December 2000

Managing Innovative IS Projects in Dot.com Companies

Eric Deakins
University of Waikato

Stuart Dillon
University of Waikato

Managing Innovative IS Projects in Dot.com Companies

Eric Deakins[1], Stuart M. Dillon
Department of Management Systems
Waikato Management School
University of Waikato
Hamilton, New Zealand.

Abstract
The key to success for contemporary software manufacturers competing in a virtual marketplace is the ability to build market share quickly via rapid time-to-market of high quality, innovative software solutions that delight every customer. Young dot.com companies in particular require an Information Systems Development Methodology (ISDM) that is simple to administer, is cognisant of the limited resources available, yet encourages the creation of exciting and relevant high quality products on time and within budget. In particular, it must allow late design changes to be incorporated in response to new competitor products and emerging trends in electronic commerce. This paper describes field research that led to a modified spiral development methodology for creating innovative, high quality web-integrated software products under severe time and resource constraints. Advantages and limitations of the approach are presented and its usefulness for similar projects is described.

***Keywords*:** Information Systems Development Methodology, Innovation, Project Management, Prototyping, Product Development.

1. Introduction

The key to success for contemporary software manufacturers competing in a virtual marketplace is the ability to build market share quickly via rapid time-to-market of high quality, innovative software solutions that delight every customer. Such success factors have created the need for an Information Systems Development Methodology (ISDM) that encourages experimentation and mass customisation, and also allows Just in Time (JIT) delivery of cutting edge solutions.

Some of the major players in the evolving Internet environment exist only in hyperspace. Usually referred to as 'dot.com' companies, they compete against larger competition by using guerrilla tactics to survive and thrive. Such start-up companies frequently own few physical assets or capital and may be run by knowledge professionals with little more than a 'good idea' for a marketable software product or software-based service. Their particular ISDM requirement is for a systems development/management approach that is simple to administer and is cognisant of the limited available resources, yet also encourages the creation of exciting and relevant high quality products on time and within budget.

In addition to electronic commerce (E-commerce) transactions and direct software downloads, the World Wide Web (web) provides opportunities for software manufacturers to delight their customers via direct enhancement of software products, on-line maintenance and support, and product updates. Customers too are increasingly willing to assist with the development effort. For example, more than 650,000 customers tested a beta version of

[1] Dr. Eric Deakins can be approached via e-mail: edeakins@waikato.ac.nz

Microsoft's Windows2000™ product in their own unique native environments, and shared with the software giant their ideas for changing some of the product's features (Prahalad and Ramaswamy, 2000).

Perhaps it is not surprising that ISDMs have failed to keep pace with such changes in the development environment. Software manufacturers routinely exceed time and cost targets causing the product to be late to market, or of low quality, with commensurate opportunity cost of lost sales and defections to competitors' products. Advances in systems development methodologies designed to remedy this situation are frequently confounded by rapid changes in project complexity (Blackburn *et al.*, 1996). In addition, emerging commercial uses of the Internet guarantee that the circumstance under which software products are developed will continue to change rapidly and dramatically.

This paper presents an approach to contemporary software systems development that evolved from fieldwork undertaken with a dot.com company during a live IS project. The paper is presented in the approximate order of actual events that took place. Thus, it begins with a brief review and appraisal of candidate development methodologies judged relevant at the time. Subsequent sections describe the goodness of fit between the chosen 'spiral' model and actual development activities that were used to deliver a high quality web-integrated software product. A modified spiral model is then presented that provides a superior fit between the conflicting needs for creative, cutting edge solutions to be developed under conditions of severe resource and time constraints. Advantages and limitations of the approach are discussed.

This paper has relevance for practitioners and academics who wish to understand the changing nature of the software product development environment as well as coping strategies.

2. Review of Current Information Systems Development Methodologies (ISDMs)

2.1 ISDM Requirements

In the last section it was argued that market-driven IS projects require an Information Systems Development Methodology that *promotes* design experimentation, Just in Time (JIT) delivery practices, and mass customisation. It was also discussed how dot.com software manufacturers (in particular) have an overriding concern to build market share quickly via rapid time-to-market of high quality, innovative software solutions that delight the customer.

It follows that an ideal ISDM for dot.com start-ups would encourage the following:

- Rapid development of high quality products
- Innovative and creative solutions via experimentation
- Continual improvement to product specifications via high levels of customer feedback and responsiveness to the external environment.

Before the dot.com company (subject) commenced any formal systems design work, the ISDM literature was reviewed for candidate methodologies and tools. The next section briefly summarises an appraisal that was used to select a suitable ISDM that would subsequently guide the activities of the software design team.

2.2 General Lifecycle Models

The IS literature is replete with ISDMs and it is not the intention of this paper to describe them all here. Sage (1995) highlighted 3 general lifecycle models (otherwise referred to as 'waterfall models') used for software development:

1. The *grand design* lifecycle model in which there is a single pass through each phase of development. The System Development Life Cycle (SDLC), illustrated in Figure 1, is a well-known example (Walters *et al.,* 1994).

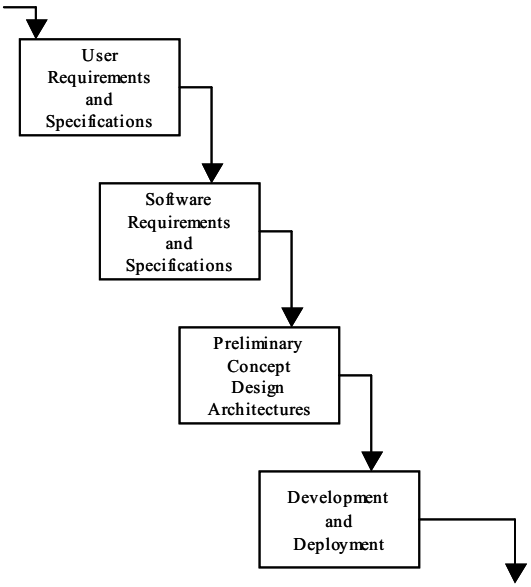

Figure 1. SDLC: example of a 'grand design' lifecycle model

2. The *incremental* lifecycle is one in which there is an iterative sequence of builds with each successive build incorporating more and more of the user requirements, as indicated by the resulting system specifications. User requirements are transformed to system level requirements and then to software requirements and specifications. Once established the software specifications do not change except as part of maintenance after the software product has been deployed.
3. The *evolutionary* lifecycle model is one in which the software system results from a sequence of iterative builds, as in the incremental lifecycle. However, following the initial build a refined set of user needs and requirements is captured prior to each successive build (Sage, 1995).

2.3 Limitations of General Lifecycle Models for dot.com companies

Potential problems posed by the *grand design* lifecycle model include the expense, frequent complexity and, most critically, the time that it consumes (Veryard, 1985). These factors are especially relevant to dot.com companies. Evidently, the *grand design* lifecycle model, as it stands, is not well suited for market driven projects, since:

- The sequential nature of a *grand design* reduces the ability to incorporate changes in requirements that emerge beyond the requirements definition stage

- Poor linkages usually exist between analysis, design, and implementation often resulting in the development of a system that addresses few of the user needs
- Systems design tends to exclude users, leading to systems that do not match user needs
- Technical staff, who tend to be the main proponents and implementers of such models, often find it difficult to understand business requirements (Howard, 1997).

Such criticisms are not confined to software development in dot.com companies. For example, a recent field study of an IS development project within a large organisation concluded that such formal approaches "…are too mechanistic to be of much use in the detailed day-to-day organization of developer's activities" (Nandhakumar and Avison, 1999 p.188).

The *incremental* and *evolutionary* lifecycle models also appear to suffer from the same problems of excessive development time and expense. Although the iterative nature of their development is intended to elicit refinement of user requirements, it is difficult to see how such methods actively encourage creativity and JIT delivery practices during software development.

Table 1 provides a tentative appraisal of the ability of general lifecycle models to meet the software product development needs of dot.com companies.

Table 1. Value of general lifecycle models for dot.com software product development

	Value
Rapid Development	Low
Innovative and creative solutions	Low
Continual improvement to product specifications, via high levels of customer feedback and responsiveness to the external environment	Low/medium

2.4 Spiral Models

Although similar to the *evolutionary waterfall model*, in which the first build (product release) is generally refined and improved on subsequent builds, the *spiral* model of software development essentially represents several iterations of a waterfall lifecycle model. This provides the possibility of alternative approaches to software development at each iteration, e.g. (Boehm, 1988; Alonso *et al*., 1996), Figure 2. It is convenient to view each cycle of development in the spiral model as corresponding to one or more of the phases in a waterfall model, where the steps undertaken in each 'cycle or round' may be described as:

- Formulation, in which the most relevant approach for that particular phase under consideration is identified and defined
- Analysis, in which risk identification and analysis are conducted to determine an appropriate systems management plan for development
- Interpretation 1, in which the results of this risk analysis are implemented in the form of a specific set of product development plans
- Interpretation 2, in which various management reviews and plans are used to implement the next phase or to stop further effort, either if a high quality product has been produced or if further development is deemed inappropriate for some reason.

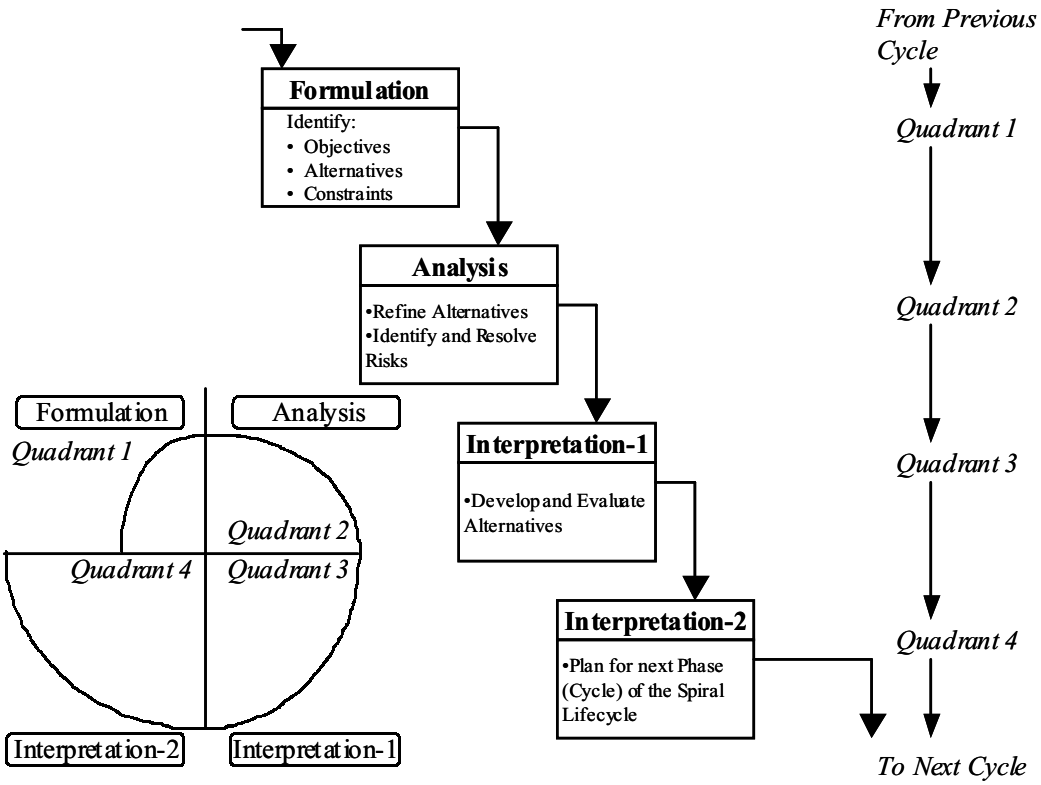

Figure 2. Spiral model of systems development

Spiral models incorporate prototypes that offer insight into the development task at hand. Prototyping is "…used primarily as a communication tool to assess and meet the information needs of the user" (Carey, 1990 pp. 119-20). This makes them especially useful for the software development needs of dot.com companies and should lead to improved requirements determination in a competitive, features-driven marketplace. However, this approach can prove frustrating not only for the developers but also for the unaccustomed user who may have unrealistic expectations based on what can quickly be done through several prototyping iterations. According to Carey there may also be a disproportionate emphasis on the user interface at the expense of other system requirements. Table 2 provides a tentative appraisal of the ability of spiral models (that utilise prototyping) to meet the software product development needs of dot.com companies.

Table 2. Value of spiral models for dot.com product development

	Value
Rapid Development	Low
Innovative and creative solutions	Medium
Continual improvement to product specifications, via high levels of customer feedback and responsiveness to the external environment	Medium/High

3. Team Dynamics and Creativity

Demands made on software development teams in dot.com companies are many and varied. Contemporary software *products* are required to be innovative, exciting, and good value as well as relevant to the end-user (customer); they must also be of highest perceived quality. In

addition, smaller dot.com software manufacturers are often required to handle their own marketing, support, and distribution of the finished product via evolving E-commerce channels (or to outsource). These demands call for truly creative and multi-skilled design teams.

It is interesting that while the IS literature frequently acknowledges the need for such creativity and innovation, e.g. (Blackburn *et al.*, 1996), well-researched methods for encouraging creativity during the software production process have received less attention. This situation may be contrasted with other highly dynamic manufacturing environments where creativity is also claimed to be the key to competitive advantage. For example, Feurer *et al*. (1996) describe Hewlett-Packard's framework for developing creative teams, which claims to produce superior manufactured products.

The realities of marketing a software solution via the Internet, where the aim is to diffuse the product quickly to achieve critical mass and capture share of the customers' mind, mean that today's dot.com companies must also address the problem of how to manufacture software *rapidly* without sacrificing either creativity or quality during product development. Studies have shown, Table 3, that most of the reduction in product development time can be attributed to the composition of the development team and the processes they employ (Blackburn *et al.*, 1996). Clearly such factors are critical to project success in dot.com companies.

Table 3. Reductions in software development time in 26 Japanese companies

	% Time Reduction	**Rank**
Prototyping	6.5	9
Comprehensive requirements determination	5.5	10
CASE tools	8	7
Concurrent development	12.5	2
Software quality practices	7.5	8
Design team composition	12.5	2
Good testing strategies	11	4
Reusable code	9.5	6
Modularisation	3	11
Intra-team communication	10	5
Programmer quality	14	1

While general principles of software quality improvement are reasonably well documented, market induced pressures have received less attention in the IS literature. In essence, this means that current software development models are rooted in an era when project control, rather than time to market, was the overriding concern. Clearly, a new systems development model is needed for developing innovative and high quality software products that can take full advantage of the creative talents of designers, while also being sensitive to the realities of the marketplace.

4. Information Systems Development Methodology – A Case Study

There has been a widespread call for fieldwork to be included as part of the development of ISDMs, e.g. (Orlikowski, 1993; Wastell, 1996; Fitzgerald, 1996). It is claimed that such an approach provides an effective method of data collection, especially for understanding the many complex human issues that are often involved (Nandhakumar and Avison, 1999). This section examines the degree of fit between a spiral ISDM model and the development processes used to create and deliver an actual software product in a dot.com environment. The study involves a small dot.com start-up company that was formed in 1998. The new product was to be a self-help Y2K assessment system that would enable small and mid-sized businesses to assess their Y2K exposure and track contingency plans without the need for expensive external consultants.

The project was begun in November 1998 in anticipation of a product release date at the end of the second quarter in 1999. Time constraints were very severe given the intended use for the product. The project team comprised one full-time member and three part-time members with skills in (traditional) project management, decision support system design, and website design. No E-commerce skills or product marketing skills were represented in the team and it was intended from the outset that specialised software coding skills would be purchased at an appropriate time. Given these constraints, a traditional project management software package was utilised to plan and communicate the future project. Figure 3 indicates the highly concurrent nature of the expected development activities.

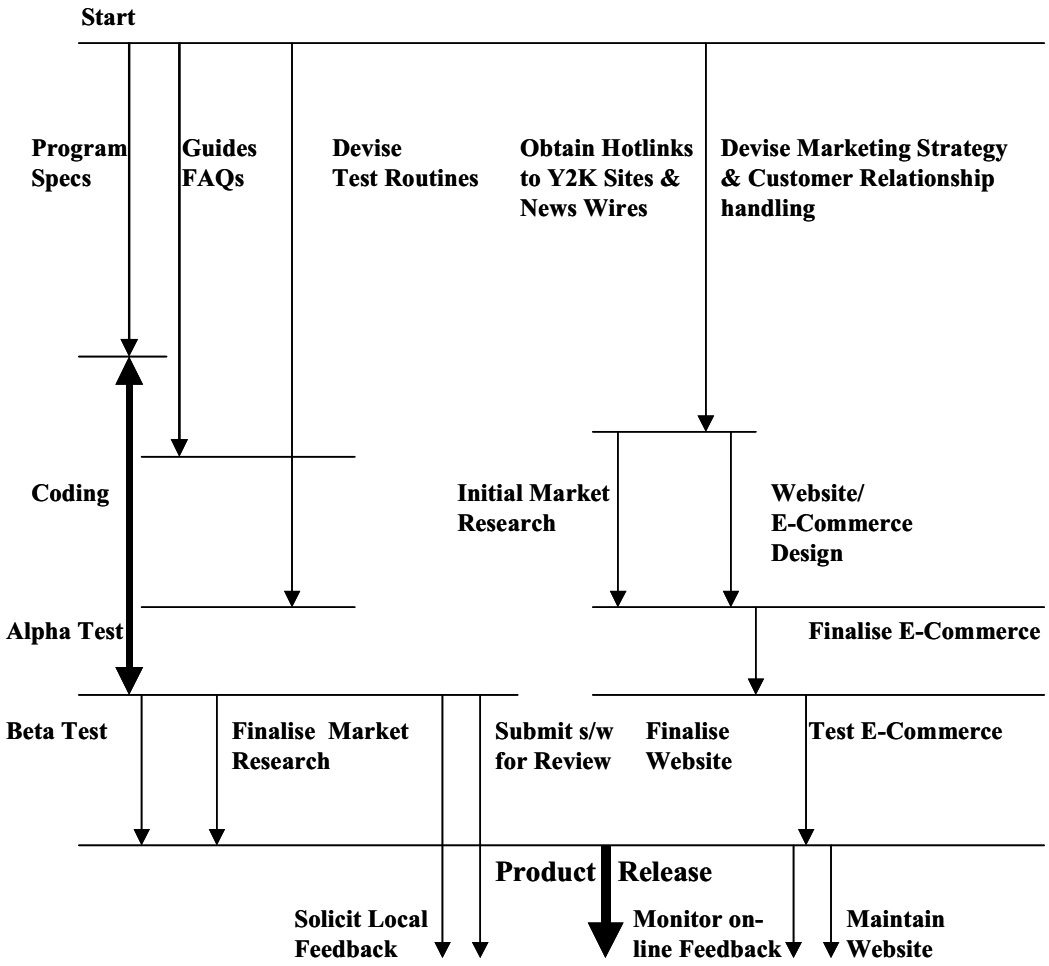

Figure 3. Project Gantt chart

In light of the ISDM literature survey it was felt that an evolutionary spiral model (Ould, 1990) with its emphasis on refinement of user requirements, via prototyping, would have the best chance of delivering an innovative software product. Consequently, once the broad user requirements had been agreed via focus groups comprising Y2K specialists, the design team members were each assigned separate tasks as indicated by the above Gantt chart. The self-help requirement of the final product meant that considerable emphasis was placed on the user interface using a rapid application development regime based on prototyping. Microsoft PowerPoint™ was used to work up the complete software layout in the form of a slideshow. Incorporation of navigation buttons enabled the program flow to be fine-tuned. The skills available in the team meant that, in the first instance, the 'users' were the developers. During the course of these early software design activities the conceptual design of a company website dedicated to the product was also initiated. As well as product features and specifications this also involved consideration of broad marketing strategies, product promotion, on-line distribution, and electronic payment systems.

Thus far the evolutionary spiral model was working reasonably well and, provided that only small changes to product specifications were required, these could be accommodated using the chosen project management tool. However, it soon became evident that there would probably be many late (and potentially large) changes required, both to the software product and to the website. This was in part because trends in software had moved on since project initiation. For example, the trend to tighter integration of external databases, tighter integration of web features, advances in online support, and security measures for online transactions (with commensurate heightening of customer expectation) made it desirable to incorporate web-enabled technologies into the final product. Such major shifts in specifications were, in the main, not foreseeable.

The inherent need for shortest time to market, to achieve early 'share of mind', soon came into conflict with the need to incorporate late design changes at short notice in response to new competitor products and emerging trends in E-commerce. Such problems came to a head when focus group feedback confirmed the need for major design changes to incorporate web support, web features and web distribution into the final product. To address such issues in a design sense it was necessary to 'back track' to earlier conceptual stages of the ISDM when such design features had been tentatively explored but not followed through.

It was also deemed necessary to release the highest quality software product possible to (hopefully) obviate the need for a potentially financially crippling global recall, or a product patch, with its attendant adverse publicity. Again, this market-driven decision heightened awareness of the conflicting requirements for highest quality with an early product release date.

Further downstream a new challenge began to emerge regarding the overall cohesiveness of the project activities. The wide variety of activities and the limited skills available in the team meant that some aspects of design were moving ahead very quickly with others hardly progressing. In addition, although the design team was being very creative and were performing well individually, the project began to lose focus. It was felt that each product component was independently 'spiralling' towards its own unique destination! Thus, a major management challenge that emerged was the need to more strongly coordinate progress on the various design components.

By this time the limitations of the basic evolutionary spiral model were becoming very apparent. Although the overall cyclical nature of the spiral ISDM for assessing risks encouraged the team members to constantly define the best way forward overall, and to refine the *overall* design concept, the need to revisit earlier *component design* 'solutions', -in response to late changes in product specifications, was not able to be so easily accommodated. Consequently, a more flexible system began to emerge.

Prototyping methods that had been used extensively in the early stages for interface design had also been found invaluable for much of the remaining cyclical development activities. For example, work on the marketing plan that had started near the beginning of the project to test the original concept was sometimes put aside for weeks at a time. This inactivity would be followed by periods of refinement and adjustment as time permitted. As the product release date approached, the 'soft-coded' marketing plans were dusted off and refined some more until the final plans became 'solidified'. This same cyclical style of development was also utilised for website design, product testing, and the building of help files and other support mechanisms. Even the programmer chose to follow a similar approach. Rather than coding a module or program component to completion, basic functionality was installed and approval then sought from the software designers before advancing further. This was found to all but eliminate misinterpretations of the *evolving* program specifications.

By the late stages of the project a new ISDM had emerged that largely overcame the limitations of the original model. This was largely necessary because, at any time, there would be several distinct developmental activities in action, each with its own cyclical design process involving prototyping. As a result of the excessive time taken to fine tune the design of the user interface, strict deadlines had to be enforced to keep the project on schedule. Careful planning was required so that concurrent activities with different start dates could be completed by a common due date. Resources allocated to each of the activities were fluidly assigned according to the importance of each at that particular point in time. This suited the designers who could be provided with a variety of work. Not only did this contribute to high levels of productivity, it also appeared to encourage creativity and enthusiasm.

The software product was launched soon after its planned release date and did not need rework or alteration. Clearly, the ISDM that evolved was critical to the success of the project. The constraints imposed on this young dot.com company led to a product development approach evolving that enabled concurrent, cyclical design activities to be changed at short notice in response to emerging requirements.

5. Information Systems Development Methodology – A New Approach

In light of the lessons learned in a dot.com environment this section proposes a modified spiral model for system development of innovative software products that are created under heavy time and resource constraints.

5.1 Systems Development

Figure 4 shows a software *Design Spiral* enclosed by a *Decision Space*. This multi-dimensional Decision Space is the domain of decision making for the design team. It expands or contracts as the project progresses, in response to changing internal and external pressures (constraints and opportunities). The design team has complete freedom (subject to time and resource constraints) to move anywhere within that space as they work through a

major design component, such as market assessment, conceptual software design, or associated product marketing.

A major departure from earlier ISDMs is that a separate spiral exists for each major design component and each spiral has its own associated decision space concerned with detailed design decisions (the complete set of such subordinate decision spaces comprises the overall Decision Space).

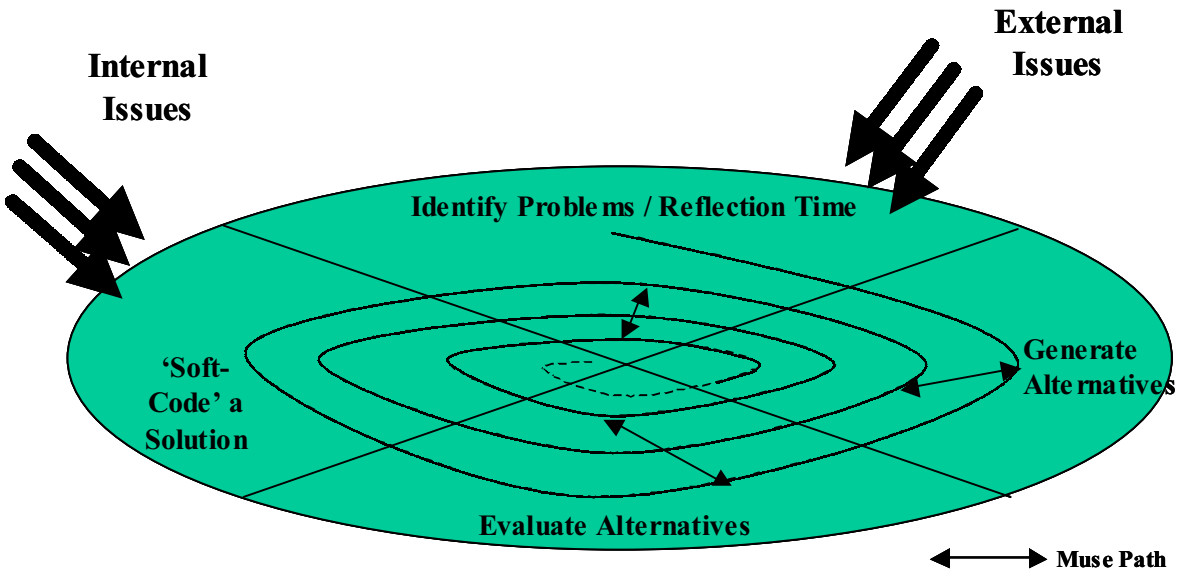

Figure 4. Helical Model - Decision Space

In practice, once the baseline requirements have been agreed, design team activities will tend to spiral inwards to the 'final solution', transiting through the 4 design stages (quadrants) of:

- **Problem Identification/Reflection** – identification of the immediate problem set to be addressed, or opportunity to be maximised, on subsequent cycles. This is also referred to as the 'Reflection' quadrant because an interim 'solution' might be temporarily set aside to allow time for reflection by the design team or for developments to become aligned in an adjacent (related) design spiral. The latter activity need not occur at a programmed calibration point (refer to later Figure 5)
- **Generation of Alternatives** – alternative solutions generated as the result of a formal requirements determination
- **Evaluation of Alternatives** – alternatives evaluated according to relevant decision space criteria, including risk assessment
- **Soft-Code of a Solution** – solutions are soft-coded (remain 'live') until such time as the project team is 'out of time' when the last solution becomes the hard-coded solution that is used in the final product, its distribution, and promotion.

Each preceding stage informs the next and rapid prototyping is used to generate and evaluate (component) design alternatives. Constant appraisal and reappraisal of the emerging design and its 'fit' with the other design components occurs in all quadrants. The actual number of design iterations depends on the nature of the problem being addressed, since the desire to produce an innovative, high quality product is tempered in resource-limited companies by the need for a rapid time to market. Project team activities iterate around the same 4 basic steps, refining the solution on each cycle and consequently achieving less progress per unit of time (Pareto's 80:20 rule).

Figure 4 also indicates the presence of so-called 'muse paths'. By keeping clear and careful records of their development activities, the design team is able to move back and forth between developments that occurred earlier in the same quadrant so as to quickly begin a fresh line of investigation, should that need arise. This may be necessary if unexpected developments (favourable or unfavourable) occur in adjacent decision spaces or in the external environment. For this reason, clear and accurate design activity records must be maintained.

5.2 Project Management

Given the severe time and other resource constraints often present in dot.com company projects, sound project management expertise is essential. Irrespective of whether the aim is to produce a groundbreaking product, the project manager must be alert to internal and external issues that could have an immediate and lasting impact on the success of the project. Internal issues impacting the Decision Space may include resource limitations and coordination across project activities, such as deadlines concerning product promotion and distribution. Relevant external issues include emerging technical developments that could be incorporated into the product, E-commerce developments to improve product distribution and marketing, as well as macro changes in the political/legal, social, and economic environments.

From the earliest stages, the project manager must identify any *critical* components of the design which, in terms of web supported and distributed software products, usually means factors or features that could 'kill' the product on its launch (e.g. poor or dated design), or could require expensive after-sales support or remediation (e.g. software error fixing). Such components deserve early attention by the designers. Non-critical components may justify adopting a satisficed solution (Simon, 1957), - a solution that is not necessarily the best or the 'optimal', but which nevertheless 'does the job'. Early identification of such components is vital since this allows sufficient time and resource to be assigned to the critical, and often more creative, product design elements.

Of course, design team efforts expended on the design spiral activities occur across time. Figure 5 shows the trajectory of a project that contains 4 major activities. The diagram indicates that 3 major activities will begin concurrently at the top of the project with the fourth beginning some time later. Stretching of the 4 design spirals across time results in the 4 *helical* shapes shown. Each represents progress in one component of the full project, e.g. website design activities, software design activities, and so on. The time axis runs from left to right whereas effort in the design spiral follows the spiral path. More design spiral iterations per unit time, indicative of greater design effort, provides a finer helical 'thread' whereas the pitch of the thread (axial distance between the threads) represents progress i.e. how quickly the developers converge towards the optimum solution. However, the initial aim of each design team is to achieve a satisficed solution since, by definition, without this the final product cannot be evolved.

Calibration points are defined as predetermined (programmed) points in time at which progress is monitored and any resource adjustments made with the aid of traditional project management software. Given the highly creative nature of the work it is intentional that the project is not over-managed while being cognisant of the need for:

- An early satisficed set of solutions

- Overall project terminal points and milestones to be met
- Design decisions to be recorded to enable backtracking to earlier decisions and activities via muse paths
- Breakthroughs occurring in one major design component to be fed into adjacent decision spaces.

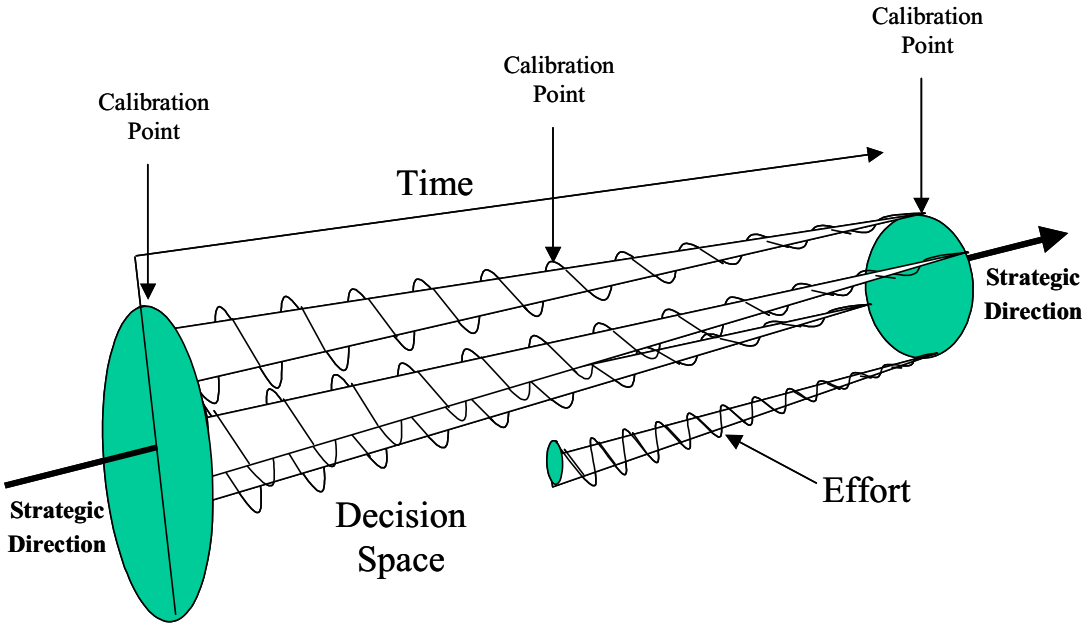

Figure 5. Helical Model - Action Trajectories

The overall direction of the project must also be aligned with the organisation's strategic direction if the final product is to be effective in business terms (Deakins and Makgill, 1998).

6. Discussion

It was stated earlier that contemporary models of information systems development, while acknowledging the need for creativity and innovation in modern IS planning, do little to cater for this need explicitly. This paper describes field research that attempted to utilise an evolutionary spiral design methodology for a heavily time- and resource-constrained project requiring innovative solutions. The method of research was unique because the field research led to a modified approach to systems design rather than simply providing a test bed of an existing method. The resulting so-called 'Helical Model' offers an improved ISDM that is likely to be of value in software development environments that require:

- Rapid, high quality product development
- Innovative and creative solutions, via experimentation
- Continual improvement to product specifications, via high levels of customer feedback and responsiveness to the external environment
- Just in Time (JIT) delivery practices.

6.1 Advantages and Limitations of the Helical Model

The model presented is cognisant of the nature of contemporary software product design, being often random, ad hoc and visionary (Tong, 1994). The 'Helical Model' has the following advantages when used for the development of innovative software products:

- It is rapid, the main aim being to achieve a satisficed solution that meets baseline requirements. Layers of refinement and quality are then iteratively and incrementally added to a set of soft-coded *product components*
- Creativity via experimentation is encouraged since all changes are reversible and no requirements are frozen. Team members are empowered to make decisions and a collaborative and co-operative approach is encouraged
- High levels of customer feedback and responsiveness to the external environment is incorporated via prototyping; testing is integrated throughout the development cycle
- Just in Time (JIT) delivery of last minute design changes is possible.

The Helical Model is also broadly in line with principles proposed by Howard (1997) who presented a Rapid Application Development (RAD) approach. Thus, it also has the following advantages over existing ISDMs for managers of innovative projects:

- It provides a lens through which to view especially creative and innovative projects
- It has intuitive appeal that helps the project manager and the design team to appreciate the nature of the processes occurring within the team
- The focus is on the end product and on opportunities for refinement rather than on time to 'complete' the design activity. This lessens the need for close project control
- Non-critical 'satisficing' elements are explicitly identified, making project planning to a clearly defined set of objectives much easier
- The distinction between satisficed and optimal solutions encourages the reconciliation of business and design tensions.

The Helical Model has the following major advantages over traditional ISDM methods for the members of the design team:

- It provides freedom to create innovative solutions subject only for the need to produce an initial set of satisficed non-critical design components. Optimal solutions for the identified critical components must also be produced, constrained only by the available time and resources remaining for the project
- (relative) Freedom from what is often perceived to be 'petty' administration and restrictions on creative freedom.

Table 4 combines the results of earlier tables to compare the value of the proposed Helical Model with some traditional ISDMs for product development in dot.com environments.

Table 4. Comparison of ISDMs for dot.com software product development

	General Models	**Spiral Models**	**Helical Model**
Rapid Development	Low	Low	Medium
Innovative and creative solutions	Low	Medium	High
Continual improvement to product specifications	Low/Medium	Medium/High	High

Specific limitations of the model include:

- Potentially narrow scope: the Helical Model is untested within any environment but a dot.com company. For example, it may be a poor fit with other organisation or systems development cultures
- By itself it lacks internal control elements but it can be readily combined with traditional project management tools
- It has the potential to be time consuming if not adequately project managed
- It may be difficult for the project manager to plan and manage a large number of concurrent design spirals
- Developers may underestimate the importance of major design decisions; knowing that they can "back track" they may be tempted to postpone 'hard' design issues.

At this stage the proposed Helical ISDM shows great promise. Further research is needed to test and refine the model in other settings, perhaps using an Action Research approach.

References

Alonso, F., Juristo, N. Mate, L., and Pazos, J. "Software Engineering and Knowledge Engineering: Towards a Common Life Cycle," *Journal of Systems and Software* (33:1), 1996, pp. 65-79.

Blackburn, J.D., Scudder, G.D., Van Wassenhove, L.N., and Hill, C. "Time-Based Software Development," *Integrated Manufacturing Systems* (7:2), 1996, pp. 60-66.

Boehm, B.W. "A Spiral Model of Software Development and Enhancement," *IEEE Computer* (21:5), May 1988, pp. 61-72.

Carey, J.M. "Prototyping Alternative Systems Development Methodology, *Information and Software Technology* (32), 1990, pp. 119-26.

Deakins, E., and Makgill, H.H. "The Importance of Alignment in Business Process Change Projects," *Proceedings of the Ninth Australasian Conference on Information Systems*, Sydney Australia, September 1998, pp. 148-160.

Feurer, R., Chaharbaghi, K., and Wargin, J. "Developing Creative Teams for Creative Excellence," *International Journal of Operations & Production Management* (16:1), 1996, pp. 5-18.

Fitzgerald, B. "Formalized Systems Development Methodologies: A Critical Perspective," *Information Systems Journal* (6), 1996, pp. 3-23.

Howard, A., "A New RAD-Based Approach to Commercial Information Systems Development: The Dynamic System Development Method," *Industrial Management & Data Systems* (97: 5), 1997, pp. 175-177.

Nandhakumar, J., and Avison, D.E. "The Fiction of Methodological Development: A Field Study of Information Systems Development," *Information Technology & People* (12:2), 1999, pp. 176-191.

Orlikowski, W.J. "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development," *MIS Quarterly* (17:3), 1993, pp. 309-340.

Ould, M.A. *Strategies for Software Engineering: the Management of Risk and Quality*, John Wiley & Sons, Chichester, UK, 1990.

Prahalad, C.K., and Ramaswamy, V. "Co-opting Customer Competence," *Harvard Business Review*," January-February 2000.

Sage, A.P. *Systems Management for Information Technology and Software Engineering*, John Wiley and Sons, New York, ISBN 0-471-01583-0, 1995.

Simon, H.A. "Rational Choice and the Structure of the Environment," in Simon, H.A. (ed), *Models of Man*, John Wiley: New York, 1957.

Tong, G. "Software Development Process Improvement: The Forgotten Son?," *World Class Design to Manufacture* (1:3), 1994, pp. 21-25.

Veryard, R., "What are Methodologies Good for?" *Data Processing* (27:6), 1985, pp. 9-12.

Walters, S.A., Broady, J.E. and Hartley, R.J. "A Review of Information Systems Development Methodologies," *Journal of Library Management* (15:6), 1994, pp. 5-19.

Wastell, D.G. "The Fetish of Technique: Methodology as a Social Defence," *Information Systems Journal* (6), 1996, pp. 25-49.