flagship' projects for RAD within BT. More recently, there have been suggestions within BT that all development projects need to include a RAD suitability assessment in their feasibility study phase. In other words, all new projects now have to consider using a RAD approach.

The project context

Access to the project described here was negotiated through one of the RAD champions within the IS division of BT. This person acted as an external RAD consultant to the project. The team assigned to the project was composed of seven people (to preserve confidentiality, all names of persons have been changed):

- The team leader, Anita, was a senior member of CRD. Anita was initially given responsibility for choosing other business members of the team and had been involved in one previous RAD project.
- Stephen and Simon were both 'librarians' at CRD. A CRD librarian is a person who archives PR materials, such as press cuttings, for use by other division members. Stephen had some previous experience in using Hypertext markup language (HTML).
- Sally and Paul were PR officers. Paul had previously trained users in the use of HTML.
- Anthony and Mortimer were developers from the IS division chosen because of their skills in the chosen tool-set – Sun workstations and PERL, the UNIX scripting language. Both had previous experience of RAD projects.

The five users, Anita, Paul, Sally, Simon and Stephen, were all referred to as 'customers' in the project plan. This was apparently an explicit and recent attempt by the IS division to acknowledge its service position within the company. All the business members of the team had been on a one-day RAD awareness session run within BT. The developers had been on the RAD awareness course and a three-day RAD practitioners' course, also run in-house by BT.

A workshop had been conducted at the start of the project in which a prioritized list of features for the eventual deliverable had been generated. A list of acceptance criteria for the deliverable and a risk analysis for the project had also been completed.

Both users and developers worked intensively at a location far removed from their normal workplace for three working weeks and were expected to deliver a working system at the end of that time. All the users stayed in the same hotel during the working week, and some travelled home at weekends. The developers normally lived and worked in the city where the project took place and, hence, commuted to the project site on a daily basis.

The work setting was the first floor of a multistorey building close to the city centre. This floor contained a large, open-plan office space. A corner of this floor was sectioned off by wall screens for use by the RAD team.

The system context

The system being developed was the second iteration of an Intranet for CRD. An Intranet normally means the application of Internet technology to facilitate an intraorganizational communications infrastructure. So-called 'firewalls' are normally placed between the Intranet and the Internet to protect access to internal data.

The system consisted of a diary module, campaign management module and a campaign management manual. The system was intended to assist PR campaigns to fit with corporate objectives. We shall call the project 'Project Face', as there were consistent formulations about the proposed application being used to supply a consistent face for the organization to the external world. The basic objective was to provide an on-line resource for all members of the CRD department in relation to BT's public relations. The aim was to ensure that 'nobody could do their jobs <within CRD> without having their computer switched on'. Outside communications from CRD should be conducted only by using Face.

Development work

In this section, we describe elements from the detail of the development work conducted on Face. Our aim is not to focus on the product developed in this project. Instead, we wish to emphasize aspects of the development process that are distinctive as far as RAD is concerned.

Week 1

The first week of development work was spent prioritizing requirements and constructing a design framework for the major modules of the information system.

Much of the first day was spent in an initiation meeting that was attended by all team members. At this meeting, Anita introduced the team to the principles of RAD and explained why this ISDM had been adopted for Project Face. She explained the roles of user, developer and the team leader, herself. The control of daily work was discussed in terms of lists of work items, referred to as a 'to-do list', which would be written on A2 sheets of paper that would be placed on the wall screens surrounding the project space. Anita discussed such project management devices in the following terms:

> 'We don't want to have to write things down, waste of time – pages and pages of procedures and things. It goes on the wall. We'll just throw it on the wall and that's documentation (apart from the developers who'll have to do a bit more than that). So there's a page a day. At the end of each day, we'll go through the list for the day and carry forward items that we've started. The deadlines will go up here. If there's something happening, just throw it on the wall so it's not forgotten. Anything at all just put it on the wall, so we'll know where it is.'

A major part of this extended meeting was taken up with a discussion of a series of requirements that were elicited at the workshop held some weeks before the commencement of this project. Such requirements were discussed particularly in terms of how easy or difficult they would be to do, and also in terms of assigning responsibility for the achievement of certain requirements to particular team members.

Items judged to be within the remit of the project were left to be done or, if 'in hand', marked with a green highlighting pen on the to-do list. Items considered outside the scope of the project were marked with a red highlighter pen. The sheet of things that were within the project was not redrawn to omit items discarded, but remained publicly available to all on a wall screen. In other meetings such as this, decisions were also made to 'park' an item, indicating that it was to be deferred to some other day within the project.

During team discussions, frequent reference was made to the notion of 'quick kills' – a development task that would be relatively easy to complete either through code reuse or the use of software packages such as spreadsheets. What is significant is that it was a user, Stephen, who was the most frequent identifier of such fixes.

The day's work terminated sometime after 17:30 h, and many evenings featured a 'team-building' activity. Team-building activities were both budgeted for and encouraged. The team went out for dinner, drinks or engaged in other collective activities such as ten-pin bowling together.

Day two set the typical pattern for the development work. The day started with team members arriving at around 08:45 h. Team members began by looking at the agreed work tasks entered on the to-do lists. They then went off to work at their PCs and workstations. Both the users and developers engaged in development work: customers worked on

word-processed manuals and HTML pages, and developers worked on threading this information into defined interfaces on workstations using PERL.

During the day, much informal interaction between particular users and developers occurred. Frequently, this took the form of either the user or the developer requesting the attention of the other for some form of demonstration. The relevant group of people would then huddle around a PC or workstation, usually for less than an hour at a time, after which they would break up to continue individual work again.

Periodically, users and sometimes developers would gather in a common area to discuss a particular problem or task. During such sessions, frequent use was made of low-technology devices such as 'brown paper' (large sheets of brown paper to act as a background), post-it notes and flip charts. Typically, these would be used to take elements from requirements documented on A2 sheets and writing them onto post-it notes, which were then placed onto a backdrop of brown paper. In this process, the elements were placed in design groups and, through this process, the functionality of the three major system modules was defined. It is interesting that the team had experienced no prior training in, or received advice on, the use of such low-technology design tools. However, such techniques were reported to be part of conventional meeting practice at CRD.

In meetings at which all the team members were present, the discourse seemed to be organized in terms of a discussion of the use of technology in a typical situation or episode of everyday work of CRD staff. For example, an episode discussed in one session was of the form: if one meets a Member of Parliament, then one should record the result of the meeting and do so in a form that allows others to check that they are not duplicating the work that has been done. Usually, once general scenarios such as this were established, the discourse then turned towards more specific or exceptional instances, usually phrased in terms of 'What happens if "X" occurs?'

Most days terminated with a full-team meeting, usually called a wash-up session, although sometimes referred to as a mop-up meeting. Broadly, a wash-up session involved the following activities:

- review of the day's progress with regard to objectives set;
- review of what had not been completed and hence remained to be done;
- documenting what is to be done the next day, and who is to do it;
- documenting how requirements were being met in a log maintained by Anita.

By the end of the first week of project work, an idea of the proposed facilities of the system had emerged.

Week 2

In the second week, the team spent their time building the major system components.

For users, there was a mixture of development work and what we may call everyday work (their 'day jobs') throughout each day. On most days, the users spent something like two hours

answering telephone calls, talking with colleagues outside the project about items, events and programmes of CRD and replying to e-mails from their home sites. Such communication was normally with those who had been designated as taking over from them for the weeks that they were on the development project. At one point, one of the user's telephones rang so often that the relevant team member made the comment that he would like to 'break the ****ing thing in two!'

The continuity of the development team was also disrupted at one point during the project. On the Wednesday of week two, a replacement for Sally, Robert, arrived. This replacement had been prearranged before the start of the project because of pressure of work. Robert came from the same BT site as Sally. She explained the development to him and described the nature of the work in the following terms – 'it's lots of bits of paper and ticks and post-it notes and things. . . you'll get to understand it, it's not that bad really'.

Week two was characterized particularly by developers demanding more frequent demonstrations of their work to users with much use of *in situ* prototyping. For instance, on the Thursday of week two, Anthony demonstrated an iteration of the diary system to Paul, Sally, Stephen and Anita. There were some changes requested by members, often by their pointing at the screen and asking if 'that could go over there'. Anthony made the changes *in situ* with recourse to the PERL repository.

The increase in the number of demonstrations may have resulted from perceived pressure on the part of developers to achieve an informal sign-off of their work. For example, on the same Thursday, Mortimer demonstrated his component of the system to Anita, Sally, Paul and Simon. They all stood around Mortimer's workstation. His development was generally well received, and there were some more on-line fixes made. As the reaction was positive, Stephen agreed to sign a part of the system off. At this point, our researcher was enjoined 'to get this on film!' While he was obviously joking, this is the first time that any form of sign-off was observed on the project. This highlights how 'sign-offs' were normally achieved informally, compared with the traditional model of formal, contractual sign-offs in IS development work.

There was much evidence of greater cohesion among the team in this second week. For instance, lunch was normally held in the same building as the setting for development work, with all members of the project team taking lunch at the same time. Normally, it could be taken any time between 12:00 h and 13:30 h. Usually, the decision to take lunch was a collective as opposed to an individual one. Often the suggestion that lunch should be taken was made after some significant waypoint such as a demonstration or a meeting regarding some part of the development. While it was not always Anita who initiated the invitation, she was most often the person who made the suggestion, perhaps indicating her role as the 'time-keeper' on the project. Lunch usually involved developers and users sitting together, frequently pulling two tables together to achieve this. It was not noted as being deviant to talk about the development during lunch, that is to say, lunch was not regarded as time away from the practical work. Indeed, solutions to a number of questions were talked through during lunch, often in a more informal manner, but usually with reference to people's ideas as to possible solutions.

During week two, a number of visits were made to the project site by managers of CRD to inspect the developing system. For instance, at the end of week two, higher level managers from CRD visited the site. Although the visit was portrayed as a friendly one, it was treated with some trepidation by the project team. One team member commented – 'you going to Hoover the red carpet then, eh Anita?'

On the morning after the demonstration, one of the high-level managers spent the morning discussing with Anita a reconception of how the system should be structured. It was noteworthy that this meeting took place in a small office away from the main project area.

Later, Anita formulated this reconceptualization of the system to the team as follows; 'well, he's looking at the system from a longer view and from a planning point of view. He's concerned to make sure that there are no fatal errors in the system'.

Week 3

The final week was spent 'polishing' the system, producing user documentation and constructing a detailed user training policy.

Robert and Paul were particularly involved in determining how the training should be done for Face. It was decided that they 'should have two or three super trainers who can cascade-train the people they work with, at least that's the theory'. Paul further noted that, 'everybody needs training on computers 'cause they're all useless anyhow, so you need to spend a day with each looking over their shoulders'.

In the middle of week three, Anita and one other team member were scheduled to present the current prototype at an annual colloquium for all CRD employees in London. Originally, this demonstration had been scheduled to occur within the working day of the forum. However, on the Monday of week three, the team learned that the demonstration had been shifted to a lunchtime session because more 'urgent' matters needed to be introduced into the forum schedule. This event served, not surprisingly, to cause some ill-feeling among the project team, and many felt that others had downgraded their work within the CRD.

Eventually, the demonstration took place much as planned. It went well, with only a few suggestions for changes or fixes to be made. The changes and fixes were made on-line the same day by the developers.

Further work

After the development described here, the Face system was implemented after a substantial period of training among members of the CRD. A number of extensions to the system were implemented over time, and a further development project was initiated a year or so after the events described here, which constituted an extension to the existing application to handle CRD communications to international PR agencies – an Extranet project. This Extranet project was run on similar lines to the one described here and was implemented successfully.

KEY FEATURES OF RAD

RAD is a comparatively new ISDM that can be seen to be a response to the changing business and development environment. In this section, we discuss aspects of this context and describe some of the key features of this ISDM.

Business and development context

Giddings (1984) characterizes information systems as domain-dependent software. Information systems are domain dependent, because there is a necessary inter-relationship between the software system and its universe of discourse. Such systems are characterized by an intrinsic uncertainty about the universe of discourse. In particular, the use of the software may change the nature of the universe of discourse and, hence, the nature of the problem being solved. For example, the development and introduction of an IS to support the domain of public relations may change how PR work is organized. Thus, what constitutes IS 'support' for PR staff may also change, leading to a redefinition of the problem an IS is designed to solve.

RAD can be seen as a response to two types of uncertainty evident in this proposition: business uncertainty and development uncertainty.

Business uncertainty

It is commonplace wisdom that the modern business environment is characterized by a high rate of change. Companies must react more quickly to changes in the business environment to survive (Scott Morton, 1991). Features of the current business environment include the globalization of markets and production, virtual organizations and customizable products (Drucker, 1994). In response to business uncertainty, there is some evidence of dissatisfaction on the part of business with the service provided by IS developers, demonstrated by articles that have been published on the so-called productivity paradox of information technology (Brynjolfson, 1993). There is also evidence of an increasing concern within organizations in the UK and elsewhere with the large amounts of money that appear to have been devoted to software projects with little apparent organizational benefit. The literature suggests that failure is a ubiquitous feature of both UK and international experience of IS development (Coopers, 1996). It is therefore perhaps not surprising to find that many companies are now pursuing outsourcing strategies in the face of dissatisfaction with internal IS (Lacity & Hirschheim, 1993).

Development uncertainty

The modern IS development environment is also subject to increasing rates of change, introducing uncertainty to the process of developing information systems. At the level of technology, for instance, developers now face a bewildering array of tools, including DBMS, 4gls, GUI

builders, CASE tools, middleware, etc. As a result, the IS development task no longer involves programming within the domain of a single language such as COBOL or C. Instead, IS development involves a hybrid of code production, reuse of established software components and distribution of software modules across local- and wide-area networks. Development uncertainty is exacerbated by the lack of a current dominant paradigm in the IS methodologies area. There appears to be something of a mismatch between established methodologies, such as SSADM, and the current development domain. Fitzgerald (1996), for instance has found a low take-up of large-scale methodologies in development in Eire. This is confirmed by research conducted by Moynihan & Taylor (1995) in the UK. A reason cited by both studies is the perceived cumbersome nature of traditional ISDMs.

RAD methods

RAD has emerged in recent years as a response to these issues. This ISDM first became topical in the IS development community with the publication of a text by Martin (1992). Martin defines the key objective of RAD as the commercial need to deliver working business applications in shorter timescales and for less investment.

A number of assumptions are seen as underlying RAD approaches:

- It is impossible to specify requirements accurately. Organizations are in a continual state of flux and, hence, it is impossible to 'freeze' requirements for a prolonged period of time.
- The application is its own model. The tendency of monolithic approaches to development work to generate large amounts of documentation is seen as inhibiting the speedy and effective delivery of systems. With the rise of high-level tools for development work, RAD proposes that applications can be largely self-documenting.
- Design and testing are iterative processes. Development work is a process of organizational learning, which demands iteration in terms of the design, production and testing of development artifacts.
- The application is always complete, but never finished. The end-product of development work is always a completed system. However, there is an acceptance within RAD approaches that systems are always subject to further change as a result of the uncertainty embodied in the relationship between organizations and IS.
- Empowerment of developers and users is crucial to the effective development of information systems. RAD makes much reference to material from the recent management literature, such as empowerment. The rationale is that effective development work involves mutual learning on the part of both developers and users. Hence, developers must be given freedom to learn about current work and formulate positions in relation to future work, and users must be given freedom to learn and make decisions about the potential of information technology.

In the UK, there has recently been an attempt to promote a standard RAD approach. The dynamic systems development method has now produced version 3 of a public domain RAD method (DSDM, 1995). However, although there are a number of methods available for RAD,

including those of Martin and DSDM, these are better seen as contingent than as prescriptive approaches. The DSDM manual, for example, is a slim, one-volume work compared with the eight or so volumes documenting SSADM.

RAD features

The assumptions defined above define the general philosophy of RAD. This philosophy is realized within RAD methods as a set of common practices or features. We divide these practices into four main areas: project features, team features, product features and process features.

Project features

RAD projects are typified by joint application design, focused problem-solving, short duration and a small number of participants.

JRP/JAD workshop

Most approaches to RAD use joint application design (JAD) workshops (Wood & Silver, 1989) at various points in the development process, particularly to elicit requirements (in which case, they are sometimes referred to as joint requirements planning or JRP workshops). In such workshops, key users, the client, some developers and a 'scribe' produce the system scope and business requirements under the direction of a 'facilitator'. Development teams are usually expected to come up with fully documented business requirements in 3–5 days. Such requirements may specify a series of phased deliverables over a given timespan.

Siting of project

JAD workshops are usually expected to take place away from the business and developer environments in 'clean' rooms – that is, places free from everyday work interruptions and provided with the requisite support facilities, such as flip-charts, post-its, coffee, computers, etc. The emphasis is on highly focused problem-solving.

Type of project

There are two types of RAD project: the intensive and the phased RAD project. In the intensive type of project, a team of developers and users are located in a clean room for several weeks and are expected to produce a working deliverable at the end of that time. A phased project is spread over several months. Such projects are normally initiated by a JAD or JRP workshop. The subsequent phases of the project are then frequently organized in terms of the delivery and demonstration of three incremental prototypes. The aim is continually to refine the prototype into something that is deliverable at the end of the project.

Project length

RAD projects are typically of relatively small scale and of short duration. Two to six months is a normal project length.

Team features

The formation of the project team in terms of size, composition, skill set and supporting activities is a critical characteristic of RAD.

Team size

RAD is characterized by small development teams – typically four to eight and a maximum of 10 people.

Team composition

RAD teams are composed of both developers and users. The team is empowered to make design decisions, and users frequently act as managers of projects.

Team skills

This means that all RAD team members must be skilled in terms of communication. Users must possess detailed knowledge of the application area, and developers must be skilled in the use of rapid development tools.

Team building

Extra-curricular activities that help the team fuse together are encouraged and often budgeted for.

Product features

The products of RAD usually demonstrate low background complexity but high interactivity. They are also normally built using rapid development tools.

Rapid tools

RAD demands good support from tools for rapid developmental change. This normally means some combination of fourth-generation languages (4gls), graphical user interface (GUI) builders, database management systems (DBMS) and computer-aided software engineering (CASE) tools. Using such tools, some changes to prototypes can be made *in situ* at user–developer meetings.

System complexity

Most RAD projects build applications that are not complex computationally. The tendency is to rule out the applicability of RAD for large-scale, infrastructure projects such as corporate-wide databases. Indeed, some practitioners propose that RAD projects can only be initiated once such infrastructure systems have been put in place.

System interactivity

Most RAD projects are conducted on applications that are highly interactive and have a clearly defined user group.

Process features

The RAD process is highly focused; it uses product-based project management, incremental prototyping and high levels of both user and developer involvement.

Focus on deliverables

A much-quoted axiom in RAD circles is that 80% of the functionality of a system can be completed in 20% of the time. The focus in traditional projects is on the activities or tasks associated with development. The focus in RAD is on the products or deliverables of the development process. A particular emphasis is on core functionality and avoiding 'copper-plating', i.e. producing an overly engineered solution for the problem at hand. Also, documentation is kept to a minimum.

Timeboxing

Project control in RAD is seen to involve scoping the project by prioritizing development and using negotiated delivery deadlines or 'timeboxes'. If projects start to slip, the emphasis in RAD projects is on reducing the requirements to fit the timebox, not extending the deadline.

Incremental prototyping

RAD is frequently discussed in terms of incremental prototyping and phased deliverables. The cycle of inspection–discussion–amendment is usually repeated at least three times in RAD projects until the user is satisfied with the system. Prototyping is used throughout the development life cycle.

User involvement

Ideally, users should focus only on project work for the duration of the RAD project. Practically, the literature suggests that such involvement generally be interleaved with other work.

Developer involvement

Ideally, developers should focus only on the RAD project. Again, sources suggest that such involvement will typically be interleaved with other development work.

User–developer interaction

Interaction between users and developers varies among projects. On some projects, particularly those in which both developers and users are highly focused, interaction may be on a daily basis. On other projects, interaction may be on a weekly, monthly or quarterly basis. Interaction may also be organized in terms of formal meetings or left to more informal interaction.

ISD THEORY AND PRACTICE

The empirical study of ISDMs is a much neglected area of information systems (IS) research. Wynekoop & Russo (1997) conducted a systematic survey of the literature on ISDMs. They found that over half the 123 research papers examined consisted of normative research, in which concept development was not based on any empirical grounding or theoretical analysis, but merely on the authors' speculations and opinions. Of those that constituted empirical research, almost half were evaluations of ISDMs or parts of ISDMs. Few studies were undertaken to identify how ISDMs are selected or adapted, or how they are used. There also appears to be little interpretive research and few practice descriptions or case studies.

We see our research as useful in documenting a previously unstudied area, particularly in highlighting some of the inherent features of RAD work in practice. It is particularly significant as an example of how a development approach is used and adapted, and also as an example of an under-studied area, that of Intranet development. As such, it provides some support for Fitzgerald's (1997) call for new canons for development based on the close empirical study of practical work. It also resonates with Rockart & Hoffman's (1992) call for different styles of development approach to be used for different types of IS applications.

Business and development contrast

On several occasions after the conclusion of the project, RAD champions within BT referred to this project as the 'purest' form of RAD they had yet conducted. Purity was expressed in terms of the use on the project of some of the key components of RAD, such as JAD, clean rooms, prototyping, rapid development and timeboxing. The intensive nature of this particular project was also important as a characteristic of purity. These champions, however, acknowledged that the project was somewhat atypical of most RAD projects they had experienced in being an Intranet development. The typical RAD project in their eyes was one using 4gls/CASE/DBMS to build interfaces, typically to large corporate databases.

There is some evidence that CRD chose to participate in a RAD approach because of previous bad experiences of other development approaches that had been applied to IS problems within CRD. Members of CRD reported that they felt more in control of the development using a RAD approach.

Project contrast

A one-day workshop run on the lines of a joint requirements planning session was run some weeks before the start of the project.

It is perhaps not surprising that frequent violations of the clean room ethic occurred in our study. For instance, a substantial amount of time was spent on a daily basis by users in managing work-based interruptions – 'my day job'. Also, one of the key users was replaced on the project in week two. This runs counter to the RAD ideal that, 'substitute attendees are discouraged because they are perceived to have a detrimental effect on the process and time is wasted' (John, 1994).

As we discussed, our case was clearly an example of an intensive RAD project. The literature and our previous interviews with RAD consultants suggest that the typical RAD project was one in which contact between users and developers was less intense than in the one we studied. Typicality was expressed in terms of a project spread out over a number of months – initial JAD session, followed by first, second and third prototype demonstrations. Such a phased project is clearly much more meeting based than the one discussed here. This would perhaps suggest that there is a need for more documentation in this approach, as communication and agreement cannot be formulated on an hour-by-hour basis.

Team contrast

The team size of seven falls within the recommended limits for RAD teams. One might therefore hypothesize that there is probably an upper limit to the size of the team and the scale of the type of work that can be conducted using an intensive approach like the one described, because of an inherent communication overhead.

In the RAD literature, continual emphasis is made on the importance of formulating representation on the project team from within the user community, while also providing authority to team members ('empowerment') to make development decisions. On a day-to-day basis, this was clearly evident in how members formulated design decisions without recourse to any outside agencies. However, when a major player from outside the project team appeared in week two and criticized the design, the design was reworked to meet the demands of a more powerful group of stakeholders. Clearly, this illustrates that representation and empowerment are likely to be negotiated throughout the life of a RAD project, and that perhaps some of the prescriptions in the RAD literature are overly idealistic in this respect.

As far as the composition of the team is concerned, both users and developers were skilled with the development tools. The users were particularly good communicators, perhaps not surprising given their role within BT.

The literature emphasizes the central importance of the team leader for managing consensus. This was certainly evident in our observation on this project, in that Anita was the key manager of to-do lists and wash-up sessions. Much of Anita's work involved mediating between developers and users and monitoring the issue of time closely.

The team seemed to 'gel' extremely effectively and completed all of the core requirements for the system within the allotted time. However, over the three weeks, a distinct 'them-and-us' attitude started to pervade the project, particularly in relation to events such as the CRD forum. This was one of several instances in which the boundary was not between users and developers but between those on the project team and those outside it. Engendering such a team spirit is seen as a necessary step in replacing the 'traditional safety net of . . . the signed-off specification' (DSDM, 1995) by an ethos of joint responsibility for any problems that develop. However, Mumford (1983) cautioned against the way in which participation may lose its representative character in the process of establishing group boundaries in this way.

Process contrast

One might propose that part of the rationale for siting projects away from developer and user sites appears to be to produce an environment in which more intensive and prolonged work takes place. However, although people referred occasionally to the 'hot-house atmosphere' of the development work on this project, the team members, although working hard, did not appear to work the long hours described in the US literature (Kerr & Hunter, 1994). They also did not appear to be under the type of pressure described in this literature. Developer and customer 'burn-out' may therefore have been somewhat less of an issue, particularly as RAD was very much in the early stages of acceptance at BT.

Although timeboxing as a technique was referred to only infrequently by team members, we found some timeboxing practice on the BT project. However, the timeboxing was much more informal in nature than that prescribed by the literature, centring on the use of to-do lists.

The RAD literature makes much reference to an approach for prioritizing requirements known as MoSCoW. A timebox can be thought of as a set of products that must fit in a finite container. Four types of product (requirement) are frequently phrased in terms of the mnemonic MoSCoW – Must have, Should have, Could have, Won't have. Normally 'must haves' and 'should haves' fill the timebox, leading to 'could haves' and 'won't haves' being pushed out. Although MoSCoW terms were not used explicitly by team members, there was clear evidence of prioritization activity in wash-up meetings throughout the project. Particularly significant was the use of flipcharts and highlighter pens for representing the priority of work items.

One of the most notable things about the day-to-day activity in this study was that few overt disagreements occurred between group members. There was some evidence that, when they did occur, disagreements seemed to be formulated 'off-stage' (Goffman, 1990), i.e. in a space outside the clean room. When the system had to be demonstrated to 'outsiders', one team member was usually nominated to explain the system. The only time that a major disagreement arose, it was conducted in a closed room away from the central project activity. This

norm may, of course, have nothing to do with intensive RAD as such, but could merely be a reflection of the inherent organizational culture, or even merely because the system was perceived as being particularly unproblematic.

Schön (1983) discussed design as a reflective conversation with the situation. The designer 'shapes the situation, in accordance with his initial appreciation of it, the situation "talks back" and he responds to the situations back-talk. In a good process of design, this conversation with the situation is reflective. In answer to the situation's back-talk, the designer reflects-in-action on the construction of the problem, the strategies of action, or the model of the phenomena, which have been implicit in his moves.'

This metaphor of design as a reflective conversation with the situation is certainly useful for understanding how the project team debated the use of the technology. Design on a RAD project, however, is different from that portrayed in Schön's work because of its inherent collaborative nature. In collaborative design, the 'situation' is not an individual cognitive construct, but the result of group appreciation and negotiation. Clearly, to achieve such a collective appreciation of the situation, certain representations are needed to communicate effectively among the group the current appreciation of the design situation. In the project studied, such representations frequently amounted to using low technology (Muller, 1992), such as brown paper and post-it notes, to mock up aspects of user interfaces. The use of flip-charts and the posting of such representations on walls, of course, are a common JAD technique (Wood & Silver, 1989). What is unusual in this project is that, because the team had unlimited access to the 'clean room' for the entire three weeks, these representations could be left up as a form of public history or 'collective memory' of activities. It was thus permanently available for all project members to orient their work towards.

We observed some variation in the nature of user and developer demonstrations. User demonstrations involved a lot of acknowledgements of a common but tacit understanding of work issues. Developer demonstrations tended to be more formal occasions in which the objective appeared to be to achieve some form of informal 'sign-off' of the technical changes made to system elements. This 'technique' is explicitly mentioned in the RAD literature.

Team members seldom formulated agreement on design decisions in an overt, formal way. Agreement was frequently achieved either by the group 'talking out' an issue or by Anita using flip-charts to note down design decisions. Frequently, 'talking out' an issue meant framing a series of scenarios, which defined the expected context for use. At a more concrete level, brown paper and post-it notes were used to represent emergent designs as physical and manipulable artifacts. It is interesting that Hirschheim & Newman (1991) posit the role of formal sign-offs as a ritual, which serves to perpetuate the imagery of 'professional' conduct. In RAD settings such as the one described, such imagery appears to be deliberately downplayed, with informal rather than formal representations of consent being relied on.

Product contrast

In terms of RAD, this project was perhaps unusual in using HTML editors and the PERL UNIX scripting language as primary development tools compared with the usual fourth-generation

language (4gl)/computer-aided software engineering (CASE)/database management system (DBMS) type of toolkit.

The level of interactivity of the information systems product was significant. The primary aim of the system was to supply an on-line resource to all organizational materials that needed to be accessed by PR personnel at BT. The level of complexity of the system was reasonably low, and it involved standard Internet technologies with much use of readily available software modules. Also, the eventual system did not appear to integrate to any BT infrastructure systems.

CONCLUSION

In this section, we address the significance of our research, summarize some of its key findings and develop some ideas for future work in this area.

Importance

One of our aims has been to describe work on a RAD development project. Such descriptions of actual development work are comparatively rare in the IS development literature. Notable exceptions are Button & Sharrock's (1993) description of the various ways in which software engineers negotiate an order of work that is distinctly different from that described in standard development methodologies and Curtis *et al*.'s (1988) study of 17 large development projects in the US, which documented the importance of project mediators to successful IS construction.

Summary

RAD

RAD is an increasingly used modern development canon, which uses distinctive development techniques, such as joint development teams, product-based project management and prototyping. This ISDM is particularly important because of its encouragement of user involvement within the domain of commercial IS development. We have argued elsewhere (Beynon-Davies & Holmes, 1998) that RAD bears a family resemblance to some strands of participatory design.

Case

The case described in this paper is particularly interesting because of its status as one of the few documented descriptions of a project devoted to the construction of an Intranet. This is perhaps surprising, given the increasing role that Internet technologies play in the modern IS portfolio.

One could argue that Intranet systems are particularly suitable for RAD approaches in that this type of IS product generally displays low levels of structural complexity but high levels of interactivity.

The case study is also interesting in that it represents an intensive RAD project. Intensive projects are characterized by continuous and focused involvement on the part of both developers and representative users in the development project. We might hypothesize that such levels of involvement appear to be rare in commercial development projects because of the financial burden imposed by extracting users from their normal work.

The way in which project work was controlled by business users is significant in addressing the perceived semantic gap between business and IS/IT. Given the primary importance of the user interface to an Intranet, one would perhaps expect that user involvement and rapid feedback are critical to success in development work. One might also argue that members of the development team need to be taken from skill domains other than the simple business domain. For instance, one would expect that people skilled in graphic design would assume significance in future projects of this nature.

The ability to make changes to developing prototypes *in situ* is reliant on appropriate high-level tools. Such *in situ* prototyping was a significant feature of work on this project, and one might argue that such rapid development tools assume a great significance in such highly focused projects as the one described here.

Contrast

The first issue to mention is the 'purity' of the project. The project described falls close to the range of desirable features for RAD projects discussed in the literature. Elsewhere, we have discussed a number of projects that lie further from the ideal type described in this paper (Beynon-Davies *et al*., 1999). This commonplace divergence from a prescriptive ideal is confirmed by research on other ISDMs, such as SSADM (Edwards *et al*., 1989).

Another critical feature of the project work was the importance of informal social organization. As well as design representations, the project team also used low technology for scheduling and monitoring work activity. For a group of people to accomplish collaborative work on any project, whatever its nature, group members need a clear understanding of what needs to be done, what has been done, what they are doing currently, what they are going to do next, and how individual activity relates to what other people are doing. This intersubjectivity is an achievement of the usually taken-for-granted working practices employed on a project. The apparent orderliness that emerged in the course of this project was not prescribed in advance by an ISDM. It was an outcome of the 'methods' (Garfinkel, 1967) that members used for collaborative working, achieving mutual awareness and intelligibility through 'organizing devices' such as to-do lists and wash-up sessions (Suchman, 1992).

To enable co-ordination of work, there was much use of time-management and co-ordination techniques. Whereas in the phased type of RAD project, the co-ordination of activities has to rely on more formal types of communication, such as documented designs, minutes and memos, on this project, 'the collective memory' was negotiated in relation to infor-

mal and tacit understandings relating to work co-ordination. In this sense, the to-do lists and wash-up sessions served as artifacts that enabled the co-ordination of work. The development work described here could certainly be described as being methodical, even though it did not adhere to any clearly defined method, only loosely following the stages and practices defined in the DSDM manual.

Further work

While RAD can be seen as a response to uncertainty in the business and development domain, it also raises questions concerning its appropriate place within IS development practice. The focus of our research has been on user–developer relations. However, developers have continually reiterated to us their concerns over issues of cost, scalability, benefits and culture in relation to RAD.

Cost

In our interviews with practitioners, many have questioned the cost implications of maintaining clean rooms and the greater degree to which business users are involved in RAD projects. Further work is needed on the financial implications of this form of development work.

Scalability

Questions have also been raised as to the scalability of RAD approaches. RAD is normally used on small-scale projects. Clearly, members of organizations are very interested in whether the benefits of RAD approaches may be achieved in relation to infrastructure as well as interface projects.

Justification

One particular problem with RAD relates to the difficulties of accounting in formal terms for the business benefits of RAD approaches such as the one described here. Many of the proposed benefits of RAD are intangible, such as greater satisfaction with systems or greater commitment on the part of users to systems. Such benefits are clearly very difficult to quantify. This makes it difficult to appraise the investments in RAD projects using traditional techniques.

Culture

Changes seem to be required in terms of the organization of both development and a business to enable the effective utilization of RAD approaches. Several interviewees formulated this in terms of a needed culture change on the part of both developers and users. Develop-

ers need to be much more prepared to engage with high levels of user involvement, while business people must be much more committed to involvement in development work.

These questions represent the rationale for more much-needed research in this important area of IS development.

ACKNOWLEDGEMENTS

The work on this project was funded by the UK Economic and Social Research Council (grant no. R000 23 5505). We would like to thank all those at BT who agreed to talk to us about, as well as letting us observe, their RAD work. Our thanks are also extended to Dr Roger Slack who was research fellow on the project.

REFERENCES

Belgnon-Davies, P. (1997) Ethnography and Information Systems Development: Ethnography of, for and within IS development. Information and Software Technology **39**(1), 531–540.

Beynon-Davies, P. & Holmes, S. (1998) Integrating rapid application development and participatory design. *IEE Proceedings Software*, **145**, 105–112.

Beynon-Davies, P.H., Mackay, H. *et al*. (1996) Prototyping in information systems development: a study of development practice in natural settings. *European Conference on Information Systems*. Lisbon, Portugal, pp. 953–969.

Beynon-Davies, P., Carne, C., Mackay, H. & Tudhope, D. (1999) Rapid Application Development: an empirical review. *European Journal of Information Systems*, **8**, 211–223.

Brynjolfson, E. (1993) The productivity paradox of information technology. *CACM*, **36** (12), 67–77.

Button, G. & Sharrock, W. (1993) Practices in the work of ordering software development. In: *The Discourse of Negotiation*. Firth, A. (ed.), pp. 159–180. Pergamon, Oxford.

Cavaye, A.L.M. (1996) Case study research: a multi-faceted research approach for IS. *Information Systems Journal*, **6**, 227–242.

Coopers (1996) *Managing Information and Systems Risks: Results of an International Survey of Large Organisations.* Coopers and Lybrand, London.

Curtis, B., Krasner, H. & Iscoe, N. (1988) A field study of the software design process for large systems. *CACM*, **31**, 1268–1287.

Drucker, P.F. (1994) The theory of the business. *Harvard Business Review*, **72**, 95–104.

DSDM (1995). *Dynamic Systems Development Method (Version 2).* Tesseract Publishing, UK.

Edwards, H.M., Thompson, J.B. & Smith, P. (1989) Experiences in the Use of SSADM: series of case studies. Part 1: first-time users. *Information and Software Technology*, **21** (8), 21–28.

Fitzgerald, B. (1996) An investigation into the use of systems development methodologies in practice. *European Conference on Information Systems*, Lisbon, Portugal, pp. 3–23.

Fitzgerald, B. (1997) *Systems Development Methodologies: a Need for New Canons. Discussion Paper*. Executive Systems Research Centre, University College, Cork.

Garfinkel, H. (1967) *Studies in Ethnomethodology*. Prentice-Hall, Englewood-Cliffs.

Giddings, R.V. (1984) Accommodating uncertainty in software design. *CACM*, **27**, 428–434.

Goffman, E. (1971) *The Presentation of Self in Everyday Life*. Penguin, Harmondsworth.

Hirscheim, R. & Newman, M. (1991) Symbolism and IS Development: myth, metaphor and magic. *IS Research*, **2**(1), 29–62.

John, T. (1994) *Joint Application Design: Theory and Practice.* Business School, University of Wales, Cardiff.

Keen, P. (1981) Information systems and organisational change. *CACM*, **24**, 531–552.

Kerr, J. & Hunter R. (1994) *Inside RAD: how to build fully functional computer system in 90 days or less*. McGraw-Hill, New York.

Kling, R. & Iacono, S. (1984) The control of IS developments after implementation. *CACM*, **27**, 1218–1226.

Kumar, K. (1990) Post implementation evaluation of computer-based information systems: current practices. *CACM*, **33**, 236–254.

Lacity, M. & Hirschheim, R. (1993) *Information Systems Outsourcing: Myths, Metaphors and Realities*. John Wiley, Chichester.

Martin, J. (1992) *Rapid Application Development*. Prentice Hall, Englewood-Cliffs, NJ.

Moynihan, E. & Taylor, M. (1995) The evolution of systems methodologies and why they are sometimes not used. *Proceedings BCS Information Systems Methodologies Conference,* Wrexham, UK.

Muller, M.J. (1992) Retrospective on a year of participatory design using the PICTIVE technique. *Proceedings of Computers and Human Interaction*, San Diego, CA.

Mumford, E. (1983) *Designing Participatively*, Manchester Business School Press, Manchester.

Rockart, J.F. & Hoffman, J.D. (1992) Systems delivery: evolving new strategies. *Sloan Management Review*, **Summer**, 21–31.

Schön, D.A. (1983) The Reflective Practioner: how professionals think in action. Ashgate, Aldershot.

Scott Morton, M.S. (ed.) (1991) *The Corporation of the 1990s: Information Technology and Organisational Transformation*. Oxford University Press, New York.

Suchman, L. (1992) Technologies of Accountability: of lizards and aeroplanes. In *Technology in Working Order: studies of work interaction and technology*. Button, G. (ed.), pp. 113–126. Routledge, London.

Walsham, G. (1993) *Interpreting Information Systems in Organisations*. John Wiley, Chichester.

Walsham, G. & Han, C.-K. (1991) Structuration theory and information systems research. *Journal of Applied Systems Analysis*, **17**, 77–85.

Weaver, P.L.N., Lambrou, N. *et al*. (1998) *Practical SSADM 4+*. Pitman, London.

Wood, J. & Silver, D. (1989) *Joint Application Design: How to Design Quality Systems in 40% Less Time*. John Wiley, New York.

Wynekoop, J.L. & Russo, N.L. (1997) Studying system development methodologies: an examination of research methods. *Information Systems Journal*, **7**, 47–65.

Yin, R.K. (1994) *Case Study Research: Design and Methods*. Sage, California.