Spiraling Waterfalls: A Hybrid Process Model for the New Reality of Software Development

Author: Peter Kulik, July 1997

Abstract: Java™, OOD, Internet and Intranet – today's business and technical realities are driving fundamental change in software development processes. This white paper discusses the Hybrid process model, which builds on the strengths of the Waterfall and Spiral models, but is optimized for the new reality of software development. The success factors underlying this model are comprehensive architectural design, project management, and risk management.

1. Motivation

The two "classic" process models of software development are the Waterfall and the Spiral [1][2]. These models were developed at a time when:

- Procedural languages were prevalent – even Assembler was in fairly common usage!
- Large, centralized systems were the norm – e.g. large COBOL applications for MVS mainframes.
- Code/debug/fix cycles were relatively long – time to market was often measured in years.
- Processor cycles were at a premium – and frequently rationed.
- User interface technology was in its infancy – think about Unix…

Since their definition, these process models have weathered many technological changes. Although several alternatives have been proposed and gained some level of industry support [3][4], the Waterfall and Spiral models continue to be the foundation for most MIS and Software Development shops around the world.

As early as the late 1980s and early 1990s, there was growing recognition that PC technology was making software development more iterative [5]. Today, though, step-function improvements in technology cry out for evolution of the tried-and-true Waterfall and Spiral process models. Some of these changes include:

- Growing prevalence of object-oriented methods
- Internet/Intranet enabling technologies; Browsers, Java™, HTML, etc.
- Multi-tier client-server distributed architectures are the choice for many applications
- Inexpensive hardware platforms – MIPS, memory, and disk capacity are relatively cheap
- Short debug /fix cycles –measured in seconds
- Intense time to market pressure – measured in months
- Greatly improved user interfaces and increased emphasis on usability

According to Ian Campbell of IDC, new technology "has replaced the old process of defining and deciding with deploying and doing"[6]. One MIS manager with whom we spoke described development cycles being pushed down from 18 months to 1 to 2 months!

Today's business and technical realities are clearly driving a fundamental shift in software development paradigms. This white paper describes a process model that builds on the strengths of the Waterfall and Spiral process models, and is optimized for the new reality of the technological environment.

The Waterfall Process Model	
Strengths	**Weaknesses**
• Adds structure to a potentially unstructured process	• Limits end-customer feedback to the beginning (gather requirements) and end (deploy product)
• Maps well to hardware development processes	• More efficient for larger projects – larger risk and potential delays
• Deterministic for time-to-market models	• Inflexible approach does not react well to "unexpected" design or requirements changes
• Lends itself to a phase-gate approach for program management	• Can cause time-to-market competitive disadvantage

Figure 1

2. The Waterfall

Strengths and weaknesses of the Waterfall Process Model are shown in Figure 1.

A software industry executive once stated that the best a product can be is 90% "right" in one pass [7]; industry experience shows this level is seldom reached. For example, if a product is only 50% right in its initial development, it will take four passes to reach 90% (actually 93.75% at the end of the fourth pass). How much chance does this product have of market acceptance in the first two passes – probably not much! There is a better way.

3. The Spiral

Many product development organizations have been reluctant to use spiral development processes. Such a project is less deterministic – from management's point of view, how many "spirals" are enough? It is also difficult to determine the state of the product – in terms of readiness for customer deployment – at any point in time. Traditional process measurement and metrics – such as defect density or cumulative failure profile – are not as useful for project management. Customer commitments are harder to make because it is more difficult to predict the level of quality the product will have achieved at a particular point in the future.

However, several very successful IT managers with whom we have worked have had great success with "spiral development". The common element of their success is to keep "projects" small – for example, 2-3 developers, 2-3 months in duration. This practice *demands* use of a spiral development process. However, most IT projects are different from product development projects.

Can a process model be developed to help product development and IT organizations realize the best of both the Spiral and Waterfall process models? Yes.

The Spiral Model	
Strengths	**Weaknesses**
• Flexible process – more adaptable to design and requirements changes	• Less deterministic – more complex to manage
• Can result in faster time to market.	• Hard for management to determine a project's status at any point in time
• Can improve quality of released product	• Greater process risk when compared to waterfall process model
• End result can more closely match user requirements	• More difficult to make customer commitments for product deliverables

Figure 2

		Duration of Code/Debug/Fix Cycle		
		Short	Medium	Long
Complexity	Low	Spiral	Spiral	Waterfall
	Medium	Spiral	Spiral	Waterfall
	High	Waterfall	Waterfall	Waterfall

Figure 3

4. Putting it Together in the New Reality

Our experience has shown that the key determinants of appropriate process models are the duration of the code/test/fix cycle, and the complexity of the component being implemented (see figure 3).

Software modules with low or medium complexity, and that are implemented using technologies with a short or medium code/debug/fix cycle, are more suited to the Spiral process model. Modules with inherently high complexity or which must use technologies that have a long code/debug/fix cycle, are more suited for the Waterfall process model.

New technologies such as Java™ and HTML have increasingly shorter code/test/fix cycles. Object-oriented approaches can increase reuse and reduce development cycles. Client-server applications can be developed much more quickly than can host-based applications. Use of these technologies makes it possible for more projects to implement and realize the benefits of spiral development processes.

In reality, however, many software projects will be a mix of technologies. Traditional and alternative process models have ignored the mix of complexity and technologies in many projects.

The Hybrid process model, shown in Figure 4, enables a project to realize the benefits of Spiral and Waterfall development processes, plus some additional benefits:

- Adds structure for more deterministic project management
- Allows frequent customer input during development
- More flexible to adapt to requirements and design changes
- Fits well with customer-funded development business strategies
- End-product can be of higher quality and more closely meet end-customer needs
- Can be implemented within phase-gate process models

This Hybrid process model has three key sections:

1. Architecture and planning
2. Iterative development and deployment of most major components
3. Linear development of those components which cannot be developed iteratively

Architecture and planning (including requirements gathering) must be completed first; these form the foundation for subsequent project activities. Iterative development literally “runs circles around” the linear development component. Executing these activities in parallel minimizes time to market.

The following sections describe these process components in more detail.

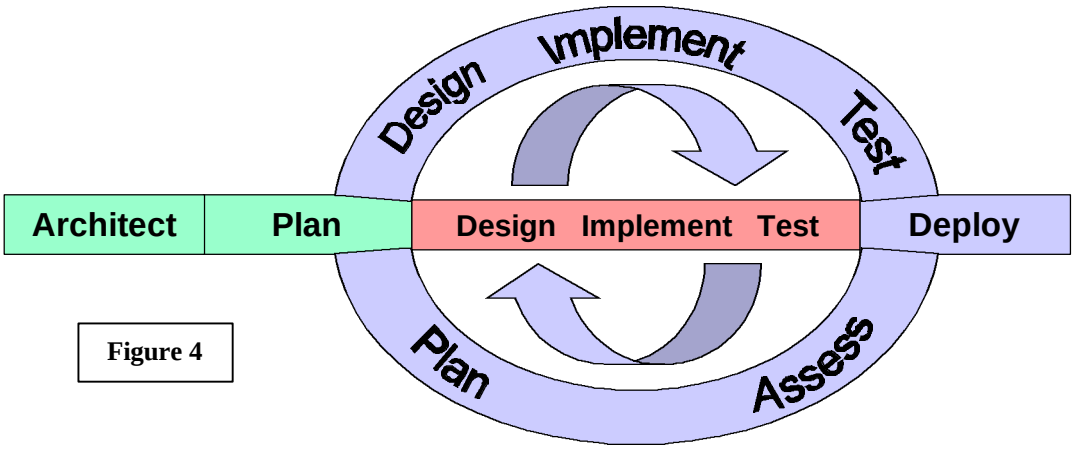

Figure 4

4.1. Architecture and Planning

Architectural design provides the technical foundation for all project modules. It ensures that the modules will "fit together" when they are integrated to form the final projects. Software modules implemented iteratively can be integrated as part of each iteration, although linearly developed modules will not be available until complete. Therefore, a well-described common architecture helps ensure that iteratively developed software modules will integrate well with linearly developed software modules and other project components.

Requirements gathering occurs in parallel with Architecture and Planning activities.

Software development is an inherently complex undertaking. Both technical and project planning are essential to optimize quality, schedule, and cost for the project. Some key planning decisions unique to the Hybrid process model include:

- Which components can be developed iteratively versus linearly
- What target feature/functionality will be implemented in each iteration
- How many iterations will be completed
- To what extent will modules be deployed at the conclusion of each iteration

The more components that can be implemented iteratively, the greater the benefits that will be seen by the project. A well-designed architecture can greatly reduce the complexity of many software modules – increasing the number that can be implemented iteratively.

The number of iterations to be completed can be set based on two factors:

- The duration of an iteration
- The duration of the linear project component

We would suggest three moderately effective iterations would produce a high-quality product that very closely meets end-customer requirements. For example, if each iteration is 75% successful at realizing end-customer requirements, the product will have achieved more than 98.4% at the conclusion of the third iteration. This compares to 90% as the theoretical limit for a single iteration! Project execution will be most efficient where the duration of an iteration is set to one-third the duration of the linear project component.

4.2. Iterative Development

Iterative development is performed on software modules selected according to Section 4.1. Each iteration should follow five general steps:

1. Design
2. Implement
3. Test
4. Assess
5. Plan

First, detailed design of each software module should be completed. Note that the complexity of the design effort should match the complexity of the module's feature/functionality to be implemented as part of the current iteration – to the greatest extent possible.

Design is followed by implementation of the target feature/functionality for the iteration. After implementation, it must be tested – both by itself and integrated with other iteratively developed components.

After testing, assessment should be completed to gather end-customer feedback. Note that assessment can be performed by deploying – in a controlled manner – developed software. Controls should allow feedback to be gathered in a structured way that allows it to be incorporated into the next iteration. After feedback is gathered, it can be used in conjunction with the overall project plan to plan the next iteration – in essence, completing a project plan focused on the next iteration.

4.3. Linear Development

Linearly developed modules are implemented using the traditional waterfall sequence to design, implement, and test the required functionality. Since other modules are being implemented iteratively, however, there may be some unique considerations. For example, simulation or "stub" modules may need to be implemented to allow the iteratively developed modules to function on each iteration.

5. Success Factors

Successful use of the Hybrid process model puts more emphasis three key areas:

- Architecture
- Project management
- Risk management

A well designed, unified architecture is essential for successful implementation of the Hybrid process model. Because of a higher degree of parallel implementation, there is greater risk of serious problems when the modules are integrated. A unified

architecture will significantly reduce this risk. A properly designed architecture will also reduce the complexity of each software module – which will allow more modules to be implemented iteratively, and increase the level of parallel implementation that can be realized. Ultimately, the impact can be to reduce time to market.

Project management is the second key success factor for implementation of the Hybrid process module. At any point in the project, there will be quite a bit of diverse activity occurring concurrently – for example, implementation and end-customer validation. Strong project management is essential to manage this activity and keep the project under control. The first key activity is initial planning, prior to the start of iterative development. Continuous monitoring and adjustment is required both on the iterative portion of the project, and the linear portion. In addition, at the conclusion of each iteration, planning for the next iteration needs to be completed – which should be linked with the linear portion of the project.

Risk management is the key final success factor for the Hybrid process model. As with architecture and project management, risk management is a key success factor for any software project. Aggressive, proactive risk management will prevent problems before they occur, reduce the time needed for "fire-fighting", and ultimately reduce time to market. Effective risk management will help project teams optimize the Hybrid process model to their project, and minimize the impact of process changes.

Several risk management tools fit well with the Hybrid process model. For example, the Project Self-Assessment Kit from **Kulik & Lazarus Consulting, Inc.**, can be used on both the iterative and linear components for measurement of key project success factors.

6. Conclusion

Relatively recent technological advantages are driving fundamental change in software development. Java™, Object-Oriented methods, the Internet, Intranets, etc., are changing both the economics and underlying foundation of how software is developed. The two prevalent software development process models – the Waterfall and Spiral models – are generally well understood by many software organizations. Both have their strengths and weaknesses, but neither provides optimal performance in the reality of today's business and technological environment.

The Hybrid process model is a powerful way to manage today's software projects. With a well-constructed architecture, most complex systems will include components that lend themselves to iterative development, and other components that are more suited for linear development. The Hybrid process model enables an organization to implement as many components as possible using iterative development, aligned with a linear development activity other components.

Success factors for implementing the Hybrid process model include:

- Unified architectural design
- Effective project management
- Aggressive, proactive risk management

Proper attention to these success factors will insure optimal implementation of the Hybrid process model, reducing time to market, improving quality, and enhancing product acceptance by end-customers.

7. References

1. Boehm, Barry W., "A Spiral Model of Software Development and Enhancement", IEEE Computer, May 1988.
2. Royce, W.W., "Managing the Development of Large Software Systems: Concepts and Techniques", Proc. ICSE 9, Computer Society Press, 1987.
3. Parnas discusses an Incremental Software Development Model in IEEE Transactions on Software Engineering, March 1979.
4. Mills, H.D., Dyer, M., and Linger, R.C., "Cleanroom Software Engineering", IEEE Software, September 1987.
5. Hanna, Mary Alice, "Beyond the Waterfall Lies a Brave New World", Software Magazine, June 1991.
6. Campbell, Ian, "The Intranet: Slashing the Cost of Business", International Data Corporation, 1997.
7. Phil Neches, Teradata Corporation, on or about 1985.

***Peter Kulik** is Managing Partner of **Kulik & Lazarus Consulting, Inc.** With more than 10 years experience in all aspects of software development, he holds an MS in Engineering Management with the thesis "Practical Quantitative Methods for Software Development Process Management", a Certificate in Economics and Finance, and a BS in Electrical Engineering. He can be reached via e-mail at pkulik@klci.com.*