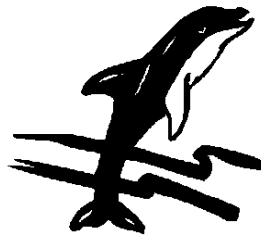


BOND UNIVERSITY
SCHOOL OF INFORMATION TECHNOLOGY

COMP420
Advanced Software Engineering



David Reilly
942-004-961

An Investigation of the Spiral Model, and its Variants

David Reilly

dodo@fan.net.au

Abstract

The practice of engineering software brings process and order to the task of developing custom systems tailored to meet the specific needs of an individual, or organisation. When a rigidly defined set of requirements, drawn from the interaction between engineer and customer, can be determined, software engineering provides techniques that allow projects to be verified and validated at each stage or milestone, providing developers with a reasonable degree of certainty that the product is being constructed in adherence to the functional and non-functional requirements of the customer.

Yet traditional techniques fall down when the customer has an inability to articulate his or her ideas for a system, and the requirements document can fall short of what the customer intends the system to be. Worse still, the customer may think that what he wants the system to be *is* defined in the requirements, even though there may be gaps between the customer's perception, and the reality of what has been asked for. Requirements engineering is a field of study in itself - leading to new techniques that can be applied, and new life-cycle models that can take the place of traditional models such as waterfall. One such model is the SPIRAL model, put forward and later revised by Barry Boehm.

Overview

Traditional approaches to the software process advocated the use of popular life-cycle models, such as the waterfall model. The waterfall model suited particular problem domains, such as rigidly defined systems in which the customer and developer could agree upon a frozen set of requirements. Yet for systems in which the customer's needs and desires were ambiguous and changing, or where there existed a high element of risk, the waterfall model alone did not offer an adequate process. Incremental models, and rapid prototyping techniques were sometimes applied, but there remained a void for a process model that would perform consistently and with a high degree of confidence.

The spiral model, while not a universal panacea for the ills of the software industry (Pressman), is an alternative life-cycle model that provides developers with the flexibility to prototype with incomplete requirements, the opportunity for incremental delivery of systems as they evolve from one build to another, and the confidence that comes with the application of risk management techniques to the software process. Yet despite these advances, the spiral model has fallen short of its intended goal, and several variations have been proposed. The purpose of this report is to analyse the spiral model, and its variations.

Spiral Model

The spiral model, in its original form, consisted of four phases (Pressman)

- Planning (objectives, constraints, alternatives)

- Risk Analysis
- Engineering
- Evaluation

Each phase is represented as a quadrant of the spiral model (shown below), and through successive iterations of these phases, a project moves along the path of the spiral. Initially, the project requirements are described, and then risk analysis is used to determine the degree of uncertainty and volatility of the requirements. Prototyping is then used in the engineering phase to build a simple version of the application, which is evaluated by the customer. Evaluation is an extremely important phase of the spiral model, as it is through the evaluation of each build that a more concrete requirement specification can be realised.

At the conclusion of each spiral, a decision must be made as to whether it is practical to continue the project. If there is still great risk, further prototypes and evaluation may take place, or the project may be cancelled altogether. If a decision is made to go ahead, the spiral progresses to the next iteration, where the four activities are repeated to produce a more detailed system. Functionality is built into the application, and more and more builds are produced. By the time the spiral has reached the third level, an almost complete system should have been produced.

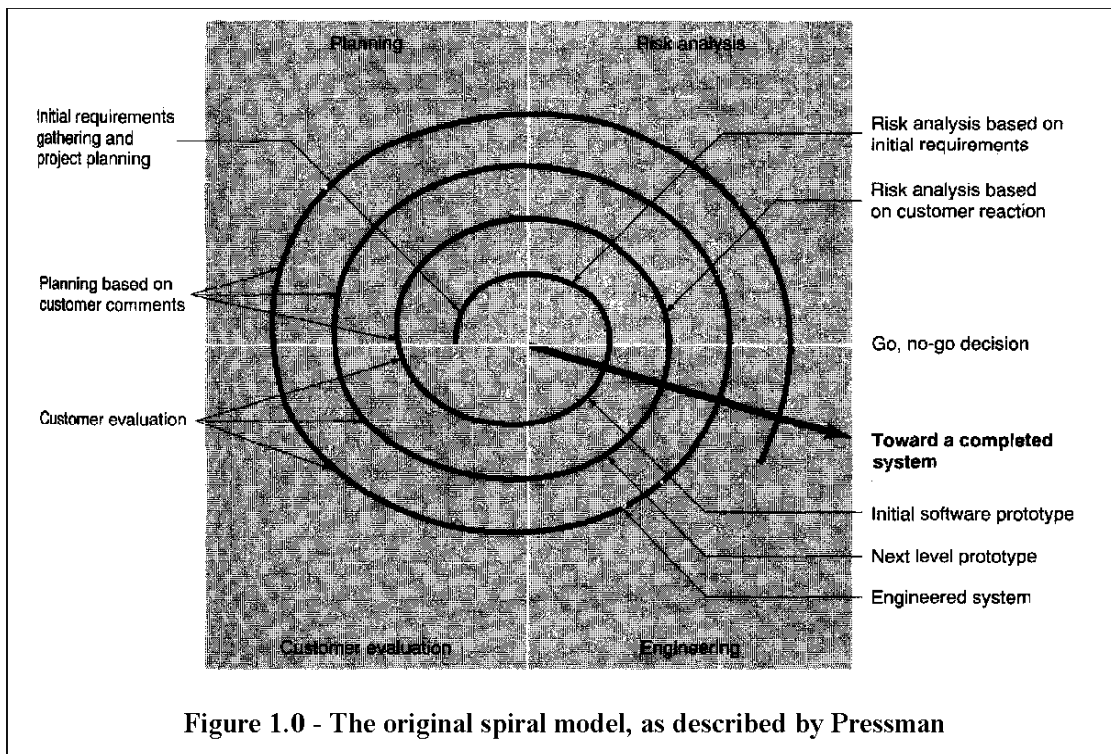


Figure 1.0 - The original spiral model, as described by Pressman

SWOT Analysis

Strengths

- Introduces formal risk management to the software engineering process.

Boehm's spiral model was the first to make risk management a part of the software management process (Charette). By identifying and managing risks earlier, they can be avoided if possible, and the project terminated without consuming more resources, if the risks become too great.

- Prototyping controls costs (Sorensen, Acosta), and conveys to the user the look and feel of a system far more thoroughly than a requirements specification (Boehm, 1996).

By using prototyping, the user can see a tangible representation of what the final product will be. It reduces costs, because if the customer is unsatisfied with an aspect of the prototype, the cost to correct it at that stage is far smaller than at a later date.

- Evolutionary development allows a product to be released for evaluation early, and seeks to provide feedback and evaluation for the development team.

This means that measures to rectify a project that has gone off track can be put in place early, rather than discovering the system doesn't meet customer needs when the final version ships. If the customer needs to justify the expense of the project, they have a partially functional system well in advance of final shipping, and the development team receives early feedback (May).

Weaknesses

- Lack of risk management experience (Charette).

Software project managers are not normally familiar with formal risk management. While the spiral model has raised the need for risk management techniques, organisations will have difficulty adjusting.

- Lack of milestones (Microsoft, Boehm).

The spiral model doesn't provide any form of milestones. For the developer, the spiral model can lead to a lack of accountability, a distinct time-frame, and specified deliverables (Microsoft). There is a tendency for projects to go off track, and bringing together the efforts of all individuals concerned can be difficult (Boehm). For the spiral model to be generally applicable to any organisation, there is a need for tangible milestones.

- Management is dubious of the spiral process, its evolutionary nature (May), and the concept of starting a project without rigidly defined objectives (Sorensen).

Management may be reluctant to commit to a new and (within the organisation) untested software process, particularly when it involves starting with incomplete objectives, and means releasing multiple builds before the release date. Project managers fear that producing a release every few weeks is impractical, considering the time taken to produce a single release can often be several years (May).

Opportunities

- The spiral model offers the opportunity to release builds of the product early, for alpha and beta testing and, if a competitor appears to be gearing towards release of an alternative product, shipping releases.

If time to market concerns are a problem, products can be developed using the spiral model and shipped before offerings by other competitors. This means that companies can leave a product in continuous development, but ship it early in the event a competitor announces an imminent release.

- Marketing and sales of products can be improved by application of the spiral model (May)

Marketing departments can access early builds of products, aiding in documentation and promotional activities (May). By releasing early versions of products to users/customers, the product's features can be shown and validated by users. When the product is finally ready to ship, the consumer audience is aware of the product, and ready to purchase, rather than the traditional "nine to fifteen months before the market believes a product will do what it says it will". Sales can start even before the product is released (May).

- Risk analysis offers the opportunity to identify project risks, and to terminate if risks are too high.

Some projects can go millions of dollars over budget, and be many years late in delivery before effective action is taken for correction (Charette). Indeed, many projects are cancelled, but only after they have consumed massive resources. The spiral model's emphasis on risk allows developers the opportunity to abandon projects before they reach more expensive levels of failure.

Threats

- Change in management composition (takeover, economic rationalization), or shift in focus, of key stakeholders

Many of the case studies recognise a genuine apprehension of management where the spiral model, and evolutionary development is concerned (May). The cost of integration (builds), the inefficiency of producing a product every few weeks rather than a single product at the end of the project life-cycle, are all concerns raised by management. A development team may have convinced upper management of the potential for success offered by the spiral model, only to have a project cancelled, or ordered to use an alternative model, when a new set of upper management in one of the key stakeholders is introduced.

- Customer, and management, may not understand the complexities of the software development process, and may be impatient when they see a fully featured user interface, but have to wait months for extra functionality such as security, reporting, etc.

There is a tendency to push difficult problems to the future (Sorensen), and while customers are often impressed at the delivery of early functionality, they can quickly become dissatisfied, because programming teams can't produce the

product at the same rate. User-interfaces can be constructed quickly and easily, but advanced functionality can take a considerably longer time period.

Spiral Model variations

The spiral model provides a good framework with which to develop other, more vigorous and robust life-cycle models. Unsatisfied with his original model, Boehm developed the WinWin Spiral model, the Software Productivity Consortium developed the Evolutionary Spiral Process, and Microsoft used the spiral model to come up with the model used in the Microsoft Solution Framework.

For the SPC, the lack of risk management experience in the software necessitated introducing a guidebook to describe solid management practices for risk analysis and reduction (SPC). For both Boehm, and Microsoft, the lack of tangible milestones in the spiral model was unacceptable, and required an enhanced version of the spiral model. Case studies for both Microsoft and Boehm's methods are analysed, and a description of Boehm's WinWin spiral model is also given.

WinWin Spiral Model

The first phase of the spiral model concerns planning the systems objectives, constraints, and alternatives for offering a solution. In the WinWin spiral model, this task is achieved by identifying the system's key stakeholders, and their "win" conditions. Often, the union of all the stakeholders' win conditions is too great, which may necessitate a reconciliation between all those concerned (Boehm). It is an important addition to the spiral model, as without determining these win conditions, a product may be developed that lets down stakeholders in some fashion, leading to dissatisfaction with the final system.

The other major addition to the spiral model is the introduction of tangible milestones to the software process. In his article, "Anchoring the Software Process", Boehm describes how organisations were left with no common frame of reference with which to gauge spiral projects, in terms of cost and schedule estimations. By providing milestones, it is hoped that organisations can determine how far a project has progressed.

Boehm defines three critical milestones for the spiral model

- life cycle objectives (LCO)
- life-cycle architecture (LCA)
- initial operational capability (ICO)

Life-cycle objectives help describe the objectives of the system, and lead into the life-cycle architecture which serves to elaborate on the original objectives. The LCO identifies the objectives of the system, and the architecture shows how it is achieved.

Milestone Element	Life-cycle Objectives (LCO)	Life-cycle Architecture (LCA)
Definition of operation concept	Top level objectives and system scope	Elaboration of objectives and scope
	<ol style="list-style-type: none"> 1. System boundary 2. Environment parameters and assumptions 3. Evolution parameters 	Elaboration of operational concept
Definition of system requirements	Top level functions, interfaces, and quality standards	Elaboration of functions, interfaces and quality
	<ol style="list-style-type: none"> 1. Growth vectors 2. System Priorities 	Identification of items that have yet to be determined
Definition of system and software architecture	Stakeholders' agreement on system essentials	Stakeholders agreement on any priority concerns
	<p>Top level definition of at least one choice of feasible architecture</p> <ol style="list-style-type: none"> 1. Physical and logical elements and relationships 2. Selection of COTS, NDI and reuse options 	<p>Choice of architecture, and elaboration on unspecified details</p> <ol style="list-style-type: none"> 1. Physical and logical components, configurations, and constraints 2. COTS, NDI, and reuse choices
Definition of life-cycle plan	Identification of which architecture items are unfeasible	
	<p>Identification of key stakeholders, including but not limited to :</p> <ul style="list-style-type: none"> • Users • Customers • Developers • Maintainers • General public 	<p>For the initial operational capability milestone, elaboration of</p> <ol style="list-style-type: none"> 1. <i>Why</i> is the system is being developed? 2. <i>What</i> will be done by <i>when</i>? 3. <i>Who</i> is responsible for a function? <i>Where</i> are they located within the organization?

		4. <i>How</i> will the job be done, both in terms of technical and management details?
		5. <i>How much</i> resources will it cost?
Feasibility rationale	Assurance of consistency of each element in the LCO through <ul style="list-style-type: none"> • Analysis, measurement, prototyping and simulation • Business case analysis of requirements, feasible architectures 	Assurance of consistency of each element in the LCA Risk management plan identifies and gives solutions for reducing risks.

Figure 2.0 - A summary of two milestones identified by Boehm (LCO/LCA)

The third milestone occurs later in the software development life-cycle. The initial operational capacity milestone (IOC) helps avoid delivering a system that has “ill-matched software, poor site preparation, or poor user preparation” (Boehm).

Milestone element	Description of element
Software preparation	Operational and support software with appropriate documentation, licenses for COTS/NDI and readiness testing
Site preparation	Hardware equipment, supplies, and COTS/NDI vendor-support arrangements
User/ Maintainer preparation	Training for usage or maintenance

Figure 3.0 - A summary of the third spiral model milestone (IOC)

A critical weakness in the spiral model was the lack of these tangible milestones. Boehm’s revised spiral model, the “Win-Win Spiral Model”, offers substantial advantages over the previous versions. Wherever the spiral model is a candidate life-cycle, the Win-Win model should be given preference, as its emphasis on stakeholder objectives and milestones makes it superior to its predecessor.

Case Studies

To illustrate the usefulness of the spiral model, summaries of four case studies involving the application of spiral techniques on real projects are presented. The case studies involve variations on the spiral model, as well as a look at evolutionary paradigms (which provide a limited subset of spiral capability).

United Airlines

United Airlines required an electronic system that would provide an alternative means of access to flight reservations. The system was developed in collaboration with Microsoft Consulting Services, as United Airlines has little experience in developing consumer software products.

The Microsoft Solution Framework (MSF) was used to develop the software product, “United Connection”. A key element of the MSF is its process model, which uses a derivative of Boehm’s spiral model. Like Boehm, Microsoft was dissatisfied with the lack of milestones in the spiral model, and by their addition, was able to realise the full benefits of the spiral model.

Microsoft, in their case study of the United Airlines project, highlighted the following benefits from using the spiral model :

- Lower software development risks
- Rapid application development, with incremental delivery of software
- Prototyping / incremental delivery allowed the target audience (customers) to preview the user-interface, and allowed the project team to better understand their needs.
- Risk analysis, and feedback from users early, helped prevent ‘scope creep’, which would cause the project to expand its schedule and budget.

By using a spiral model with milestones, Microsoft and United Airlines were able to complete the project four months before its target completion date. Incremental delivery of the system (internal alpha, beta and final product), allowed the team to get feedback, and tailor the system to the specific needs of the user. The product was an overwhelming success, and at the date the case study was written, had sold 74,000 copies which retailed at \$25.

Hewlett-Packard

Hewlett-Packard has used the evolutionary paradigm on several of its software development projects, with varying degrees of success and failure. Evolutionary development releases products in small increments, like the incremental model. It differs, however, in that it allows the project to proceed without defining all requirements (MII, -498). Evolutionary paradigms provide a small subset of the functionality of the spiral model, eliminating all risk analysis, and do not retain the focus on stakeholder objectives and milestones of the Win-Win spiral model. Without milestones (Boehm), or even risk analysis which is critical to large-scale software management (Charette), the success of evolutionary paradigms alone is dubious. Nonetheless, the HP case study is presented to illustrate the benefits of evolutionary development, which forms part of the spiral model.

May and Zimmer, in their paper “Evolutionary Product Development at Hewlett-Packard” examine the success of three divisions within HP, operating under the fictitious names of ABC, LMN, and XYZ.

ABC Division

ABC used evolutionary development on two large projects. The first used evolutionary development for two-thirds of the project life-cycle (finally abandoning evolutionary development due to time constraints). The remainder of the project introduced significant quality defects, in the absence of evolutionary development, though the project was considered a success.

In its second, and much larger project, ABC division used internal users initially to provide feedback, and later, external customers. ABC division found that the feedback from external customers was so good that sale orders were placed for the product ahead of its completion and shipping date. Through early exposure to the product, the time in which customers became familiar and confident enough to purchase was diminished.

LMN Division

The first project of LMN to apply evolutionary development was considered a failure. The first build was an unusable “paper prototype” that provided no real feedback. The second release took six weeks of development, and was considered to be wasteful in terms of integration and logistical effort (May). Evolutionary development was abandoned.

The second project to use evolutionary development was smaller, and delivered a quality product. Feedback from focus groups, who evaluated prototypes, helped the team to understand user needs, and helped clarify the “vision” of the project within the division. Evolutionary development proved a success for this project.

XYZ Division

Despite initial reservations about using evolutionary development, XYZ was pleased with its effect on its project. By breaking down a system into two week sections, engineers were able to adapt to the new evolutionary style of development. There were problems in estimation, but they became more adept at estimating how much work could be completed before each two week build, and this in itself was of benefit, as project estimation benefited from the evolutionary paradigm.

Multimedia Applications with the WinWin spiral model

Boehm presents a case study, in which fifteen teams developed components for a multimedia application, using the WinWin spiral model, and the three milestones he offers to anchor the software project. Boehm investigated whether the WinWin spiral model would be a good approach to take for the development of multimedia applications, and whether it would be an efficient life-cycle model.

His findings suggest that with only two cycles of the spiral model, multimedia components could be developed within a reasonable time period (11 weeks), and that the approach resulted in meeting all of the win conditions for the stakeholders concerned with the project. Boehm believes that the WinWin spiral approach is flexible enough to adapt to real-world conditions (problems with COTS/NDI, personnel changes, initial lack of domain expertise), and that it improved the relationship between developer and client. Boehm does cite some complications, however, such as document overkill, and a lack of time to produce a third

spiral cycle. He also believes that, by using the WinWin spiral model, it would be possible to automate the outputs of the model into a requirements specification. Currently, this is done manually, but the potential for automatically generating requirements specifications is promising indeed.

Boehm concludes his case study by suggesting that while the WinWin spiral model can be used successfully for developing multimedia applications, it can also be applied to projects with similar characteristics (rapidly changing technology, lack of developer domain knowledge, projects with risk and uncertainty), and that the three milestones he advocates with the WinWin spiral model were of great benefit in maintaining project focus. Through the use of the WinWin model, stakeholder confidence increased, and the relationship between customer and developer changed from one of adversarial conflict to a supportive co-operation.

STARS Project

The Department of Defense's STARS project applied the spiral model, under the guidance of Boehm, to its software engineering tasks in 1989. Charged with developing a set of software-engineering environments, Boehm used the risk analysis techniques and stakeholder objectives of the WinWin spiral model to complete the project.

The project's LCO milestone was that a strategy success plan would be developed, and endorsed by each of the major stakeholders. The LCA milestone was that each of the development teams would provide risk-driven life-cycle architectures for developing the STARS software-engineering environments. The IOC milestone was delivery of a software-engineering environment for use with a major defense project.

The WinWin spiral model delivered significant advantages to the STARS project, in terms of efficiency, cost, and quality. For the Air Force Space Command component of the project, the cost per thousand lines of delivered code dropped from \$140 to \$57, and the rate of defects dropped from 3/KDSI to 0.35. In the Navy component of the project, a "quality factor" of 10 was reached, compared to a previous of 3.

Factor for selection

The spiral model is more suited to some project situations than to others. Sorensen, in "A comparison of software development methodologies", outlines situations in which the spiral model is more appropriate than traditional techniques, and similar guidelines are offered by MIL-STD 498.

Factors which may influence the spiral model as a candidate over waterfall :

- Delivery of early functionality, for feedback and evaluation
- Project does not have a clear set of requirements, and needs clarification with prototyping to help stakeholders visualise their needs.
- Risk analysis is required, as project is large, deals with new technologies or a lack of domain experience, or there is the potential for the project to exceed its budget/time constraints.

- New features may have to be added at a later date, and incremental delivery of features is viable.

Factors which may influence waterfall as a candidate over the spiral model :

- Customer prefers to phase out old system at once, rather than running incremental builds concurrently
- Customer can articulate all requirements, and agrees to be bound by the requirements specification offered by the developer.
- Small project, with no need for risk analysis

Conclusion

Case studies that use the spiral model, or its variants, show significant benefits for key system stakeholders, in terms of quality, efficiency, and cost. While not applicable to every situation, the spiral model does stand out as a candidate for large scale projects, or projects in which customer requirements are vague and incomplete. Its emphasis on risk management, a factor essential to the success of any large project, and on evolutionary development and prototyping, means that products can be released early to customers for evaluation, with enough time to tailor a system to suit changing needs. Yet developers should also be mindful that the spiral model, in its simplest form, doesn't offer any milestones, and risk management techniques require some experience before they can be put into place. On the proviso that managers have the prerequisite risk management techniques, and the superior WinWin spiral model is used to provide milestones, the spiral model is a good candidate life-cycle model, and offers significant advantages over waterfall.

Bibliography

Acosta, R. et al. Applying Rapid Prototyping Techniques in the Requirements Engineering Environment CrossTalk, October 1994.

Available : <http://www.stsc.hill.af.mil/Crosstalk/1994/oct/> [1997, October]

Boehm, B. *Anchoring the Software Process*, IEEE Software (July 1996).

Boehm, B. *Multimedia Development with the WinWin Spiral Model* [online]

Available : <http://sunset.usc.edu/TechRpts/Papers/usccse97-504/usccse97-504.html> [1997, October]

Boehm, B. *Software Risk Management*, IEE Computer Society Press, 1989

Charette, R. *Large-Scale Project Management Is Risk Management*, IEEE Software (1996).

Department of Defense, MIL-STD 498 [online adobe pdf]

Microsoft Corporation. *United Airlines Case Study* [online]

Available : <http://www.eu.microsoft.com/CIO/articles/UNITEDAIRLINES.htm> [1997, October]

Pressman, Roger S. *Software Engineering : a practitioner's approach*, McGraw-Hill, 1992.

Software Productivity Consortium. *Evolutionary Spiral Process Guidebook (product description)* [online]

Available : http://www.software.org/pub/Products/esp_gb.html [1997, October]

Sorensen, R. A. *Comparison of Software Development Methodologies*, CrossTalk, January 1995.

Available : <http://www.stsc.hill.af.mil/Crosstalk/1995/jan/> [1997, October]