A Software Development Simulation Model of a Spiral Process

Carolyn Mizell[(1)], Linda Malone[(2)]

(1) NASA/EA-C, Kennedy Space Center, FL 32899;phone: 321-867-8814;
E-mail: Carolyn.A.Mizell@nasa.gov
(2) Department of Industrial Engineering and Management Systems, 4000 Central Florida Blvd.,
P.O. BOX 162993, Orlando, FL 32816-2993 ;Phone: 407 823-2204;
E-mail: lmalone@mail.ucf.edu

ABSTRACT

This paper will present a discrete event simulation model of a spiral development lifecycle that can be used to analyze cost and schedule effects of using such a process in comparison to a waterfall process. There is a need for simulation models of software development processes other than the waterfall due to new processes becoming more widely used in order to overcome the limitations of the traditional waterfall lifecycle. The use of a spiral process can make the inherently difficult job of cost and schedule estimation even more challenging due to its evolutionary nature, but this allows for a more flexible process that can better meet customers' needs. Cost figures for the spiral process may initially appear higher but can ultimately prove to be much more realistic or even lower than the final cost for a waterfall process since the data on which they are based is continuously updated. The goal of this work is to present a preliminary model that provides insight into the impacts of selecting a spiral development approach and that demonstrates the usefulness of such a model in order to encourage the development of more detailed spiral development models.

Keywords: Software Development Project Management, Spiral Development, Discrete Event Process Model, Project Cost Estimation

1- INTRODUCTION

Software simulation models based on the traditional waterfall process exist as discrete event models and system dynamics models, but the literature points out the need to develop simulation models of other lifecycle processes. Since the traditional waterfall lifecycle development method has its drawbacks (including such poor performance as to result in project cancellation prior to completion), other lifecycle approaches are gaining popularity, especially for large and complex software development projects.

Of particular interest is the spiral development lifecycle which is gaining popularity for use in large government and military projects. The goal of this work is to provide a new discrete event simulation model of a spiral development lifecycle and to use the new model to analyze the cost and schedule estimation effects of using such a development approach.

This work is new because discrete event simulation models have not been used for comparison of cost and schedule estimation differences between the waterfall and the spiral development lifecycles. The benefits of performing such a comparison are important because experience with the waterfall process shows that only one-third of software development projects that follow a waterfall process are completed successfully within budget and schedule(Brooks[6]. Also, it is not realistic to assume that large and complex software development projects can follow a once-through sequential process like the waterfall process. Most large projects that attempt to follow a waterfall process end up being evolutionary and iterative in nature as requirements change and evolve, and this has a substantial impact on cost and schedule estimates.

The spiral development model was developed by Barry Boehm and is based on experience with large government software projects [4]. The goal was to provide a model with greater flexibility that could better serve these types of projects.

Boehm describes the spiral development model as a risk-driven process model generator that consists of a cyclic approach to incrementally implementing a system while decreasing the degree of risk [1].

The waterfall model is the traditional lifecycle development approach that was introduced by Winston Royce in 1970 [13]. This model consists of a sequential cycle of activities that include requirement analysis, design, coding, testing, and support. The spiral model can be thought of as a repeating waterfall model that emphasizes risk assessment and that is executed in an incremental fashion. Each pass through the spiral model consists of risk assessment, requirements analysis, design, coding, testing, delivery, and evaluation. Figure 1 shows a graphical representation of a single increment of a spiral model.

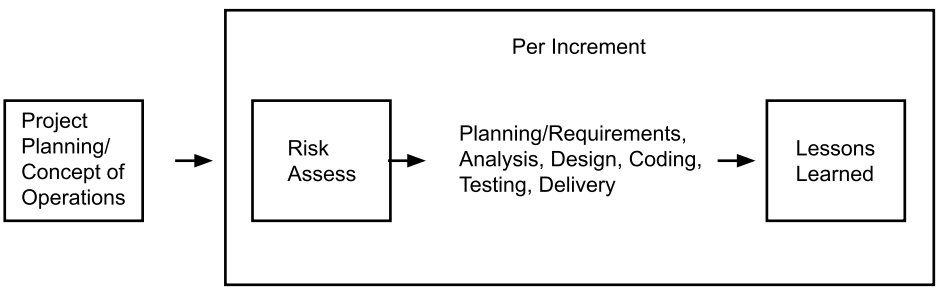

Figure 1 Spiral Development Model

This process has been shown to be successful in a variety of environments, including NASA's Marshall Space Flight Center [10]. Fig. 1 depicts the fact that the waterfall phases of requirements analysis, design, coding, testing and delivery are accomplished for each increment along with additional phases of risk assessment and lessons learned. Initial overall project planning and development of a concept of operations is accomplished prior to the first increment to establish high level requirements and the overall conceptual

framework for the product. Detailed requirements evolve during the spiral portion of the lifecycle. The number of spiral passes that must occur for each increment depends on the areas of risk and the development state of the product. The goal of the increments is to provide the customer with limited, but useable operational capability. The customer will eventually get full operational capability after several increments.

A key difference between the waterfall and spiral models is that the waterfall model considers requirements to be fixed from the beginning of the project with fixed documents being produced as a result of each phase of the lifecycle. Therefore, this process is not flexible enough for some projects, especially when requirements are not known at the beginning of a project. Changes in requirements later in the process lead to major cost and schedule overruns, especially for very large projects. This approach can lead to other problems such as: delayed integration, late risk resolution, and focus on documents and review meetings as opposed to tangible increments of the product14]. Therefore, more and more software development projects are following lifecycle models other than the waterfall model.

The spiral model is designed to be flexible and to evolve into other types of models such as evolutionary or even waterfall based on the results of risk assessments and where key risks exist [2]. It is also considered product driven rather than document driven like the waterfall process [14]. The government and military are using this process more and more in order to overcome the limitations of the more traditional processes. The US Dept. of Defense has determined that the spiral development model is the preferred method/process for software-intensive development lifecycles [18].

Disadvantages of the spiral model include continual deferral of planned functionality in order to stay on schedule and within budget. The deferral of work can accumulate until an insurmountable amount of work is left for the end of project. This is known as the “Death Spiral” [7]. Even though projects that follow a spiral development process may appear to cost more and take longer, they should better meet customers’ needs and expectations and often provide a more realistic cost and schedule.

A discrete event process simulation model of the spiral development lifecycle process will allow the evaluation of different scenarios in projects using this type of approach. A process model for spiral development can also enable analysis of the effects of using this type of approach versus the traditional approach in terms of effort and schedule. The waterfall approach assumes all requirements are known up front and yet this is often unrealistic. Changes to requirements will affect size and therefore, the cost and schedule of the project. In order to handle changes that occur as a project evolves, NASA’s Manager’s Handbook for Software Development recommends that a minimum of five re-estimates should be made after the initial estimate at key life cycle phase points [15]. This will allow for more accurate estimates to be developed as the project progresses and as more information becomes available. The handbook states that the

uncertainty of an estimate will decrease from completely uncertain at the initial estimate to almost certain with the sixth estimate after system test.

Cost estimation is an especially difficult area of software development project management. The impacts of uncertainty in key areas such as product size, productivity, and defect injection rates can dramatically affect a project's cost and schedule. The job of estimating becomes even more difficult when requirements are allowed to evolve throughout a project as is the case for a spiral lifecycle process. The same unknowns of size, productivity, and defect injection rates exist, but there is also the additional unknown of the number of spirals that will need to be completed before an incremental product is delivered. The evolutionary nature of the process allows requirements to change and this makes the job of estimating size even more difficult and uncertain. Since most software development projects do undergo changes, it is beneficial to consider the impacts on cost and schedule which can be used to develop a more realistic estimate.

Fig. 2 shows the layout for a typical software development process simulation model of a waterfall type project.

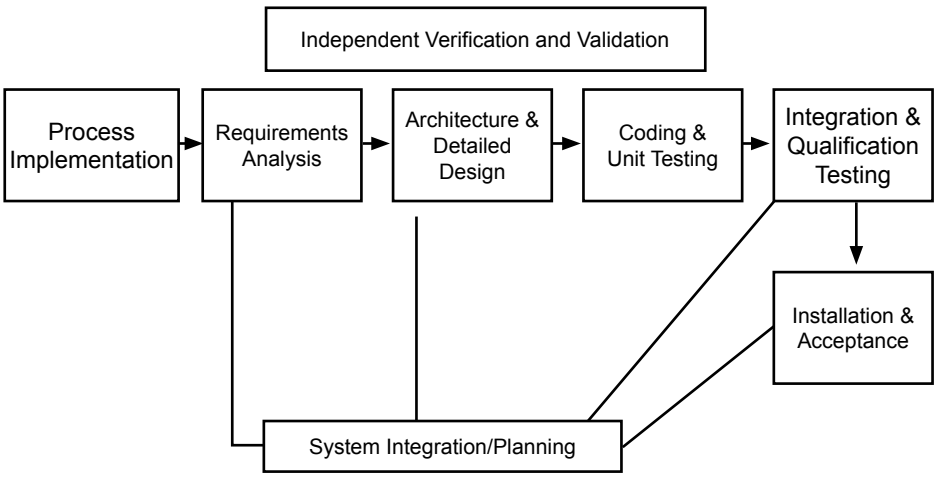

Figure 2 Waterfall Process Simulation Model

An existing software development process simulation model will serve as the core for the development of this spiral process simulation model. The Process Analysis Tradeoff Tool, PATT ©, is a discrete event process simulation model that was developed for NASA to assess the benefits of Independent Verification and Validation (IV&V) on the IEEE 12207 software development process which is a waterfall type process and which is represented by Fig. 2 [12]. The model typically uses industry average data for input variables such as product size, productivity (LOC/Hr), and defects (per KSLOC). The user provides number of resources, % of overall effort that should be allocated to each process step, and

the number of desired staff for each step. The model outputs the size, effort, rework effort, entire process duration, average duration, number of injected defects, detected defects, and corrected defects. For this article, data that is typical for the NASA software development environment will be used for analysis with the spiral model.

2- MODEL DEVELOPMENT

The approach to developing the spiral model was to add steps to the existing PATT waterfall model and to repeatedly run through the steps to represent increments for the entire project.

Fig. 3 shows how PATT serves as the core of the spiral model.

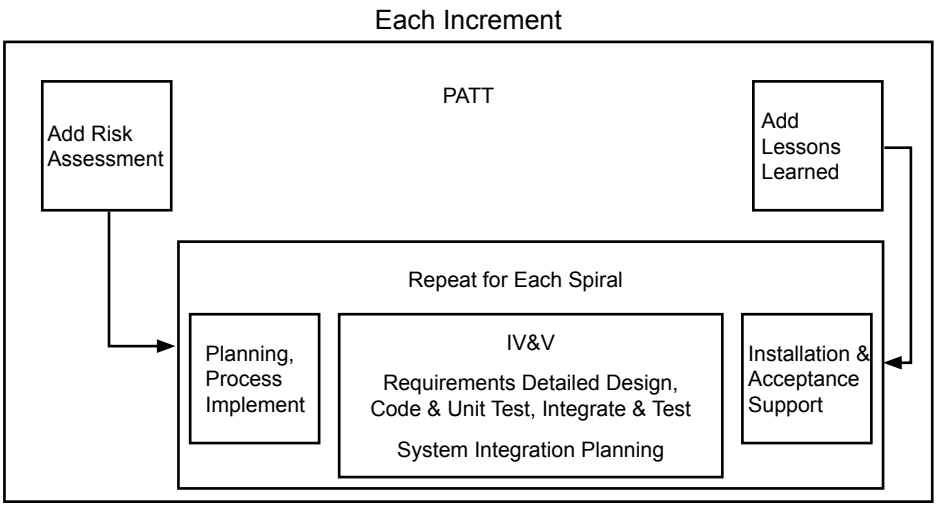

Figure 3 Use of PATT for Spiral Model

Verification of the Incremental Spiral Model was straightforward since it was based on the IEEE 12207 model that was previously verified and validated. Changes to the original model were made incrementally and the model was verified to run without issues and to produce reasonable results. Final verification was accomplished by forcing the incremental spiral model to behave as a waterfall model and comparing results with that obtained from similar runs of the IEEE 12207 model. The output data from both models was similar when the spiral model was limited to one increment and one spiral, thereby reducing it to a similar process. Expected variances between the model outputs were explained due to the additional steps of risk assessment and lessons learned and % overall effort differences by lifecycle type.

3 -MODEL INPUTS

The following information is input to either the IEEE 12207 PATT model or the spiral model: size of product (lines of code), number of resources, productivity (lines of code/hour), defect injection rate (defects/ksloc), % of overall effort for each process step, and desired staff for each process step. Industry averages or theoretical probability distributions can be used for productivity and defect injection rate.

The following table provides guidelines for percentage of effort by phase that is based on several literature sources [3] [16] [5].

Table 1 Percentage of Effort by Phase for Waterfall and Spiral Processes

Waterfall Activity	% Effort (Boehm 2000)	% Effort (SEL)	Spiral Activity	% Effort
Plan/Requirements	8%	12%	Risk Assessment	5%
Product Design	18%	8%	Planning/Requirements Analysis	5%
Detailed Design	25%	15%	Product Design	20%
Coding/Unit Test	26%	40%	Detailed Design	17%
Integration/Testing	31%	25%	Coding/Unit Test	26%
			Integration/Testing	22%
			Lessons Learned	5%

Note that the % effort for a waterfall process based on data found in Estimating with COCOMO II totals to 108%. This is due to the assumption that all plans and requirements for a waterfall approach will be completed prior to the beginning of a project and thus the total percent effort is equal to 100% without including the planning/requirements activity. Data in the second % Effort column for waterfall is based on Software Engineering Laboratory data and totals to 100%. Data from Table 1 will serve as the % of overall effort input for the models.

Industry averages from the literature suggest an average productivity of 3.6 LOC/Hr [11] and defect injection rates of 60 defects per thousand lines of code [8]. Since this work is based on the NASA environment, it was necessary to populate the model with productivity and defect injection rates more typical for this environment. These factors tend to be lower for the NASA environment, with an average productivity of about 3.2 LOC/Hr [15] and defect injection rates of approximately 30 errors per KSLOC [8]. Since productivity and defect injection rates are affected by project factors that are very dynamic, the use of average values does not adequately account for the range of values of these key parameters. In order to better represent the range of values that can occur for productivity and defect injection rates in a similar development environment, probability distributions were developed and used in the models.

These distributions are based on data collected during the 25 year history of the Software Engineering Laboratory [15] [9]. The Software Engineering Laboratory (SEL) began collecting data for NASA flight software development projects at Goddard Space Flight Center in 1976 and served as a major resource in software process improvement activities. Extensive project and product data was collected for over 200 projects and is available to the public. For the purposes of this research, subsets of the data that were collected and organized by the NSF Center for Empirically Based Software Engineering [9] were used. Probability distributions were fitted for productivity data and defect injection data using ExpertFit software to determine which theoretical probability distribution best represented the data set. In all cases, the Kolmogorov-Smirnov test was passed with $\alpha = 0.05$.

The following table provides the distributions and parameter values used as inputs to the models for defect injection rates:

Table 2 Defect Injection Rates by Phase

Requirements	**Lognormal (2.62, 7.1)**
Design	**Lognormal (17.13, 73.35)**
Coding	**Weibull (28.39, 0.81)**
Testing	**Exponential (40.9)**

A distribution of **Erlang (1.36, 3)** was used for productivity**.**

4 -ANALYSIS

In general, large NASA software development projects begin with a bottoms-up estimate prior to the start of a project. For this analysis, assume a staff of 350 was available. The bottoms-up estimate included size estimates for each increment that added up to 3.425 Million LOC. Funding was provided based on a total size of 3.8 Million LOC, 1400 labor years, and a schedule of five years. At this point, a Concept of Operations with high level requirements was completed. The goal was to follow an incremental development process that consisted of 10 increments and allowed requirements to evolve during the project.

Of interest is the question of how does a waterfall approach compare with a spiral approach, especially when developing early estimates. First, the IEEE 12207 process model and the spiral model will be run with the same input data so that the output effort and duration data from each can be compared. For this first analysis, the assumption will be that the size estimate is accurate (which would be highly unlikely at the beginning of a project and thus favors the waterfall model), so a total size of 3425 KSLOC will be input to the IEEE 12207 model. Table 3 provides size for each increment of the spiral model:

Table 3 Size Estimate per Increment

Increment Number	Size
1	100
2	250
3	675
4	700
5	650
6	200
7	325
8	300
9	175
10	50

The spiral development approach assumes that there will often be more than one pass through the development phases. Therefore, the discrete event spiral simulation model will include an additional probability distribution for the number of spirals that must be completed within each increment. This will be set to a Uniform [1, 3] distribution for this analysis.

Table 4 provides the output data from each type of model:

Table 4 Comparison of Outputs for Waterfall and Spiral Models

Output	Waterfall	Spiral
Mean Duration	5.9 years	8.27 years
Mean Effort	1391 labor years	1520 labor years

This data shows that a budget of 1400 labor years and a schedule of five years are very risky for this project, even if a waterfall process is followed. This data also shows that a spiral process will take longer than a waterfall process and this should be expected, although it may not be considered when preparing a budget. The initial estimates for length of effort and cost will generally show that the spiral process will take longer and cost more than the waterfall process because the process is repeated and has additional process steps. (However, recall that experience with the waterfall process shows that only one-third of software development projects that follow a waterfall process are completed successfully within budget and schedule [6] so it is clear that the spiral process provides more realistic estimates of cost and effort although they are off putting to management because of appearing higher at the project start.). This makes a onetime estimate done at the beginning of a project very impractical. A rough estimate can be developed, but detailed phased funding should be considered so that more accurate estimates can be developed based on an incremental basis rather than for the entire project. The intended benefit of spending more time and money on such a process is that the user will get a better product and

will get incremental functionality with each delivery.

Size growth due to requirements changes and unknowns can be extensive. The literature points out that very early size estimates are likely to be much lower than the actual final size of the project due to requirements changes and unknowns[16,5]. This is often the case with either the waterfall or spiral process, even though requirements should not drastically change in a true theoretical waterfall lifecycle. Requirements evolution is a key part of a spiral process and therefore, plans for accommodating size changes should be considered when estimating a project that will follow such a process. This is a more realistic view of the situation for large, complex projects than a theoretical waterfall process.

Since the project was set up for ten increments, each increment will have a uniform size distribution with parameters: **(Increment Size Estimate, 2 X Increment Size Estimate) LOC**. The uncertainty in the size at this point in the project when only high level requirements are understood is based on data from the literature [5]. Table 5 provides the size inputs to the model for each increment:

Table 5 Incremental Size Distributions

Increment	Size (KSLOC)
1	Uniform [100, 200]
2	Uniform [250,500]
3	Uniform [675,1350]
4	Uniform [700,1400]
5	Uniform [650,1300]
6	Uniform [200,400]
7	Uniform [325,650]
8	Uniform [300 ,600]
9	Uniform [175, 350]
10	Uniform [50,100]

At the beginning of each increment, a draw will be taken from a probability distribution for the number of spirals that are to be completed and the total increment size will be divided by the number of spirals to provide the size for each spiral. This will be compared to the waterfall model. For this analysis, the waterfall model will be run with a size distribution of Uniform [3425, 6850] and the % of effort in Table 6 that totals to 100% to represent a more realistic waterfall where requirements analysis is done as part of the project.

Table 6 Comparison of Spiral Model to Waterfall Model with Uncertain Requirements

	Waterfall	Spiral
Mean Duration	9.89 years	14.67 years
Mean Effort	2356 labor years	2414.6 labor years

This data also shows that the spiral process will appear to initially cost more and take longer than a waterfall approach. This initial difference is expected since a waterfall process theoretically proceeds sequentially through each phase one time with no substantial changes in requirements. The spiral process, however, is more representative of the typical iterative nature that most software development projects actually follow. Therefore, the process of developing cost and schedule estimates that consider uncertainty in a development process that is iterative in nature should lead to a more realistic estimate. The analysis also shows the significant impact size uncertainty has on a project's cost and schedule for either lifecycle approach. A benefit of following a spiral process that consists of multiple spirals per increment is that particular risk areas can be resolved by expending only a portion of the increment's budget. The spirals serve as a risk resolution plan that should enable deliverable functionality for each increment. The fact that this type of process will provide benefits but may take longer and cost more must be considered when developing estimates for a project that will follow this approach.

5- CONCLUSION

This work has developed a new software development process model that enables assessment of an incremental or spiral lifecycle approach. This provides a new insight into the effects of developing cost and schedule estimates for an iterative development process that is a more realistic representation of most large software development projects. The traditional waterfall process provides for a structured sequential process, but the literature shows that this is often not realistic since requirements tend to evolve and phases of the lifecycle may need to be repeated.

Using estimation data from the NASA environment, this work has analyzed a project's early cost and schedule estimate using both a waterfall and spiral approach. The results show that the cost and schedule estimates for a spiral process may be higher than for a waterfall process and this is in agreement with military experiences (Brown 2004). This type of process should provide a quality product that better meets users' needs by allowing evolution of requirements and by providing functionality with each increment. This process emphasizes risk management and is designed to be flexible. This should lead to more realistic budgetary planning since it is obvious that requirements will change and cost and schedule will be affected throughout the project. This work also shows that uncertainty in areas such as size, productivity, and defects should be accounted for when developing an estimate, no matter which lifecycle is selected. The

spiral process model can be used throughout a project to analyze the project as more information becomes available. For instance, data from early increments can serve as inputs to the model and an estimate to complete based on this data can be assessed. More work can be done to refine the spiral model based on other projects' experiences.

REFERENCES

[1] B. Boehm, "Spiral Development: Experience, Principles and Refinements," Proc. Software Engineering Institute Spiral Development Workshop, p.49, 2000.

[2] B. Boehm and F. Belz, "Experiences with the Spiral Model as a Process Model Generator," IEEE pp. 43-45, 1990.

[3] B. Boehm, Software Engineering Economics. Englewood Cliffs, N.J.: Prentice Hall, 1981.

[4] B. Boehm, "A Spiral Model of Software Development and Enhancement," IEEE Computer, vol. 21, pp.61-72, 1988.

[5] B. Boehm, C. Abts, et al, Software Cost Estimation with COCOMO II. Upper Saddle River, N.J.: Prentice Hall, 2000.

[6] A. Brooks, "Estimating Cost with Project Management Software Can Boost Efficiency," Computing Canada, vol. 24, p.34, 1998.

[7] D. Brown, "Evolutionary Acquisition and Spiral Development," Proc. Defense Acquisition University, p. 34, 2004.

[8] CeBASE, "EWorkshop on Software Inspections and Pair Programming," http.www.cebase.org. 2004.

[9] CeBASE, "Software Engineering Laboratory Data," http.www.cebase.org. 2005.

[10] D. Hendrix and M.Schneider, "NASA's TReK Project: A Case Study in Using the Spiral Model of Software Development," Communications of the ACM, vol. 45: pp. 152-159, 2002.

[11] C. Jones, Software Assessments, Benchmarks, and Best Practices, Boston, MA,: Addison-Wesley, 2000.

[12] D. Raffo and W. Wakeland, "Assessing IV&V Benefits Using Simulation," Proc. 28th Annual NASA Goddard Software Engineering Workshop, 2003.

[13] W. Royce, Software Project Management: A Unified Framework, Boston, MA.: Addison-Wesley, 1998.

[14] SEL, “Manager’s Handbook for Software Development,” Software Engineering Laboratory Series, 1990.

[15] SEL, “Cost and Schedule Estimation Study Report,” Software Engineering Laboratory Series, 1993.

[16] SEL, “An Overview of the Software Engineering Laboratory,” Software Engineering Laboratory Series, p.55, 1994.

[17] D. Surber, “Spiral Evolution Applied to Legacy Avionics Systems,” IEEE A&E Systems Magazine, vol.19, pp.3-9, 2004.