The Framework and suitability of RAD Toolset for DSS

Howard J Harris
Brunel University
Uxbridge, Middlesex
Howard.Harris@Brunel.ac.uk

Jason C Peters
Brunel University
Uxbridge, Middlesex
Jason.Peters@Brunel.ac.uk

Abstract:

This paper seeks to address the need of adaptability and flexability in DSS system development. The aim of DSS systems is to give managers control of raw data and assist in the process of knowledge creation. This requires Information System's that are capable of continual change responding to the environment and importantly encapsulate a user centred design philosophy. Reflecting many of the wider concerns of Information System development in general. One suggested approach of addressing the conceptual difficulties of the fit between organisations, users and that of Information System designers is Flexibility Analysis. This paper configures an adopted framework based upon the RAD approach. RAD's original goal was to facilitate a faster production of system realisation; given its pedigree and its specified eclectic tool kit it makes it an ideal candidate.

Keywords: Rapid Application Development (RAD); Decision Support Systems (DSS); Flexibility Analysis (FA); User Centred Design (UCD);

Section 1: Introduction

This study addresses the issues of systems development of Decision Support Systems (DSS) within the context of continual change. In particular we concentrate on the software project managerial issues of time, cost and quality and the focus on user centred design (UCD) as a way of assessing the suitability of Rapid Application Development (RAD) methodology for implementing DSS. The broader issues can be portrayed as general problems to information systems, however we will address matters that inherently affect DSS directly, as it forms a sub-set of the information systems arena. The linking of the above issues to DSS is important given the current state of business environments and organizations seeking continual change. Therefore the requirements and qualities of DSS is demanding in terms of data/information that is accurate, timely, appropriate and relevant.

Authors such as Ho & Sculli and Arinze have studied individual aspects of the issues of systems complexity and design and user inquiry types with specific reference to DSS (Ho and Sculli 1995). This has led, we feel, to a gap or at least passé research to the above concerns and therefore seek to address this within this paper. Having identified RAD for its prototyping capabilities and its continual development process approach, its framework and

suitability is addressed in relation to DSS. However RAD has not concentrated on specifically addressing the organizational change problems and issues. We have identified that the business environment is becoming more complex, with increased competition, global challenges and market shifts coupled with rapid technological developments and the increasing importance of the World Wide Web (WWW) and electronic commerce (Fitzgerald, Philippides et al. 1999).

This paper contextually addresses information systems development (ISD) in an organizational context and thus effects and focuses on users in relation to UCD applications for DSS due to its knowledge based needs and assets. The uses of DSS vary from applications such as simulation packages to artificial intelligence. This clearly emphasizes on the diversity of the potential uses and users of such systems.

With the rise of distributed computing and especially the WWW and Intranets, it is timely to re-examine the methodological approach adopted for the development of DSS. Recognizing that organizational environments should not be considered as fixed or static entities but have to respond to change in relation to the environment, users and technology as acclaimed by Flexibility Analysis (FA). The requirement of building flexibility into information systems such that they will be more adaptable and easy to change as new requirements are encountered over the life of a system (Fitzgerald 1990).

Specifically this paper assesses the suitability of the RAD methodical approach to address the issues of FA, UCD and the project managerial issues of cost, time & quality, within the context of the IT modern modular architecture. This is achieved through a proposed adapted framework.

The approach we have taken is to introduce a conceptual paper that seeks to develop a coherent framework to the development of DSS systems within the environmental context. The paper begins by addressing rapid application development as a potential methodology to develop DSS, firstly presenting a definition of RAD and its relationship to dynamic systems development method (DSDM). The RAD tool-sets and components are then noted and summarized which are in turn analysed for their suitability towards DSS. The following section of this report then highlights emerging issues relating to decision support systems. This section also provides the background to the development of systems and underlines the problems that are inherent to information systems. The report then progresses on to describe flexibility, tailorability, end user design and users in order to encapsulate these issues into an appropriate framework. Lastly the report discusses integrated frameworks and presents a comprehensive discussion and graphical representation of the proposed RAD conceptual framework as depicted by the authors.

Section 2: Rapid Application Development

Essentially, RAD's contingent qualities have long been proposed as an approach to alleviate some of the problems found in traditional information system development (Martin 1991). The main concepts and techniques are outlined and discussed in this section. This paper believes that the approach is *adaptable* and *flexible* enough to cater for today's dynamic environment and as such aid in the development of DSS. In essence, we appropriate and update the approach from a systems engineered philosophy into a social technical systems perspective.

In its original form, James Martin introduced RAD as "a development lifecycle designed to give much faster development and higher-quality results than those achieved with the traditional lifecycle." (Martin 1991)

Paul Beynon-Davies *et al*, defines Rapid Application Development as "an important contemporary ISDM in that it uses techniques such as mixed development method that is particularly reliant on rapid development tools." (Beynon-Davies, Mackay et al. 2000)

RAD can be summarised as a methodology that enables organisations to develop important systems faster while reducing development costs and maintaining quality. It is from this context that RAD will be used as the basis of study, to address and counter the problems experienced by information system development projects, and therefore DSS.

The qualities of the RAD approach can be summarised as; a process through which the development cycle of an application is expedited. Martin identifies the key objectives of RAD as; high quality systems that are developed and delivered in short periods of times that in turn reduces production costs (Martin 1991; Beynon-Davies, Carne et al. 1999; Beynon-Davies, Mackay et al. 2000). RAD thus enables quality products to be developed faster, saving valuable resources. The key qualities mentioned above also addresses the software project management issues of time, quality and cost (termed as the triple constraint) (Yeates and Cadle 1996, pg, 149) and can be summarised as "the commercial need to deliver working business applications in shorter time-scales and for less investment" (Beynon-Davies, Carne et al. 1999).

It is our contention that the RAD literature has ill defined the difference between RAD as an approach and the frameworks and methods to conduct RAD projects, which this paper has identified as a gap. As such we build upon the Dynamics Systems Development Method (DSDM) whose objective aims was to produce a more disciplined directive to the development of systems that follow the RAD approach but does not specify how or when to apply the tools and techniques.

Essentially, the DSDM method adheres to several core principles, as discussed below, however, its un-formulised structure especially in terms of the 'fit' into the business environment has left a gap that this paper now seeks to fill.

This paper directs the development of RAD systems to a further refined framework approach with improved methods in terms of applying selected tools, techniques, components and levels of involvement of participants of RAD to DSS systems and therefore information systems in general.

For the purposes of this study we have abstracted only the principles of DSDM and RAD, presented below in a tabular format, in order to gain a wider perspective of the methodology we would direct you to the work by Beynon-Davies (Beynon-Davies, Carne et al. 1999).

The DSDM consortium was formed in 1994 to produce an industry standard definition of the RAD process (approach) and DSDM was subsequently defined. The DSDM framework

defines structure and controls to be used in a RAD project but does not specify a development methodology but instead incorporates the RAD approach. DSDM takes a fundamentally different perspective on project control. Rather than viewing requirements as fixed and attempting to match resources to the project, DSDM fixes resources for the project, fixes the time available and then sets out to deliver only what can be achieved within these constraints. Also the issues of quality may not be plausible in the time scale of a DSDM project; this is not to say that it is completely irradiated from the process but instead a certain level of imperfection is acceptable. "Software has to be 'good-enough' – and no more or less" (Stapleton 1997). DSDM is based on a number of underlying principles that can be used as a guide in order to conduct a RAD project. A summary of the nine principles as described by Stapleton and Beynon-Davies *et al* has been described below:

1. Active user involvement in the system development process is imperative.
2. DSDM team members must be empowered to make decisions.
3. The focus in a project is on frequent delivery of products rather then on activities.
4. Every deliverable is fit for its business purpose.
5. Iterative and incremental development is a powerful way to build systems.
6. All changes are reversible no requirements are frozen.
7. Base-lining of high-level system requirements.
8. Testing is integrated throughout the development cycle.
9. A collaborative and co-operative approach to development is essential.

Components of RAD

In this section we summarise the components of RAD, giving an outline of the main components in order to clarify where and how they correspond to the proposed framework.

The use of RAD and its components varied from project to project although the underlying principles remain. This is not uncommon as with most information system development methodologies, organisations adapt them to suit their own personal needs (Fitzgerald 1997). The RAD features and components were abstracted in order to gain clarity and to identify the potential components that could be used in relation to the development of DSS projects. The below table was formulated from the information as presented by Martin, Stapleton and Beynon-Davies.

PROJECT FEATURES	
Joint Requirements Planning (JRP), Joint Application Design (JAD) workshops	• Used at various points in the project • To elicit requirements • Key users, the client, some developers and a scribe to produce the system scope and the business requirements under the direction of a facilitator. • Business requirements produced within 3-5 days
Siting of Projects (Clean rooms)	• JAD workshops to take place away from the business or developer environment • Free from every day work interruptions • Highly focused problem solving • Office utilities should be made available e.g. desks and computers
Type of project	• First type: the Intensive - Team of developers and users utilise a clean room for several weeks to produce a working deliverable at the end of that time. • Second type: a Phased project - Spread over several months - Frequently organised in terms of the delivery and demonstration of three incremental prototypes - Aim to refine prototype into something that is deliverable
Project length	• Relatively small scale and of short duration • 2-6 months normal project length
TEAM FEATURES	
Team size	• Small development teams: 4-8 people with a maximum of 10
Team composition	• Both developers and users • The team is empowered to make design decisions • Users frequently act as managers of projects
Team skills	• Team members must have communication skills • Users require a detailed knowledge of the development area • Developers to be skilled in the use of rapid application tools
Team Building	• Extra-curricular activities to encourage 'team spirit'

PRODUCT FEATURES	
Rapid tools	• Fourth generation languages (4gls) • Graphical user interfaces (GUI) • Database management systems (DBMS) • Computer aided software engineering (CASE) tools
Systems Complexity	• Build applications that are not complex computationally
System interactivity	• Projects that are highly interactive and have a clearly defined user group
PROCESS FEATURES	
Focus on deliverables	• The focus is on the products or deliverables of the development process • Core functionality – avoiding copper plating
Timeboxing	• Project control: scoping the project by prioritising development and using negotiated delivery deadlines or 'time-boxes' • Reducing the requirements to fit the time-box if project is slipping
Incremental prototyping	• Inspection-discussion-amendment is usually repeated three times in RAD projects until user is satisfied with the system • Prototyping is used throughout the development lifecycle
User Involvement	• Users should focus only on project work for the duration of the RAD project • RAD work should be interleaved with other work
Developer involvement	• Developers should solely focus on RAD project but the literature suggests that such involvement will typically be interleaved with other development work
User-developer interaction	• Interaction between the two parties will vary according to the project, however Martin suggests that there should be high user involvement from the start of the project (Martin 1991; Stapleton 1997) • Formal and informal meetings – example 'wash-up' sessions on a weekly/monthly basis

Table 1 *– Showing components of RAD*

The above components can be adopted by any of the RAD methods i.e. DSDM, however it should be noted that these methods do not specify any fixed tools to develop a RAD system, therefore given rise to RAD projects that use different methods, components and techniques.

The components of RAD that have been adopted and even in some cases adapted to suit the required needs of decision support systems are described further in the report. However in brief the following components have been analysed and were incorporated into the proposed development models; JRP/JAD workshops, clean rooms, team compositions, team skills,

rapid tools, time-boxing, incremental prototyping, user involvement and developer involvement.

Decision support system and its relationship to methodologies has not been emphasized in literature and as such DSS are not constructed with any formal method. However DSS being a form of information system will suffer from the same problems inherent to such systems. The problems identified by Paul and the general statement that all systems disappoint (Paul 1994) will therefore effect DSS. RAD development has been offered as an alternative to the conventional lifecycle model (traditional waterfall model) to alleviate some of these problems, however if RAD is not utilised correctly then it also will be likely to fail. As described later in the paper the proposed RAD model uses various tools, techniques and methods in an attempt to overcome these issues, as well as trying to develop applications faster, cheaper and with less maintenance work.

Section 3: Decision Support Systems

DSS need to reflect the environment in which they operate. In the previous section we have established that RAD has the capability as apposed to more traditional structured approaches to adapt and respond to users needs.

According to Gorry and Scott Morton, DSS is a system that supports managers in unstructured decision-making situations (Gorry and Morton 1971).

Arun Sen has appropriately defined DSS systems and its use in today's society, describing the wide and diverse use DSS is put to. Drawing upon his research of low-cost and rise of remote computing (Sen 1998). This reflects a very different practice to the mainframe centralised planned system of information system development, to nowadays encompassing networked desktop PC's and artificial intelligence techniques.

DSS as a class of computer-based solutions have their own unique characteristics (Ho and Sculli 1995), which necessitates their development being different from traditional system development lifecycle approach. Because of the semi-structured or unstructured nature of problems addressed by DSS, managers perceived needs for information will change and so DSS must also change in order to meet users new requirements. This problem occurs and affects majority of information systems, however it is appreciated that in DSS the changes may occur more so and that the impact can be much greater. This problem may in turn have the knock on effect of systems resulting in short life-spans that lose their utility quite rapidly, therefore becoming a system that is incorrect in its decision making capabilities and therefore disallowing the organisation to progress (Dennis, Quek et al. 1996).

The traditional system development life cycle approach has given way to prototyping and rapid application development (Dennis, Quek et al. 1996). Developers constantly have to deal with the dilemma of deciding whether revisions to systems (maintenance or enhancements) should be completely rewritten in the latest version, or with another tool or language or even the reinvention of an entire system. This dichotomy of innovation on one end and continuity on the other highlights the challenges faced when developing DSS in a

rapidly changing business environment and technological advances. For DSS to be viable and popular (i.e. in terms of cost, time and quality), the choice of the right development strategy and tool become ever more important to determine their success as the application requirements also become more complex (Dennis, Quek et al. 1996).

Within this section the link between RAD and DSS has been established with the need to accommodate the environmental continual change. Establishing the foundations upon which this paper is building and extending the RAD and DSDM existing framework.

Section 4: Flexibility and *evolutionary* approach for users

Fitzgerald's review of software development (1996) distinguished the difference between 'hard' and 'soft' approaches of information system development. The research findings highlighted that information system development is underpinned by a fundamental difference between philosophical, technique and methods employed (Fitzgerald 1996).

Galliers suggested that the quest to link the so called 'soft' approaches dealing with managerial complexity with that of the structured information system design methodologies is in fact 'misplaced'; what is required is a focus on the development of flexible systems (Galliers 1993).

The predisposition to flexible information system development is the adoption of a strategy of 'modularization' (Hanseth and Monteiro 1998). Flexibility as used in the technical information system domain is defined as "the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed" (1990) for our work the term lacks the fullness and the naturalistic inclusions of the softer approaches, as it primarily addresses the technical qualities of system design. For a fuller inclusion, and perhaps, a more appropriate term we considered two terms, firstly, *adaptability*; orginaly defined as "the ease with which software satisfies differing system constraints and user needs" (Evans and Marciniak 1987) and secondly, *tailorable* aptly put by Stiemerlimg et al as "a tailorable software system can be defined as a system, which can be appropriately adapted to changing or diversified requiments" (Stiemerlimg and Cremers 1998). Evaluating the two terms we can express that a flexible information system development can be viewed as the union between *adaptability*, the modification of system component design and *tailorability*, referring to the more holistic organisational aspects, see figure 1

Given the expressed architecture design we are now able to further expound the complete methodological approach toward the realisation of the system. Which consists of three aspects. Firstly, analysis and understanding of the problem situation. Secondly, a set of methods and structures for the implementation of a DSS system, and lastly, metrics to assess technology flexibility. This paper primarily concentrates upon the first two issues, which are tightly integrated into our framework.

Firstly analysis, in this paper we adopt Guy Fitzgerald's 1990 paper entitled: "Achieving flexible information systems: the case for improved analysis" framework discussed briefly below. The broad aim of which, ensures that any action undertaken, pursues the goal of continual flexibility. The term analysis as used here has been developed to be inclusive in a context of user-initiated request for change. The principal adopted is that the control,

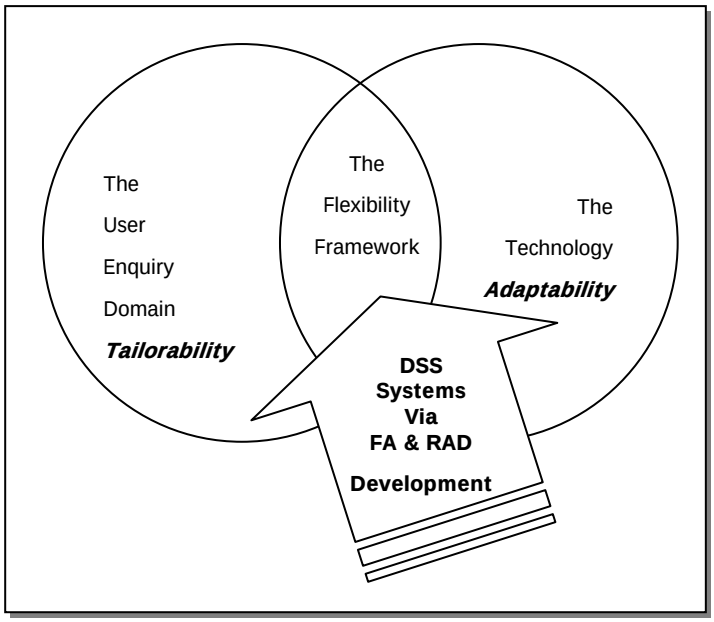

Figure 1*: Bridging and Integration of the IS – IT Flexible architecture.*

initiation and the start up management of the process is user led. The fit of this process to that of the organisational systems of control is subject within the FA analysis stage. The process may initiated by a simple two-line request via an E-mail, or a complete formalised document procedure. By whatever process, FA is initiated and led by the user, but organisationally managed within the proposed architecture.

Secondly; a set of methods, tools and structures for implementation; here we propose that flexibility is delivered via two methods, the reconfigured sympathetic framework of RAD as previously discussed, which acts as a monitoring and control mechanism. Together with a layered architectural approach of partitioning using connection protocols rules. Which we outline the current debates below and in the next section expand the framework analysis.

Thirdly, a set of metrics to assess technology flexibility, this paper does not address this issue but directs you to the work undertaken by Nelson, K. M. and M. Ghods paper (1998).

The adoption of a flexible system strategy 'naturally' suggests an *evolutionary* approach and supporting framework, as suggested by Paul, as a departure from the traditionalist life cycle model. Where it was argued that it was time to move away from Software Engineering (Paul 1994). Paul suggestion is that information systems design should be replaced by a core system that reflected an organic constituency and growth pattern (Paul 1994). Increasingly evidence of modularisation and hence flexible evolutionary system development can be found in frameworks for integrating components, applications and databases across interfaces enabled through standardisation of Intranet and Internet protocols. Such

standardisation supports a framework of *interoperability* and *reusability* of distributed objects by allowing developers to build systems by assembling reusable components from different vendors that communicate in a Distributed Computing Environment (DCE).

The technical qualities of which are mainly quantifiable see table 2. This papers concern is found in the fact that as developers build systems using pre existing components, benefits of maintainability and adaptability are potentially achieved.

Quantifiable Measures	Interoperability Portability Scalability Security Maintainability Complexity Throughput

***Table 2**: Adapted from The Software Engineering Institute: Distributed Computing Environment*

Figure 2 (Layering) depicts the architectural integration model, providing a framework for the integration of the patterning and software components. Components are becoming the standardised and integral to the building of software systems, as bricks and timber prefabricated parts are to architects. Cheesman and Daniels have defined components as a set of 'component forms' with an interface that defines a set of behaviours (Cheesman and Daniels 2000).

Various layering architectures have been suggested, which facilitate business patterning, Paul Allen typically suggests three layers

User interface – Business layer – Technical infrastructure layer (Allen 2001). However, to allow a sufficient level of abstraction for the common features between different aspects of an organisational environment that use DSS we propose a five level architecture. By doing so the empirical question of 'user requirement analysis' becomes inverted to 'user led design' as users initiate adaptation through Tailorability and End User Computing. See Buschmann for a full discussion and a list of the issues and criteria that need to be addressed for architectural design (Buschmann, Meunier et al. 1996).

Authors have already suggested ways to facilitate requirements gathering under organisationally dynamic conditions using Tailorable operations upon system architecture (Patel, Gardner et al. 1995; Patel 1997; Stiemerling, Kahler et al. 1997; Mørch and Mehandjiev 2000; Stiemerlimg and Cremers 2000). However, what is generally lacking is the project management control mechanisms that assess the assets of cost quality and time implementations in the business environment. As such we are suggesting that by the inclusion of FA and RAD approaches many of the issues of the metrics of accountability can be addressed.

One possible path that seeks such an approach re-conceptualises 'design' to be deferred to the user, proposed by Patel in Deferred System's Design (DSD) (Patel 1999). Another root proposed here uses an integrated framework between the human and technology It also requires a mechanism (FA) to ensure ongoing growth. With such a system in place DSS are deferred to users, supports the knowledge worker, her skills and work practices routines, as identified and termed as 'reflective' by Donald Schön (Schön 1991).

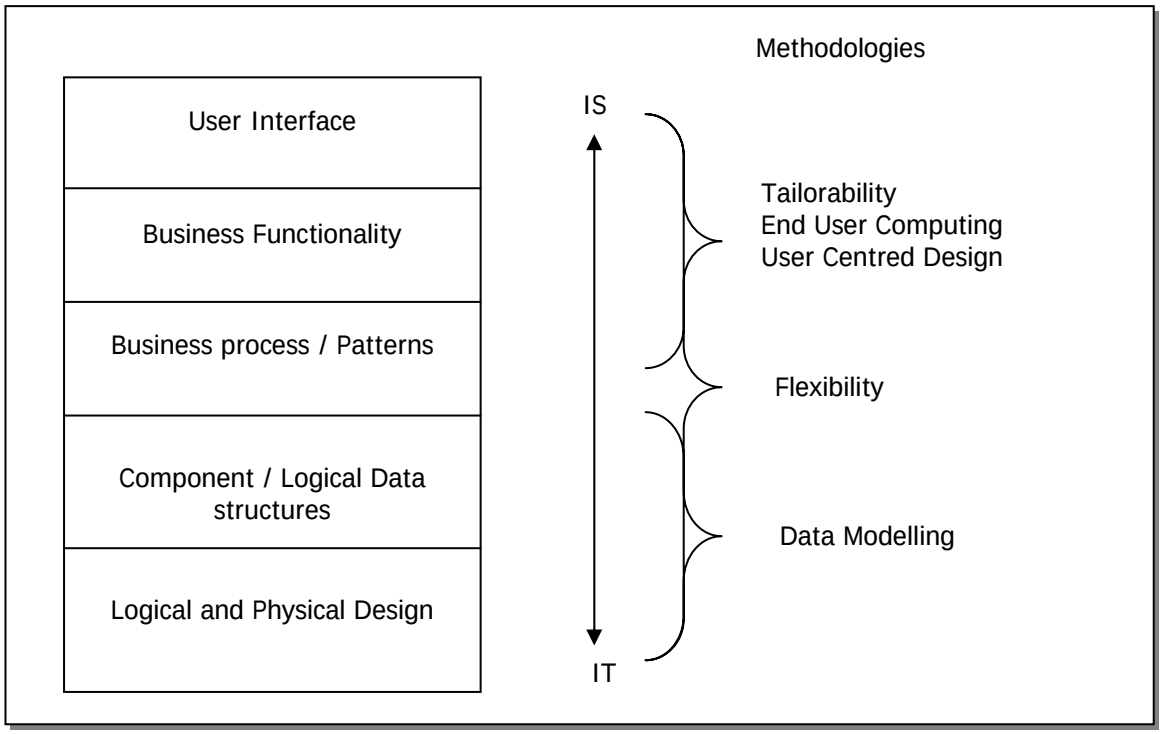

Figure 2: *Architecture layering of the component development approaches*

Lars Mathiassen's article on Reflective Systems Development has similarly reflected that information system development needs facilitated the reflective practices of the modern knowledge worker (Mathiassen 1998). Devolved management structure that support reflective practices and especially the rise of the knowledge worker has led to 'conflicts' of islands of knowledge, which has forced towards a pluralistic form of organisation (Scarbrough 1999). This potentially leaves the organisation with an information system infrastructure that is similarly fragmented and distributed. DDS is the natural bridge and provides the interconnected fabric to the organisation structure, as organisations merge into what Mintzberg coined 'operating adhocracy' (Mintzberg 1983). And FA addresses the strategy for adaptation and 'fit' as it forms a part of the organisations 'survival kit' (Avison, Powell et al. 1995) for ongoing change.

Flexibility Analysis has been developed from empirical work as it acknowledges that it is desirable and indeed necessary to accommodate change in information system design (Fitzgerald 1990). This paper develops a potentially promising framework using RADs. The paper now outlines the development and general principles of FA, for a fuller exposition see Fitzgerald, 1990 and Fitzgerald *et al*, 1999.

Fitzgerald, 1990 and Fitzgerald *et al*, 1999 suggests that FA is achievable through the maintaince phase of the lifecycle, but particularly through the preventive maintenance

approach (Lientz, Swanson et al. 1978). But both papers state that greater beneficial results would be obtained by building in flexibility; 'at source rather than adding it once a system has been developed' (Fitzgerald, Philippides et al. 1999), this is the primary aspect that is developed further in this paper.

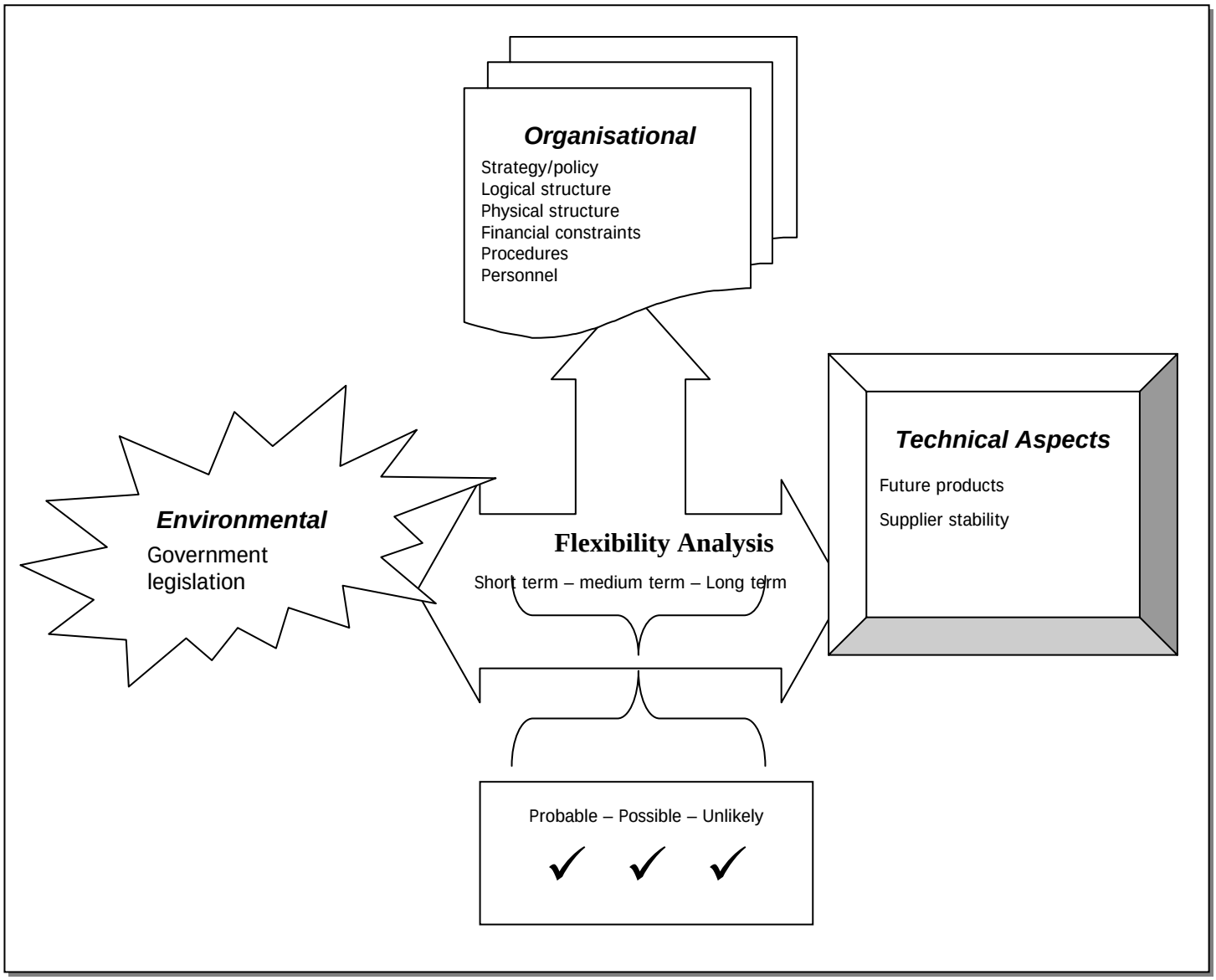

Figure 3: *Flexibility Analysis adapted from Fitzgerald, 1990.*

In Avisons *et al* 1995 paper it was suggested that FA could be achieved through incorporating FA into a modified information systems development methodology Multiview. As a way for identifying ***future*** requirements. As the above discussion has indicated, such a fundamental positional stance is questionable, since it invites the analyst to have accurately predicted the future.

Principally, FA as Fitzgerald, 1990 outlines is about 'identifying possible events or changes that might occur in the future' (Fitzgerald 1990). It starts by identifying factors that (may) influence the organisation see figure 3, which can be further subdivided. From an organisational perspective, an environmental and from the technical aspects. The FA

examines these potential impacts as well as time scale impacts, and produces an output of the - Probable – Possible – Unlikely outcomes.

In summation FA, as it is was originally conceived, covers:

1. Time - and development time scales

2. Supports software and software development procedures

3. A stage of systems development - along side prototyping

4. Examination of potential changes to the organisation / the business / the environment AND the time scale

(Fitzgerald 1990)

This section discussed the bridging and integration of the information system and information technology architecture, from the perspective of information system development catering for FA and USD with the aim of the realisation of adaptability and tailorability.

Section 5: Integrated bridging framework

In this section we propose an alternative strategy that instead of taking the 'current work practice' of preventive maintenance when system change is undertaken, DSS systems can be continually adapted and tailored using the flexibility approach and an updated RAD structure, which we now expand upon.

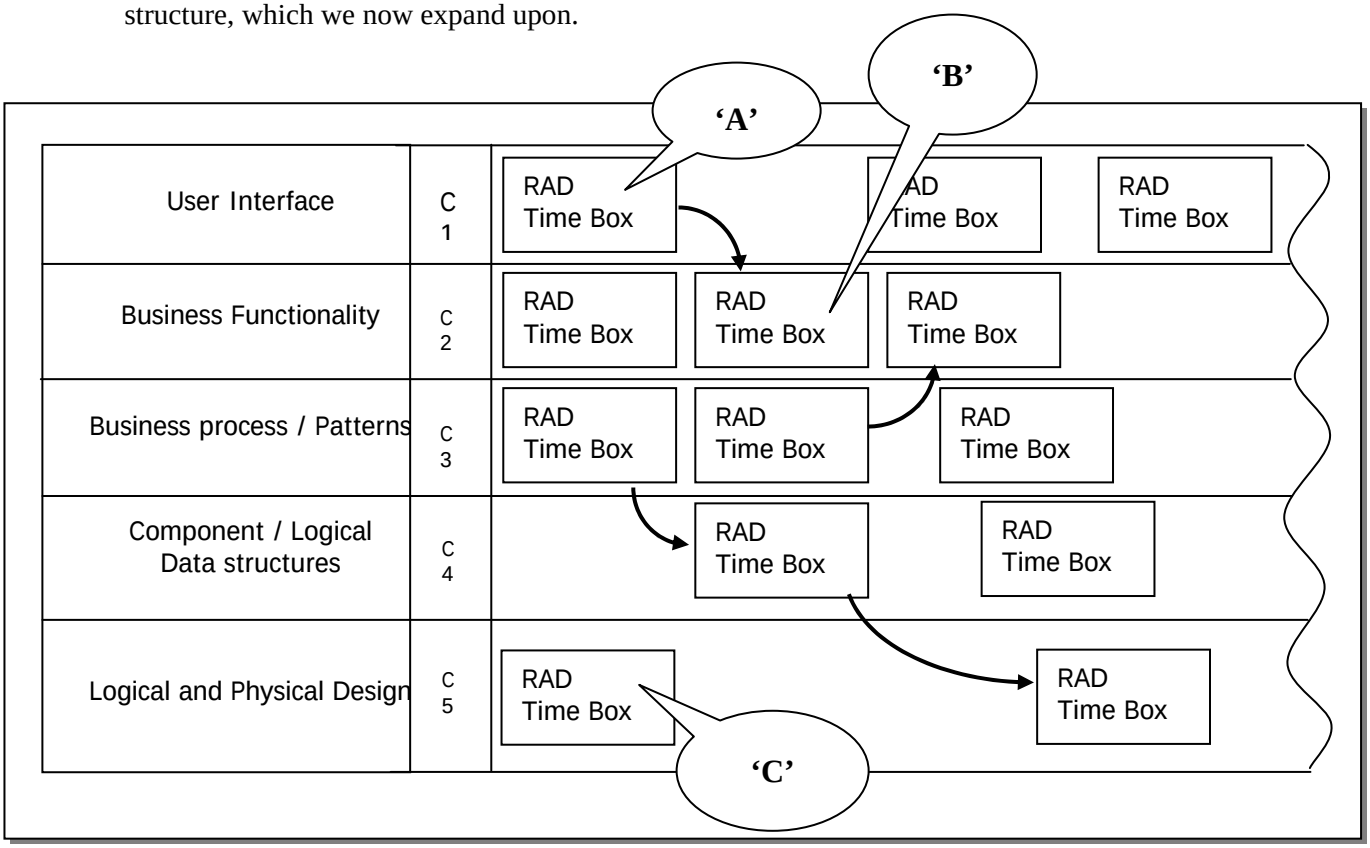

Figure 4: *RAD Layered Architecture*

In this section we address the details of the architecture and expand the framework that has been previous outlined.

This paper defines the RAD Time Box as a 'self contained, modular transformational process' with a distinct beginning and end outcome. Taking the overall picture as in figure 4 we can see that, each RAD Time Box process being self-contained and is isolated from the whole 'environment', that is each level is an abstraction and a particular perspective of the system. Such that it is capable of being initiated at any level of the architecture. For example RAD Time Box 'C' is a potential starting position from a technical perspective.

Each RAD Time Box is related to it own architectural layer. And the output of each RAD Time Box is bi-directional, for example, the RAD Time Box 'A' at the user interface architecture in Figure 4 has two possible outputs. Firstly it gets resolved at that level, therefore the user interface has been altered and the requirements met. Alternatively the output has implications for the business functionality, therefore the output of 'A' becomes in input requirements documentation for RAD Time Box 'B'.

RAD Time Box 'B' is bounded by its level. In other words, an initiated Business Functionality alteration will only be considered as inputs for RAD Time Box's above it (User Interface), below it (Business process/Patterns) or requiring further RAD Time Boxing at its current level.

At the same level of architecture abstraction, we can also see the time, cost and quality dimensions. See figure 5 together with the selected RAD components.

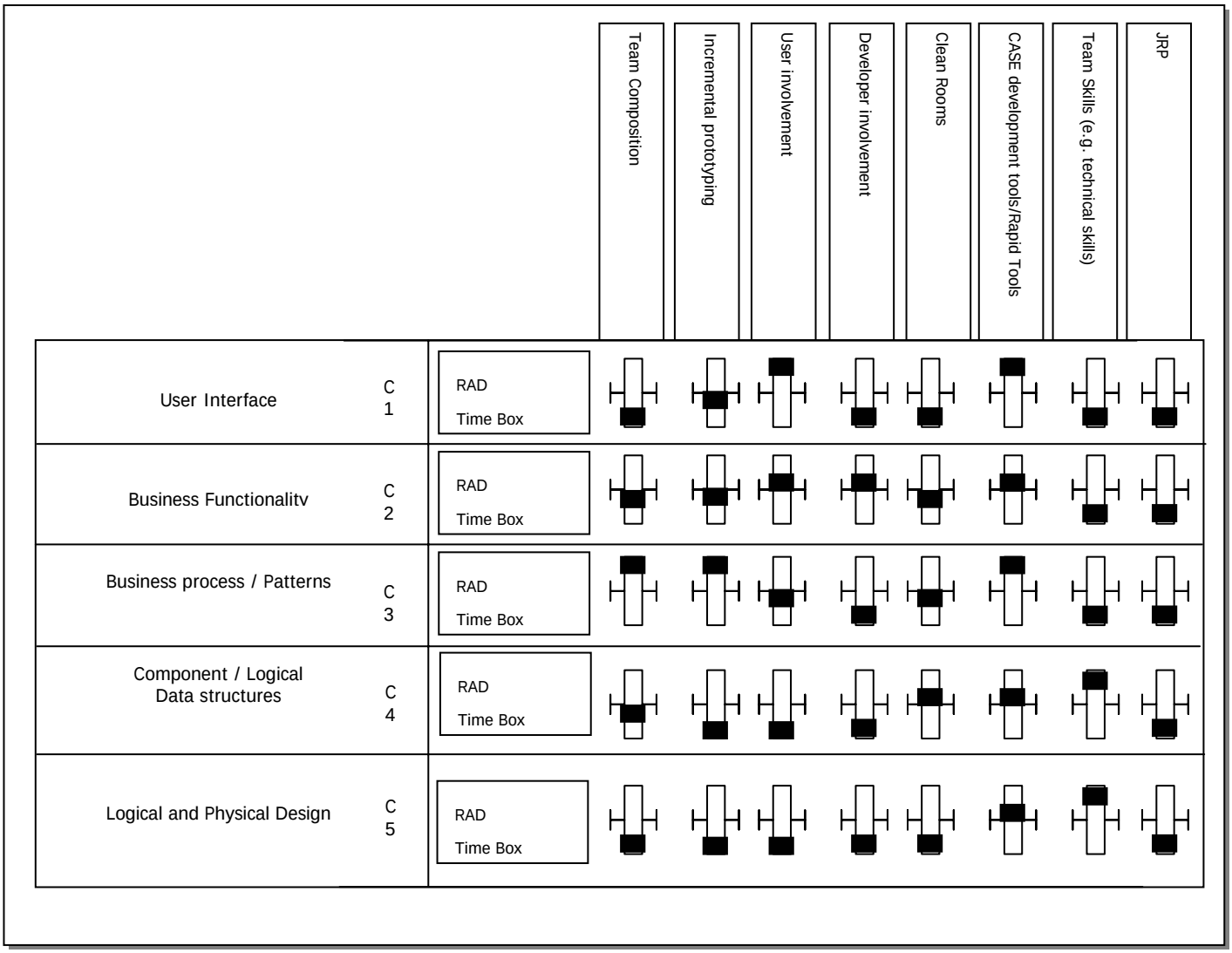

Figure 5: *RAD Compositional Tool Set Use – Composition and Cost Structure*

Firstly cost; we recognise that at each level of the architecture has inherently a different cost structure, as each has its own generic composition. For example, each level has a different mix of people in regard to skill sets. Therefore it will require different levels of resources and different procedures.

In line with the RAD methodology the Time Box restraints is fixed hence it caters for accurate planning, and hence constrains Time.

Quality has been defined as `fitness for purpose' (Juran 1979), also a widely used definition has been supplied by the International Standards Organization (ISO 1986):

``The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs.''

Our framework facilitates quality by fine-tuning and adjustments of the tools, techniques and team composition. Figure 5 demonstrates the appropriate mix of the RAD components at the differing levels of architecture. This fine-tuning of each individual matrix constituent enhances quality assurance.

Analysis of the RAD Time Box, which makes up the template of the core framework and

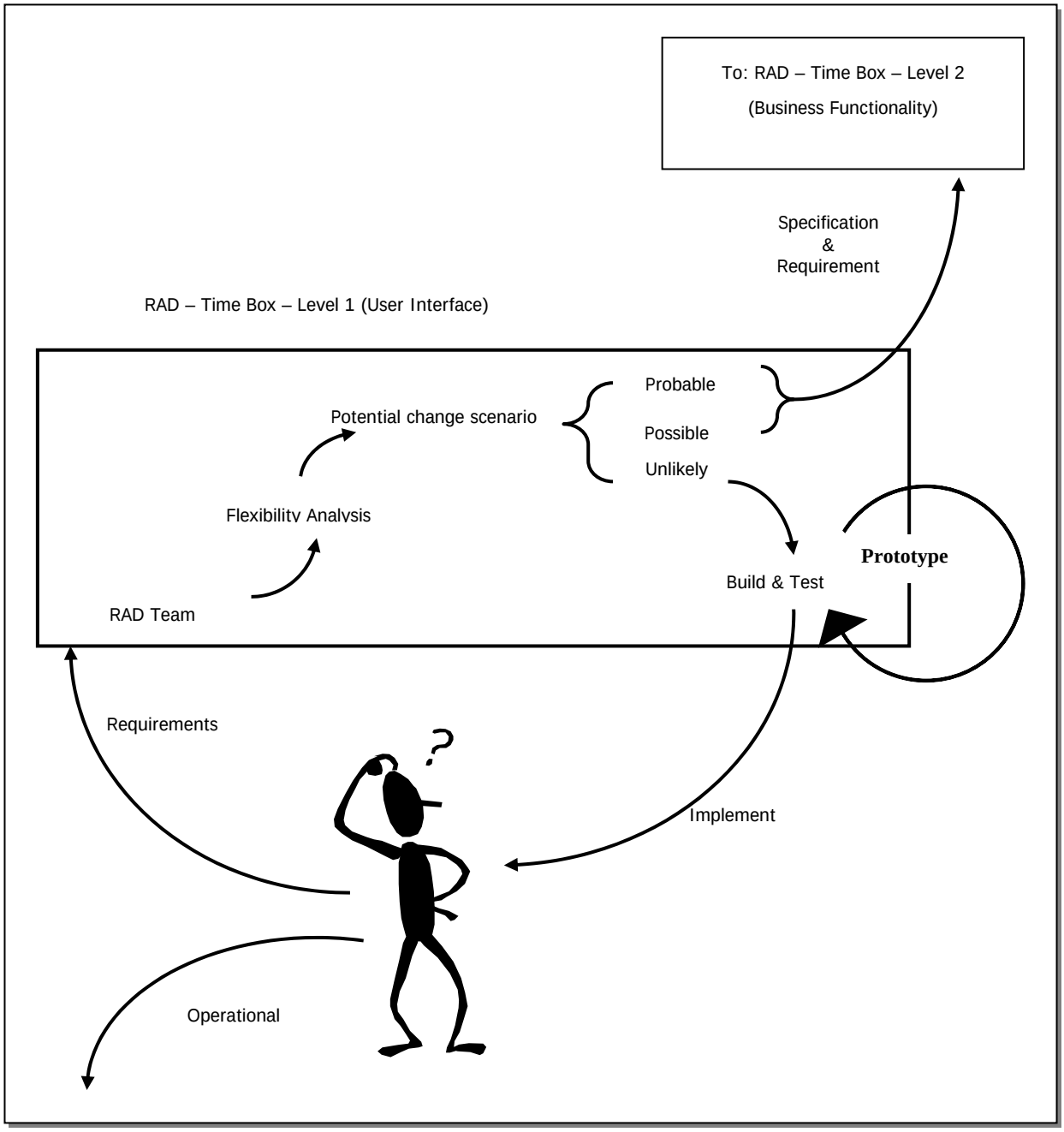

Figure 6: *Flexibility Analysis in RAD Time Box*

cements FA and RAD Time Box together.

The process outlined here starts from the user initiating the requirements (figure 6) to a FA/RAD Team whose composition and make up match the skill set of that level. (In this case - C1-figure 5 the User interface layer). The team composition involves the collaborative work of several specialists with different areas of expertise with the co-

ordinating framework controlled within the RAD process time-box. As a reiteration, the framework proposed in this paper, the RAD process Time Box is an amalgamation of FA and the RAD methodological approach.

Figure 6 suggests a possible template scenario that is repeatable at all levels of the layered architecture figure 4 & 5. As the system is user led and initiated the FA team assesses the impact of the requirement (figure 6) upon the organisation. If the request has an 'unlikely' impact upon the rest of the organisation the team working within the Time-Box constraints of TCQ (Time, Cost, Quality cost structure C1 figure 5) prototypes, develops and implements the change.

Having undertaken a FA a Probable or a Possible outcome automatically triggers off and escalates into a requirement and specification contract to the RAD team at the next layer. The advantages of such an approach ensures that the correct skill set can be utilised, the demands upon the software development team are controlled by the RAD time Box, finally, the user initiates and generates the need. It is envisaged that over time the historical RAD time-box record attributes feeds into a future estimation process.

Section 6: Conclusion

This paper assessed the suitability of the RAD methodical approach to address the issues of ongoing complexity and information system design realisation. This was achieved through the extension and development of the existing RAD and DSDM approach. The new extension to the framework introduced FA, UCD and the modern modular architecture. Thus, not only addressing and bridging the project managerial issues of cost, time & quality, but also within the context of formulating and producing DSS.

In summary, the paper firstly addressed RAD as an approach and DSDM as a method, focusing upon the elicitation of potential RAD components. The paper then proceeds to discuss DSS in relation to inherent development problems, reflecting the wider context of information system development in general. The third aspect of the theoretical construct progresses on to describe flexibility, tailorability, end user design and users. Which encapsulates the issues into an appropriate framework. Lastly the report expands upon and discusses the new proposed integrated framework, presenting a comprehensive discussion and graphical representation of the proposed amalgamated RAD conceptual framework.

The advantages realised we can summarise as:

1. The suggested framework of this paper, helps to alter a perceptional change on how information system developers perceive system development, by taking RAD out of the structured development approach to seeing systems development that is semi-structured or even unstructured, and therefore caters for environmental continual change.
2. The analysis stages of the life cycle is encapsulated within the RAD Time Box, and is not seen as independent process.
3. RAD has tried and tested history, tools and methods. Making it an ideal candidate.

4. The framework as proposed removes the traditional perspective of hard and soft. And as such the recognised issues of bridging information systems development and that of the business functionality has changed to an integrated process approach.
5. The layering addresses the single analysts' perspective and replaces it with a team with the relevant detail of knowledge, who undertake a FA to decide if it is necessary to escalate out of the RAD Time-Box without resorting to the employment of the generality specialist continuum issues.

With the introduction of the proposed framework the triple constraints of time, cost and of quality are considered together with the impact of development upon the organisation. However the main advantages are seen to be the user centred approach.

References:

IEEE Standard Computer Dictionary.(1990) A Compilation of IEEE Standard Computer Glossaries. New York, NY, Institute of Electrical and Electronics Engineers.

Allen, P. (2001). Realizing e-Business with components, Addision-Wesley.

Avison, D. E., P. L. Powell, et al. (1995). "Addressing the Need for Flexibility in Information Systems." Journal of Management Systems **7**(2): 43-80.

Beynon-Davies, P., C. Carne, et al. (1999). "Rapid application development (RAD): an empirical review." European Journal of Information Systems **8**: 211-223.

Beynon-Davies, P., H. Mackay, et al. (2000). "'It's lots of bits of paper and ticks and post-it notes and things...': a case study of a rapid application development project." Information Systems Journal **10**: 195-216.

Buschmann, F., R. Meunier, et al. (1996). Pattern Orientated Software Architecture. Chichester, John Wiley.

Cheesman, J. and J. Daniels (2000). UML Components, Addison-Wesley.

Dennis, A. R., F. Quek, et al. (1996). Using the Internet to implement support for distributed decision making. London, Chapman & Hall.

Evans, M. W. and J. Marciniak (1987). Software Quality Assurance and Management. New York, NY, John Wiley & Sons.

Fitzgerald, B. (1996). "Formalized systems development methodologies: a critical perspective." Information Systems Journal **6**: 3 - 23.

Fitzgerald, B. (1997). "The use of systems development methodologies in practice: a field study." Information Systems Journal **7**: 201-212.

Fitzgerald, G. (1990). "Achieving flexible information systems: the case for improved ananlsis." Journal of Information Technology **5**: 5-11.

Fitzgerald, G., A. Philippides, et al. (1999). "Information system development, maintenence and enhancement: findings from a UK study." International Journal of Information Management **19**: 319-328.

Galliers, R. D. (1993). "Towards a flexible information architecture: integrating business strategies, information systems strategies and business process redesign." Information Systems Journal **6**: 131 - 146.

Gorry, G. A. and S. Morton (1971). "A framework for management information systems." Sloan Management Review: 55-70.

Hanseth, O. and E. Monteiro (1998). CHAPTER 5 Openness and flexibility, Understanding Information Infrastructure: Manuscript 27. Aug., 1998.

Ho, J. K. K. and D. Sculli (1995). "System Complexity and the Design of Decision Support Systems." Systems Practice **8**(5).

Juran, J. (1979). Quality Control Handbook, McGraw-Hill.

Lientz, B. P., E. B. Swanson, et al. (1978). "Characteristics of Application Software Maintenance." Communications of the ACM **21**(6): 466-71.

Martin, J. (1991). Rapid Application Development. New York, Macmillan.

Mathiassen, L. (1998). "Reflective Systems Development." Scandinavian Journal of Information Systems **10**(1&2): 67 - 117.

Mintzberg, H. (1983). Structure in Fives: Designing Effective Organisations. Englewood Cliffs, NY, Prentice-Hall.

Mørch, A. I. and N. D. Mehandjiev (2000). "Tailoring as Collaboration: The Mediating Role of Multiple Representations and Application Units." Computer Supported Cooperative Work **9**: 75-100.

Nelson, K. M. and M. Ghods (1998). "Measuring technology flexibility." European Journal of Information Systems **7**: 232 - 240.

Patel, N. V. (1997). Tailorable Information Systems: Deferred System Design for Changing Organisations. BIT '97 Proceedings of the 7th annual BIT conference - Business information management: alternative futures, Manchester Metropolitan University.

Patel, N. V. (1999). "The Spiral of Change Model for Coping with Changing and Ongoing Requirements." Requirements Engineering **7**: 77 - 84.

Patel, N. V., L. A. Gardner, et al. (1995). Moving Beyond the Fixed Point Theorem with Tailorable Information Systems. Proceedings of 3rd European Conference on Information Systems.

Paul, R. J. (1994). "Why Users Cannot 'Get What They Want'." International Journal of Manufacturing System Design **1**(4): 389 - 394.

Scarbrough, H. (1999). The Management of Knowledge Workers. Rethinking Management Information Systems: An Interdisciplinary Perspective. W. Currie and R. Galliers. Oxford, Oxford University Press**:** 474-496.

Schön, D. A. (1991). The Reflective Practitioner, How professionals think in action. Aldershot, Ashgate Arena.

Sen, A. (1998). "From DSS to DSP a taxonomic retrospective." Communications of the ACM **41**(5es).

Stapleton, J. (1997). DSDM Dynamic Systems Development Method. Harlow, England, Addison-Wesley.

Stiemerlimg, O. and A. B. Cremers (1998). Tailorable component architectures for CSCW-systems. Proceedings of the Sixth Euromicro Workshop on Parallel and Distributed Processing PDP '98.

Stiemerlimg, O. and A. B. Cremers (2000). "The Evolve Project: Component-Based Tailorability for CSCW Applications." AI & Society **14**(1): 120-141.

Stiemerling, O., H. Kahler, et al. (1997). "How to Make Software Softer -Designing Tailorable Applications." ACM.

Yeates, D. and J. Cadle (1996). Project Management for Information Systems. London, Pitman Publishing.