

Spiral Development: Experience, Principles, and Refinements

**Spiral Development Workshop
February 9, 2000**

Barry Boehm, edited by Wilfred J. Hansen

July 2000

SPECIAL REPORT
CMU/SEI-2000-SR-008



**Carnegie Mellon
Software Engineering Institute**

Pittsburgh, PA 15213-3890

Spiral Development: Experience, Principles, and Refinements

**Spiral Development Workshop
February 9, 2000**

CMU/SEI-2000-SR-008

Barry Boehm, edited by Wilfred J. Hansen

July 2000

COTS-Based Systems

Unlimited distribution subject to the copyright.

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Norton L. Compton, Lt Col., USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2000 by Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (<http://www.sei.cmu.edu/publications/pubweb.html>).

Table of Contents

Abstract	vii
1 Introduction	1
1.1 Success Stories from the Workshop	1
1.2 The Spiral Development Model	3
2 The Invariants and Their Variants	5
2.1 Spiral Invariant 1: Concurrent Determination of Key Artifacts (Ops Concept, Requirements, Plans, Design, Code)	6
2.2 Spiral Invariant 2: Each Cycle Does Objectives, Constraints, Alternatives, Risks, Review, Commitment to Proceed	9
2.3 Spiral Invariant 3: Level of Effort Driven by Risk Considerations	11
2.4 Spiral Invariant 4: Degree of Detail Driven by Risk Considerations	13
2.5 Spiral Invariant 5: Use of Anchor Point Milestones: LCO, LCA, IOC	14
2.6 Spiral Invariant 6: Emphasis on System and Life Cycle Activities and Artifacts	17
3 Anchor Point Milestones	20
3.1 Detailed Descriptions	20
3.2 Relationships to Other Process Models	22
Evolutionary Development	22
Rational RUP Phases	23
WinWin Spiral Model	24
MBASE Electronic Process Guide	25
4 Summary	29
References	33
Acronyms	37

List of Figures

Figure 1: Original Diagram of Spiral Development	2
Figure 2: Two System Designs: Cost vs. Response Time	7
Figure 3: Models Excluded: Sequential Phases without Key Stakeholders	10
Figure 4: Pre-Ship Test Risk Exposure	12
Figure 5: Scientific American Order Processing	18
Figure 6: Anchor Points and the Rational RUP Phases	24
Figure 7: The WinWin Spiral Model	25
Figure 8 EPG Top-Level Outline of Activities, Artifacts, and Agents	26
Figure 9 EPG Diagram of the Inception Phase	27
Figure 10 EPG Outline and description of Risk Driven Analysis	27
Figure 11 The "Outputs" section of the description on the right of Figure 10.	28

List of Tables

Table 1: WinWin Spiral Anchor Points (with risk-driven level of detail for each element)	21
Table 2: Invariants of Spiral Processes: Name, Rationale, and Variants	30
Table 3: Hazardous Spiral Look-Alikes	31

Abstract

Spiral development is a family of software development processes characterized by repeatedly iterating a set of elemental development processes and managing risk so it is actively being reduced. This paper characterizes spiral development by enumerating a few “invariant” properties that any such process must exhibit. For each, a set of “variants” is also presented, demonstrating a range of process definitions in the spiral development family. Each invariant excludes one or more “hazardous spiral look-alike” models, which are also outlined. This report also shows how the spiral model can be used for a more cost-effective incremental commitment of funds, via an analogy of the spiral model to stud poker. An important and relatively recent innovation to the spiral model has been the introduction of anchor point milestones. The latter part of the paper describes and discusses these.

Editor’s Note: This document began as a set of slides prepared and annotated by Barry Boehm and presented by him at the Spiral Development Workshop, February 2000. With Barry's consent, I un-

dertook the task of converting these slides to the text you now see. The original slides are available on the workshop Web site: <http://www.sei.cmu.edu/cbs/spiral2000/Boehm>.

1 Introduction

This presentation opened the Workshop on Spiral Development Experience and Implementation Challenges held by the University of Southern California (USC) and the Software Engineering Institute (SEI) on February 9-11, 2000 at USC. The workshop brought together leading executives and practitioners with experience in doing spiral development of software-intensive systems in the commercial, aerospace, and government sectors. Its objectives were to distill the participants' experiences into a set of critical success factors for implementing and conducting spiral development, and to identify the most important needs, opportunities, and actions to expedite organizations' transition to successful spiral development. For the workshop, "development" was defined to include life cycle evolution of software-intensive systems and such related practices as legacy system replacement and integration of commercial-off-the-shelf (COTS) components. Although of greatest utility for software developments, the spiral model can also be used to develop hardware or integrate software, hardware, and systems.

To provide a starting point for addressing the workshop objectives, I have tried in this talk to distill my experiences in developing and transitioning the spiral model at TRW; in using it in system acquisitions at the Defense Advanced Research Projects Agency (DARPA); in trying to refine it to address problems that people have had in applying it in numerous commercial, aerospace, and government contexts; and in working with the developers of major elaborations and refinements of the spiral model such as the Software Productivity Consortium's (SPC) Evolutionary Spiral Process (SPC) [SPC 94] and Rational, Inc.'s Rational Unified Process (RUP) [Royce 98, Kruchten 98, Jacobson 99]. I've modified the presentation somewhat to reflect the experience and discussions at the Workshop and this report is a further refinement.

One of the findings of the workshop was a need for a clear and widely understood definition of the spiral development model. The characteristics of the model noted here should suffice as a starting point for this work.

1.1 Success Stories from the Workshop

A number of projects and project frameworks successfully exploiting the spiral model were presented at the workshop, often with supplementary material elsewhere. C-Bridge's RAPID approach has been used successfully to develop e-commerce applications in 12-24 weeks. Its Define, Design, Develop, and Deploy phases use the equivalent of the anchor point milestones (see Section 2.5) as phase gates [Leinbach 00]. The large spiral telecommunications

1.2 The Spiral Development Model

Figure 1 is a redrawing of the original spiral model diagram published by Boehm [Boehm 88]. It captures the major spiral model features: cyclic concurrent engineering; risk driven determination of process and product; growing a system via risk-driven experimentation and elaboration; and lowering development cost by early elimination of nonviable alternatives and rework avoidance. As a result of planning and risk analysis, different projects may choose different processes. That is, the spiral model is actually a risk-driven process model generator, in which different risk patterns can lead to choosing incremental, waterfall, evolutionary prototyping, or other subsets of the process elements in the spiral model diagram.

For a number of reasons, however, the spiral model is not universally understood. For instance, Figure 1 contains some oversimplifications that have caused a number of misconceptions to propagate about the spiral model. These misconceptions may fit a few rare risk patterns, but are definitely not true for most risk patterns. The most significant misconceptions to avoid are: that the spiral is just a sequence of waterfall increments; that everything on the project follows a single spiral sequence; that every element in the diagram needs to be visited in the order indicated; and that there can be no backtracking to revisit previous decisions. In addition to these misconceptions, other similar—but hazardously distinct—processes have been held up as spiral processes.

To promote understanding and effective use of the spiral model, this report more precisely characterizes the spiral model. We begin with a simple overview definition to capture the essence of the model:

The spiral development model is a *risk-driven process model* generator. It is used to guide multi-stakeholder concurrent engineering of software-intensive systems. It has two main distinguishing features. One is a *cyclic* approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk. The other is a set of *anchor point milestones* for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.

Risks are situations or possible events that can cause a project to fail to meet its goals. They range in impact from trivial to fatal and in likelihood from certain to improbable. A risk management plan enumerates the risks and prioritizes them in degree of importance, as measured by a combination of the impact and likelihood of each. For each risk the plan also states a mitigation strategy to deal with the risk. For instance, the risk that technology is unready may be mitigated by an appropriate prototype implementation in an early spiral cycle.

A *process model* answers two main questions:

- What should be done next?

- For how long should it continue?

Under the spiral model the answers to these questions are driven by risk considerations and vary from project to project and sometimes from one spiral cycle to the next. Each choice of answers generates a different process model. At the start of a cycle, all of the project's success-critical stakeholders must participate concurrently in reviewing risks and choosing the project's process model accordingly. (Risk considerations also apply toward ensuring that progress is not impeded by stakeholders' overparticipation).

The *cyclic* nature of the spiral model was illustrated in Figure 1.

Anchor point milestones drive the spiral to progress toward completion and offer a means to compare progress between one spiral project and another. The second half of the report expands on these milestones. It also presents some experience-based refinements of the spiral model developed to address spiral usage problems encountered over the years: evolutionary development, Rational Unified Process (RUP), the WinWin spiral model, and the Model-Based (System) Architecture and Software Engineering (MBA SE) approach. These approaches are compared and contrasted with invariant properties and their variants. Invariant 5 invokes the relatively new concept of "anchor point milestones." These are considered in more depth in the third section. The fourth section presents tables summarizing the material.

2 The Invariants and Their Variants

Those successfully following the spiral model discipline will find that their cycles invariantly display these six characteristics:

1. Concurrent rather than sequential determination of artifacts.
2. Consideration in each spiral cycle of the main spiral elements:
 - critical-stakeholder objectives and constraints
 - product and process alternatives
 - risk identification and resolution
 - stakeholder review
 - commitment to proceed
3. Using risk considerations to determine the level of effort to be devoted to each activity within each spiral cycle.
4. Using risk considerations to determine the degree of detail of each artifact produced in each spiral cycle.
5. Managing stakeholder life-cycle commitments with three anchor point milestones:
 - Life Cycle Objectives (LCO)
 - Life Cycle Architecture (LCA)
 - Initial Operational Capability (IOC)
6. Emphasis on activities and artifacts for system and life cycle rather than for software and initial development.

Subsequent sections describe each of these invariants, the critical-success-factor reasons why it is an essential invariant, and its associated optional variants. Examples are given, including an analogy with stud poker which demonstrates how the spiral model accommodates cost-effective incremental commitment of funds. Many processes are adopted which may seem to be instances of the spiral model, but lack essential invariants and thus risk failure. Each invariant excludes one or more such process models, which we call “hazardous spiral look-alikes.” They are cataloged and pilloried as part of describing the invariants.

2.1 Spiral Invariant 1: Concurrent Determination of Key Artifacts (Ops Concept, Requirements, Plans, Design, Code)

Spiral Invariant 1 states that it is success-critical to concurrently determine a compatible and feasible combination of key artifacts: the operational concept, the system and software requirements, the plans, the system and software architecture and design, and the key code components including COTS, reused components, prototypes, success-critical components, and algorithms.

Summary of Invariant 1

Why invariant
avoids premature sequential commitments to
system requirements, design, COTS,
combination of cost / schedule / performance
Example: "one second response time"

Variants
1a. Relative amount of each artifact developed in each cycle
1b. Number of concurrent mini-cycles in each cycle

Models excluded
Incremental sequential waterfalls
with high risk of violating waterfall model assumptions

Why is this a success-critical invariant? Because sequential determination of the key artifacts will prematurely overconstrain, and often extinguish, the possibility of developing a system which satisfies the stakeholders' essential success conditions. Examples are premature commitments to hardware platforms, to incompatible combinations of COTS components [Garlan 95], and to requirements whose achievability has not been validated, such as the one-second response time requirement Example just below.

Variants 1a and 1b indicate that the product and process internals of the concurrent engineering activity are not invariant. For a low technology, interoperability-critical system, the initial spiral products will be requirements-intensive. For a high-technology, more standalone system, the initial spiral products will be prototype code-intensive. Also, there is no invariant number of mini-cycles (e.g., individual prototypes for COTS, algorithm, or user-interface risks) within a given spiral cycle.

Example: One-Second Response Time

Figure 2 provides an example of the kinds of problems that occur when high-risk requirements are prematurely frozen. In the early 1980s, a large government organization contracted with TRW to develop an ambitious information system. The system would provide more than a thousand users, spread across a large building complex, with powerful query and analysis capabilities for a large and dynamic database.

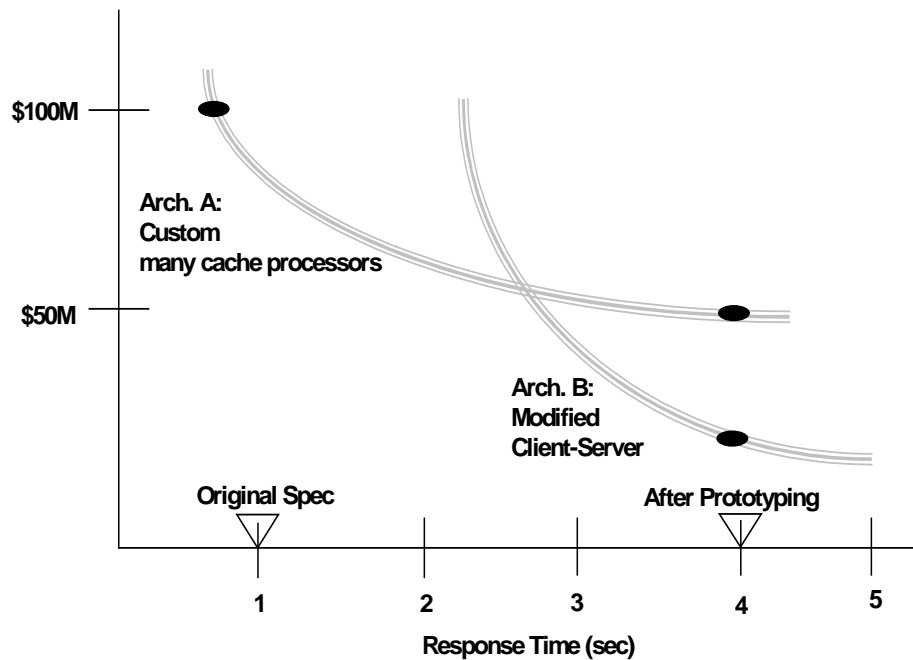


Figure 2: Two System Designs: Cost vs. Response Time

TRW and the customer specified the system using a classic sequential-engineering waterfall development model. Based largely on user need surveys and an oversimplified high-level performance analysis, they fixed into the contract a requirement for a system response time of less than one second.

Two thousand pages of requirements later, the software architects found that subsecond performance could only be provided via Architecture A, a highly customized design that attempted to anticipate query patterns and cache copies of data so that each user's likely data would be within one second's reach. The resulting hardware architecture had more than 25 super-minicomputers busy caching data according to algorithms whose actual performance defied easy analysis. The scope and complexity of the hardware-software architecture brought the estimated cost of the system to nearly \$100 million, driven primarily by the requirement for a one-second response time.

Faced with this unattractive prospect, the customer and developer decided to develop a prototype of the system's user interface and representative capabilities to test. The results

showed that a four-second response time would satisfy users 90 percent of the time. A four-second response time could be achieved with Architecture B, cutting development costs to \$30 million [Boehm 00a]. Thus, the premature specification of a one-second response time inserted the hidden risk of creating an overly expensive and time-consuming system development.

Hazardous Spiral Look-Alike: Violation of Waterfall Assumptions

Invariant 1 excludes one model often labeled as a spiral process, but which is actually a “hazardous spiral look-alike.” This is the use of a sequence of incremental waterfall developments with a high risk of violating the underlying assumptions of the waterfall model. These assumptions are

1. The requirements are knowable in advance of implementation.
2. The requirements have no unresolved, high-risk implications, such as risks due to COTS choices, cost, schedule, performance, safety, security, user interfaces, and organizational impacts.
3. The nature of the requirements will not change very much either during development or evolution.
4. The requirements are compatible with all the key system stakeholders’ expectations, including users, customer, developers, maintainers, investors.
5. The right architecture for implementing the requirements is well understood.
6. There is enough calendar time to proceed sequentially.

These assumptions must be met by a project if the waterfall model is to succeed. If all of these are true, then it is a project risk not to specify the requirements, and the waterfall model becomes a risk-driven special case of the spiral model. If any of the assumptions are untrue, then specifying a complete set of requirements in advance of risk resolution will commit a project to assumptions/requirements mismatches that will lead the project into trouble.

Assumption 1—the requirements are knowable in advance of implementation—is generally untrue for new user-interactive systems, because of the IKIWISI syndrome. When asked for their required screen layout for a new decision-support system, users will generally say, “I can’t tell you, but I’ll know it when I see it (IKIWISI).” In such cases, a concurrent prototyping/requirements/architecture approach is essential.

The effects of invalidity in Assumptions 2, 4, and 5 are well illustrated by the example in Figure 2. The one-second response time requirement was unresolved and high-risk. It was compatible with the users’ expectations, but not with the customer’s budget expectations. And the need for an expensive custom architecture was not understood in advance.

The effects of invalidity in Assumptions 3 and 6 are well illustrated by electronic commerce projects. In these projects the volatility of technology and the marketplace is so high that requirements and traceability updates will swamp the project in overhead. Furthermore, the amount of initial calendar time it takes to work out a complete set of detailed requirements

that are likely to change several times downstream is not a good investment of the scarce time to market available to develop an initial operational capability.

2.2 Spiral Invariant 2: Each Cycle Does Objectives, Constraints, Alternatives, Risks, Review, Commitment to Proceed

Spiral Invariant 2 identifies the activities in each quadrant of the original spiral diagram that need to be done in each spiral cycle. These include consideration of critical-stakeholder objectives and constraints; elaboration and evaluation of project and process alternatives for achieving the objectives subject to the constraints; identification and resolution of risks attendant on choices of alternative solutions; and stakeholders' review and commitment to proceed based on satisfaction of their critical objectives and constraints. If all of these are not considered, the project may be prematurely committed to alternatives that are either unacceptable to key stakeholders or overly risky.

Summary of Invariant 2

Why invariant

- Avoids commitment to stakeholder-unacceptable or overly risky alternatives.**
- Avoids wasted effort in elaborating unsatisfactory alternatives**

Example: "Windows-only COTS"

Variants

- 2a. Choice of risk resolution techniques: prototyping, simulation, modeling, benchmarking, reference checking, etc.**
- 2b. Level of effort on each activity within each cycle**

Models excluded

- Sequential phases with key stakeholders excluded**

Project groups must also guard against having the appearance but not the reality of stakeholder participation by accepting an unqualified member of an integrated product team (IPT). A good set of criteria for qualified IPT members—as described in Boehm and adopted in USAF [Boehm 98, USAF 00]—is to ensure that IPT members are representative (of organizational rather than personal positions), empowered (to make commitments which will be honored by their organizations), knowledgeable (of their organization's critical success factors), collaborative, and committed.

Spiral Invariant 2 does not mandate particular generic choices of risk resolution techniques. However, there are risk management guidelines that suggest, for example, the best-candidate

risk resolution techniques for the major sources of project risk [Boehm 89a]. This invariant also does not mandate particular levels of effort for the activities performed during each cycle. Levels must be balanced between the risks of learning too little and the risks of wasting time and effort gathering marginally useful information.

Example: Windows-Only COTS

Ignoring Invariant 2, can lead to a good deal of wasted effort in elaborating an alternative that could have been shown earlier to be unsatisfactory. One of the current USC digital library projects is developing a web-based viewer for oversized artifacts (e.g., newspapers, large images). The initial prototype featured a tremendously powerful and high-speed viewing capability, based on a COTS product called ER Mapper. The initial project review approved selection of this COTS product, even though it only ran well on Windows platforms, and the Library had significant Macintosh and UNIX user communities. This decision was based on initial indications that Mac and UNIX versions of ER Mapper would be available soon.

However, subsequent investigations indicated that it would be a long time before such Mac and UNIX capabilities would become available. At a subsequent review, ER Mapper was dropped in favor of a less powerful but fully portable COTS product, Mr. SID, but only after a good deal of effort was wasted on elaborating the ER Mapper solution. If a representative of the Mac or UNIX user community had been involved in the early project decisions, the homework leading to choosing Mr. SID would have been done earlier, and the wasted effort in elaborating the ER Mapper solution would have been avoided.

Hazardous Spiral Look-Alike: Excluding Key Stakeholders

Excluded by Invariant 2 is another “hazardous spiral look-alike”: organizing the project into sequential phases or cycles in which key stakeholders are excluded. Examples are excluding developers from system definition, excluding users from system construction, or excluding system maintainers from either definition or construction.

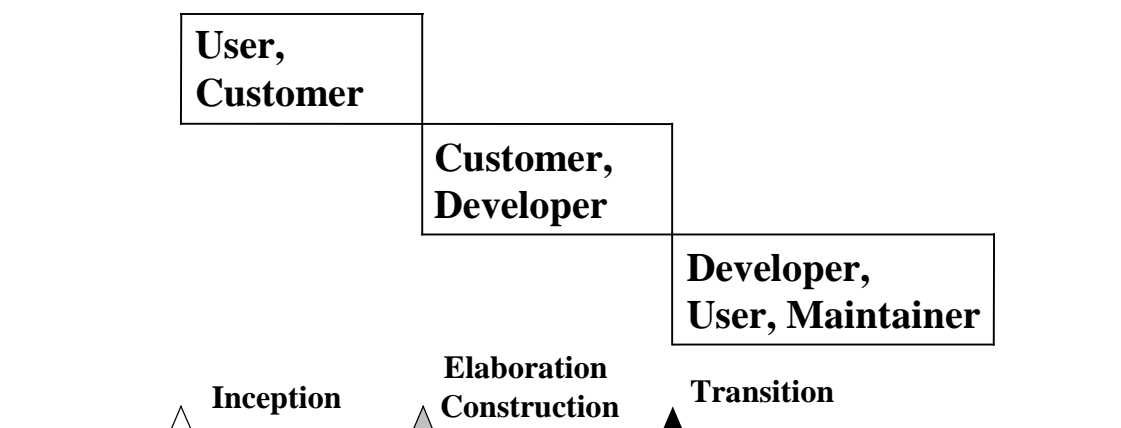


Figure 3: Models Excluded: Sequential Phases without Key Stakeholders

Even though the phases shown in Figure 3 may look like risk-driven spiral cycles, this spiral look-alike will be hazardous because its exclusion of key stakeholders is likely to cause critical risks to go undetected. Excluding developer participation in early cycles can lead to project commitments based on unrealistic assumptions about developer capabilities. Excluding users or maintainers from development cycles can lead to win-lose situations, which generally evolve into lose-lose situations [Boehm 89b].

2.3 Spiral Invariant 3: Level of Effort Driven by Risk Considerations

Spiral Invariant 3 dictates the use of risk considerations to answer the difficult questions of how-much-is-enough of a given activity. How much is enough of domain engineering? prototyping? testing? configuration management? and so on.

Summary of Invariant 3

Why invariant
Determines “how much is enough” of each activity:
domain engineering, prototyping, testing, CM, etc.
Avoids overkill or belated risk resolution

Example: Pre-ship testing

Variants
3a. Choice of methods used to pursue activities:
MBASE/WinWin, Rational RUP, JAD, QFD, ESP, . . .
3b. Degree of detail of artifacts produced in each cycle

Models excluded
Risk-insensitive evolutionary or incremental development

If you plot a project's risk exposure as a function of time spent prototyping, there is a point at which risk exposure is minimized. Spending significantly more time than this is an overkill leading to late market entry and decreased market penetration. Spending significantly less time prototyping is an underkill, leading to premature development with significant delays due to unanticipated snags. Given that risk profiles vary from project to project, this means that the risk-minimizing level of prototyping effort will vary from project to project. The amount of effort devoted to other activities will also vary as a function of a project's risk profile.

Variants to be considered include the choice of methods used to pursue activities (e.g., MBASE/WinWin, Rational RUP, JAD, QFD, ESP) and the degree of detail of artifacts produced in each cycle. Another variant is an organization's choice of particular methods for risk assessment and management.

Example: Pre-Ship Testing

Figure 4 shows how risk considerations can help determine “how much testing is enough” before shipping a product. This can be determined by adding up the two main sources of Risk Exposure, $RE = \text{Probability (Loss)} \cdot \text{Size (Loss)}$, incurred by two sources of loss: loss of profitability due to product defects, and loss of profitability due to delays in capturing market share. The more testing that is done, the lower becomes the risk exposure due to defects, as discovered defects reduce both the size of loss due to defects and the probability that undiscovered defects still remain. However, the more time spent testing, the higher are both the probability of loss due to competitors entering the market and the size of loss due to decreased profitability on the remaining market share.

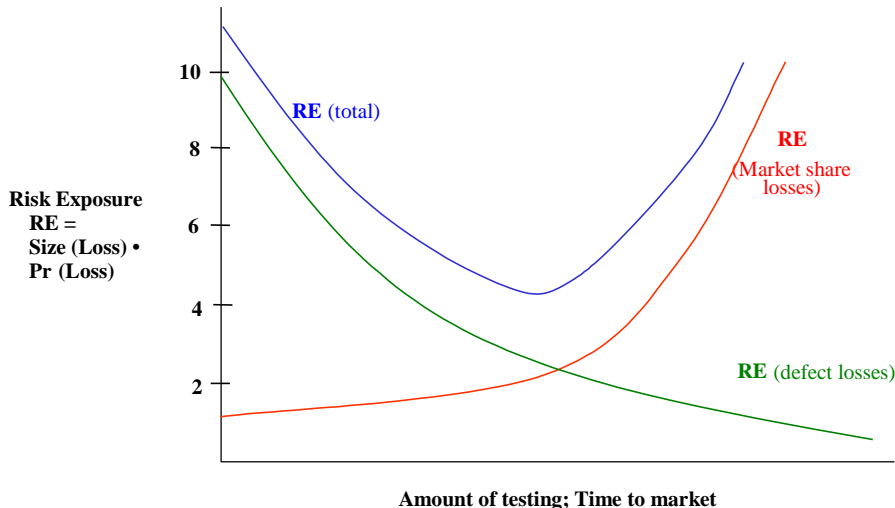


Figure 4: Pre-Ship Test Risk Exposure

As shown in Figure 4, the sum of these risk exposures achieves a minimum at some intermediate level of testing. The location of this minimum-risk point in time will vary by type of organization. For example, it will be considerably shorter for a “dot.com” company than it will for a safety-critical product such as a nuclear power plant. Calculating the risk exposures also requires an organization to accumulate a fair amount of calibrated experience on the probabilities and size of losses as functions of test duration and delay in market entry.

Hazardous Spiral Look-Alikes: Risk Insensitivity

Hazardous spiral model look-alikes excluded by Invariant 3 are

- risk-insensitive evolutionary development (e.g., neglecting scalability risks)
- risk-insensitive incremental development

(e.g., suboptimizing on increment 1 with a point-solution architecture which must be dropped or heavily reworked to accommodate future increments)

- impeccable spiral plans with no commitment to managing the risks identified.

2.4 Spiral Invariant 4: Degree of Detail Driven by Risk Considerations

Spiral Invariant 4 is the product counterpart of Invariant 3: that risk considerations determine the degree of detail of products as well as processes. This means, for example, that the traditional ideal of a complete, consistent, traceable, testable requirements specification is not a good idea for certain product components, such as a graphic user interface (GUI) or COTS interface. Here, the risk of precisely specifying screen layouts in advance of development involves a high probability of locking an awkward user interface into the development contract, while the risk of not specifying screen layouts is low, given the general availability of flexible GUI-builder tools. Even aiming for full consistency and testability can be risky, as it creates a pressure to prematurely specify decisions that would better be deferred (e.g., the form and content of exception reports). However, some risk patterns make it very important to have precise specifications, such as the risks of safety-critical interface mismatches between hardware and software components, or between a prime contractor's and a subcontractor's software.

Summary of Invariant 4

Why invariant
Determines “how much is enough” of each artifact
(OCD, Requirements, Design, Code, Plans) in each cycle
Avoids overkill or belated risk resolution

Example: Risk of Precise Specification

Variants
Choice of artifact representations
(SA/SD, UML, MBASE, formal specs,
programming languages, etc.)

Models excluded
Complete, consistent, traceable, testable requirements
specification for systems involving significant levels of
GUI, COTS, or deferred decisions

This guideline shows when it is risky to over-specify and under-specify software features:

- If it's risky to not specify precisely, DO specify
(e.g., hardware-software interface, prime-subcontractor interface)
- If it's risky to specify precisely, DO NOT specify

(e.g., GUI layout, COTS behavior)

Spiral variants related to Invariant 4 are the choices of representations for product artifacts.

Example: Risk of Precise Specification

One editor specification required that every operation be available through a button on the window. As a result, the space available for viewing and editing became unusably small. The developer was precluded from moving some operations to menus because the GUI layout had been specified precisely at an early step. (Of course, given too much freedom programmers can develop very bad GUIs. Stakeholder review is essential to avoid such problems.)

2.5 Spiral Invariant 5: Use of Anchor Point Milestones: LCO, LCA, IOC

A major difficulty of the original spiral model was its lack of intermediate milestones to serve as commitment points and progress checkpoints [Forsberg 96]. This difficulty has been remedied by the development of a set of anchor point milestones: Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC) [Boehm 96]. These can be described as stakeholder commitment points in the software life cycle: LCO is the stakeholder's commitment to support architecting; LCA is the stakeholders' commitment to support full life cycle; and IOC is the stakeholders' commitment to support operations.

Summary of Invariant 5

Why invariant

Avoids analysis paralysis, unrealistic expectations, requirements creep, architectural drift, COTS shortfalls and incompatibilities, unsustainable architectures, traumatic cutovers, useless systems

Example: Stud Poker Analogy

Variants

- 5a. Number of spiral cycles or increments between anchor points**
- 5b. Situation-specific merging of anchor point milestones**

Models excluded

Evolutionary or incremental development with no life cycle architecture

The anchor point milestones were defined in a pair of USC Center for Software Engineering Affiliates' workshops, and as such represent joint efforts by both industry and government

participants [Clark 95]. One of the Affiliates, Rational, Inc., had been defining the phases of its Rational Unified Process, and adopted the anchor point milestones as its phase gates.

The first two anchor points are the Life Cycle Objectives (LCO) and Life Cycle Architecture (LCA). At each of these anchor points the key stakeholders review six artifacts: operational concept description, prototyping results, requirements description, architecture description, life cycle plan, and feasibility rationale (see Section 3.1 for details).

The feasibility rationale covers the key pass/fail question: “If I build this product using the specified architecture and processes, will it support the operational concept, realize the prototyping results, satisfy the requirements, and finish within the budgets and schedules in the plan?” If not, the package should be reworked.

The focus of the LCO review is to ensure that at least one architecture choice is viable from a business perspective. The focus of the LCA review is to commit to a single detailed definition of the review artifacts. The project must have either eliminated all significant risks or put in place an acceptable risk-management plan. The LCA milestone is particularly important, as its pass/fail criteria enable stakeholders to hold up projects attempting to proceed into evolutionary or incremental development without a life cycle architecture.

The LCO milestone is the equivalent of getting engaged, and the LCA milestone is the equivalent of getting married. As in life, if you marry your architecture in haste, you and your stakeholders will repent at leisure. The third anchor point milestone, the Initial Operational Capability (IOC), constitutes an even larger commitment: It is the equivalent of having your first child.

Appropriate variants include the number of spiral cycles of development increments between the anchor points. In some cases, anchor point milestones can be merged. In particular, a project deciding to use a mature and appropriately scalable fourth generation language (4GL) or product line framework will have already determined its choice of life cycle architecture by its LCO milestone, enabling the LCO and LCA milestones to be merged.

Further elucidation and discussion of the anchor point milestones is deferred to Section 3.

Spiral Model and Incremental Commitment: Stud Poker Analogy

A valuable aspect of the original application of the spiral model to the TRW Software Productivity System was its ability to support incremental commitment of corporate resources to the exploration, definition, and development of the system, rather than requiring a large outlay of resources to the project before its success prospects were well understood [Boehm 88]. These decisions are codified with the specific guidelines of the LCO and LCA.

Funding a spiral development can thus be likened to the game of stud poker. You can put a couple of chips in the pot and receive two cards, one hidden and one exposed, along with the other players in the game. If your cards don't promise a winning outcome, you can drop out without a great loss. If your two cards are both aces, you will probably bet on your prospects aggressively (although perhaps less so if you can see the other two aces as other players' exposed cards). In any case, you can decide during each round whether it's worth putting more chips in the pot to buy more information about your prospects for a win or whether it's better not to pursue this particular deal, based on the information available.

Stud Poker Analogy

- **Evaluate alternative courses of action**
 - **Fold: save resources for other deals**
 - **Bet: buy at least one more round**
- **Use incomplete information**
 - **Hole cards: competitive situation**
 - **Rest of deck: chance of getting winner**
- **Anticipate future possibilities**
 - **Likelihood that next betting round will clarify outcome**
- **Commit incrementally rather than all at once**
 - **Call only the most recent bet**
 - **Raise an amount of your own choice**

One of the main challenges for organizations such as the Department of Defense (DoD), is to find incremental commitment alternatives to its current Program Objectives Memorandum (POM) process which involves committing to the full funding of a program (putting all of its chips in the pot) based on very incomplete early information.

2.6 Spiral Invariant 6: Emphasis on System and Life Cycle Activities and Artifacts

Spiral Invariant 6 emphasizes that spiral development of software-intensive systems needs to focus not just on software construction aspects, but also on overall system and life cycle concerns. Software developers are particularly apt to fall into the oft-cited trap: "If your best tool is a hammer, the world you see is collection of nails." Writing code may be a developer's forte, but it stands in importance to the project as do nails to a house.

The spiral model's emphasis on using stakeholder objectives to drive system solutions, and on the life cycle anchor point milestones, guides projects to focus on system and life cycle concerns. The model's use of risk considerations to drive solutions makes it possible to tailor

each spiral cycle to whatever mix of software and hardware, choice of capabilities, or degree of productization is appropriate.

Summary of Invariant 6

Why invariant

**Avoids premature suboptimization on
hardware, software, or development considerations**

Example: Order Processing

Variants

- 6a. Relative amount of hardware and software determined in each cycle**
- 6b. Relative amount of capability in each life cycle increment**
- 6c. Degree of productization (alpha, beta, shrink-wrap, etc.)
of each life cycle increment**

Models excluded

**Purely logical object-oriented methods (because they are insensitive to
operational, performance, and cost risks)**

Example: “Order Processing”

A good example is the Scientific American order processing system sketched in Figure 5. The software people looked for the part of the problem with a software solution (their “nail”), pounded it in with their software hammer, and left Scientific American worse off than when they started.

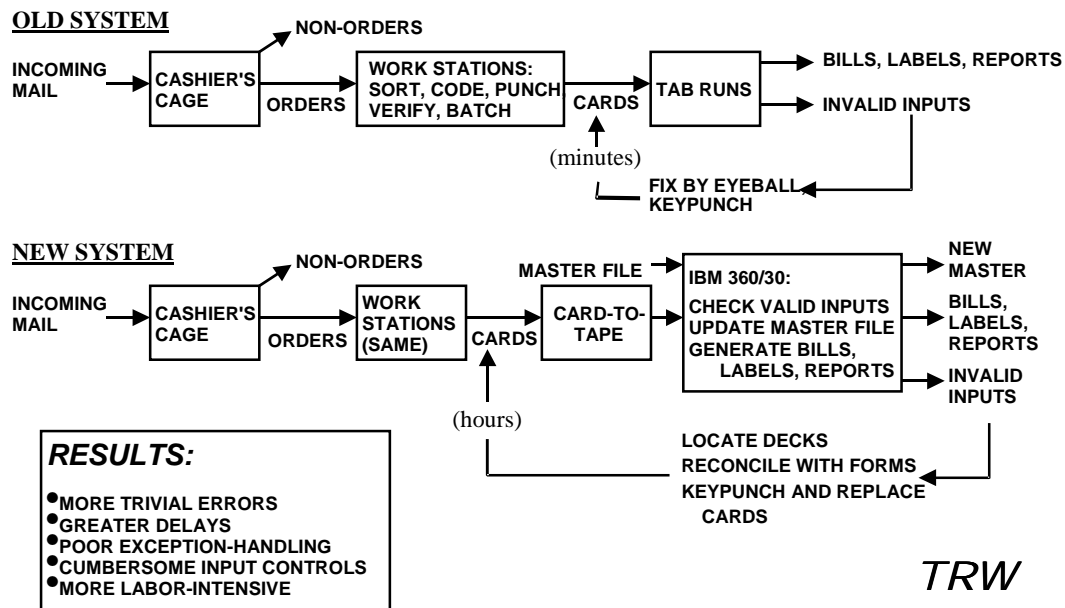


Figure 5: Scientific American Order Processing

Scientific American's objectives were to reduce its subscription processing system's costs, errors, and delays. Rather than analyze the sources of these problems, the software house jumped in and focused on the part of the problem having a software solution. The result was a batch-processing computer system whose long delays put extra strain on the clerical portion of the system that had been the major source of costs, errors, and delays in the first place. As seen in the chart, the business outcome was a new system with more errors, greater delays, higher costs, and less attractive work than its predecessor [Boehm 81].

This kind of outcome would have resulted even if the software automating the tabulator-machine functions had been developed in a risk-driven cyclic approach. However, its Life Cycle Objectives milestone package would have failed its feasibility review, as it had no system-level business case demonstrating that the development of the software would lead to the desired reduction in costs, errors, and delays. Had a thorough business case analysis been done, it would have identified the need to re-engineer the clerical business processes as well as to automate the manual tab runs. Further, as shown by recent methods such as the DMR Benefits Realization Approach, the business case could have been used to monitor the actual realization of the expected benefits, and to apply corrective action to either the business process re-engineering or the software engineering portions of the solution (or both) as appropriate [Thorpe 98].

Hazardous Spiral Look-Alikes: Logic-Only OO Designs

Models excluded by Invariant 6 include most published object-oriented analysis and design (OOA&D) methods, which are usually presented as abstract logical exercises independent of

system performance or economic concerns. For example, in a recent survey of 16 OOA&D books, only six listed the word “performance” in their index, and only two listed “cost.”

3 Anchor Point Milestones

The anchor point milestones from invariant 5 are

- Life Cycle Objectives (LCO)
- Life Cycle Architecture (LCA)
- Initial Operational Capability (IOC)

Since these milestones [Boehm 96] are relatively new additions to the Spiral Development Model, they are covered in some depth in succeeding pages. The next two subsections describe the anchor points themselves and are followed by a discussion of how an “evolutionary” development process can benefit from the LCA milestone. Succeeding sections summarize other aspects of the spiral model relevant to the anchor point milestones, such as their support of incremental commitment and their relation to the Rational Unified Process and the USC MBASE approach.

3.1 Detailed Descriptions

Table 1 lists the major features of the LCO and LCA milestones. Unlike most current software milestones

- Their focus is not on requirements snapshots or architecture point solutions, but on requirements and architectural specifications which anticipate and accommodate system evolution. This is the reason for calling them the “Life Cycle” Objectives and Architecture milestones.
- Elements can be either specifications or executing programs with data (e.g., prototypes, COTS products).
- The Feasibility Rationale is an essential element rather than an optional add-on.
- Stakeholder concurrence on the milestone elements is essential. This establishes mutual stakeholder buy-in to the plans and specifications, and enables a collaborative team approach to unanticipated setbacks rather than an adversarial approach as in most contract models.

These characteristics explain why LCO and LCA are critical to success on projects, and thus why they are able to function successfully as anchor points across many types of software development.

A key feature of the LCO milestone is the need for the Feasibility Rationale to demonstrate a viable business case for the proposed system. Not only should this business case be kept up to date, but also it should be used as a basis for verifying that expected benefits will actually be realized, as discussed in the “Order Processing” example for Invariant 6.

Table 1: WinWin Spiral Anchor Points (with risk-driven level of detail for each element)

Milestone Element	Life Cycle Objectives (LCO)	Life Cycle Architecture (LCA)
Definition of Operational Concept	<p>Top-level system objectives and scope</p> <ul style="list-style-type: none"> - System boundary - Environment parameters and assumptions - Evolution parameters <p>Operational concept</p> <ul style="list-style-type: none"> - Operations and maintenance scenarios and parameters - Organizational life-cycle responsibilities (stakeholders) 	<p>Elaboration of system objectives and scope of increment</p> <p>Elaboration of operational concept by increment</p>
System Prototype(s)	<p>Exercise key usage scenarios</p> <p>Resolve critical risks</p>	<p>Exercise range of usage scenarios</p> <p>Resolve major outstanding risks</p>
Definition of System Requirements	<p>Top-level functions, interfaces, quality attribute levels, including:</p> <ul style="list-style-type: none"> - Growth vectors and priorities - Prototypes <p>Stakeholders' concurrence on essentials</p>	<p>Elaboration of functions, interfaces, quality attributes, and prototypes by increment</p> <ul style="list-style-type: none"> - Identification of TBD's (to-be-determined items) <p>Stakeholders' concurrence on their priority concerns</p>
Definition of System & Software Architecture	<p>Top-level definition of at least one feasible architecture</p> <ul style="list-style-type: none"> - Physical and logical elements and relationships - Choices of COTS and reusable software elements <p>Identification of infeasible architecture options</p>	<p>Choice of architecture and elaboration by increment</p> <ul style="list-style-type: none"> - Physical and logical components, connectors, configurations, constraints - COTS, reuse choices - Domain-architecture and architectural style choices <p>Architecture evolution parameters</p>
Definition of Life-Cycle Plan	<p>Identification of life-cycle stakeholders</p> <ul style="list-style-type: none"> - Users, customers, developers, maintainers, interoperators, general public, others <p>Identification of life-cycle process model</p> <ul style="list-style-type: none"> - Top-level stages, increments <p>Top-level WWWWWHH* by stage</p>	<p>Elaboration of WWWWWHH* for Initial Operational Capability (IOC)</p> <ul style="list-style-type: none"> - Partial elaboration, identification of key TBDs for later increments
Feasibility Rationale	<p>Assurance of consistency among elements above</p> <ul style="list-style-type: none"> - Via analysis, measurement, prototyping, simulation, etc. - Business case analysis for requirements, feasible architectures 	<p>Assurance of consistency among elements above</p> <p>All major risks resolved or covered by risk management plan</p>

*WWWWWHH: Why, What, When, Who, Where, How, How Much

A feature distinguishing the LCA milestone from the LCO milestone is the need to have all of the system's major risks resolved, or at least covered by an element of the system's risk management plan. For large systems, passing the LCA milestone is the point of significant escalation of staff level and resource commitments. Proceeding into this stage with major risks unaddressed has led to disaster for many large projects. Some good guidelines for software risk assessment can be found in [Boehm 89a, Charette 89, Carr 93, and Hall 98].

The Initial Operational capability (IOC) is the first the users will see of a functioning system, so getting things wrong in the IOC can have serious consequences. Greeting users with a new system having ill-matched software, poor site preparation, or poor users preparation has been a frequent source of user alienation and project failure.

The key elements of the IOC milestone are

- *Software preparation*, including both operational and support software with appropriate commentary and documentation; data preparation or conversion; the necessary licenses and rights for COTS and reused software, and appropriate operational readiness testing.
- *Site preparation*, including facilities, equipment, supplies, and COTS vendor support arrangements.
- *User, operator and maintainer preparation*, including selection, teambuilding, training and other qualification for familiarization, usage, operations, or maintenance.

As with the Pre-Ship Testing Example given with Invariant 3, the IOC milestone is risk-driven with respect to the system objectives determined in the LCO and LCA milestones. Thus, for example, these objectives drive the tradeoff between IOC date and quality of the product. These will differ markedly between such systems as the safety-critical Space Shuttle Software and a market-window-critical commercial software product. The difference between these two cases is narrowing as commercial vendors and users increasingly appreciate the market risks involved in buggy products [Cusumano 95].

3.2 Relationships to Other Process Models

This section sketches four process models that have adopted portions of the spiral model or extended the spiral model.

Evolutionary Development

All too often, a project will be started on an evolutionary development approach based on a statement such as, "We're not sure what to build, so let's throw together a prototype and evolve it until the users are satisfied." This approach is insensitive to several risks corresponding to the set of assumptions for a successful evolutionary. These assumptions are:

1. The initial release is sufficiently satisfactory to key system stakeholders that they will continue to participate in its evolution.
2. The architecture of the initial release is scalable to accommodate the full set of system life cycle requirements (e.g., performance, safety, security, distribution, localization).

3. The operational user organizations are sufficiently flexible to adapt to the pace of system evolution
4. The dimensions of system evolution are compatible with the dimensions of evolving-out the legacy systems it is replacing.

Without some initial attention to user needs, as required for LCO and LCA, the prototype may be so far from the users' needs that they consider it a waste of time to continue. As discussed above, it will be risky to proceed without a life cycle architecture to support evolution. Another risk is “information sclerosis”: the propensity for organizations to lock into operational procedures making it difficult to evolve toward better capabilities [Boehm 88]. A final frequent risk is that legacy systems are often too inflexible to adapt to desired directions of evolution. In such cases, a preferable process model is incremental development, with the increments determined by the ease of evolving-out portions of the legacy system.

Rational RUP Phases

Versions of Figure 6 appear in the three main books on the Rational Unified Process (RUP) [Royce 98, Kruchten 98, Jacobson 99]. It shows the relations between LCO, LCA, and IOC milestones and the RUP phases of Inception, Elaboration, Construction, and Transition. It also illustrates that the requirements, design, implementation, and deployment artifacts are incrementally grown throughout the phases. As indicated in Variant 3b, the size of the shaded bars (the relative efforts for Requirements, Design, Implementation, and Deployment) will vary from project to project.

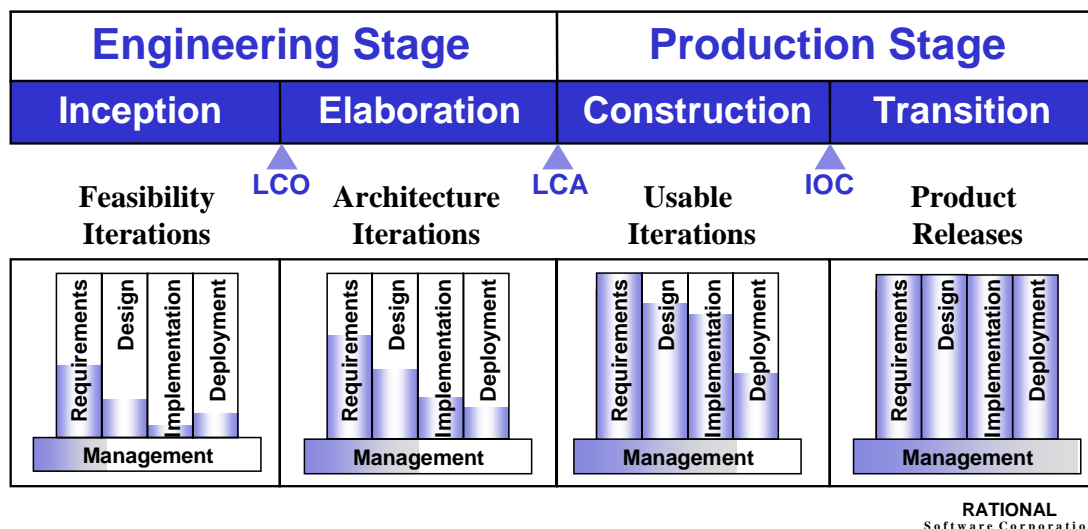


Figure 6: Anchor Points and the Rational RUP Phases

WinWin Spiral Model

The original spiral model [Boehm 88] began each cycle of the spiral by performing the next level of elaboration of the prospective system's objectives, constraints and alternatives. A

primary difficulty in applying the spiral model has been the lack of explicit process guidance in determining these objectives, constraints, and alternatives. The Win-Win Spiral Model (Figure 7) [Boehm 94] uses the Theory W (win-win) approach [Boehm 89b] to converge on a system's next-level objectives, constraints, and alternatives. This Theory W approach involves identifying the system's stakeholders and their win conditions, and using negotiation processes to determine a mutually satisfactory set of objectives, constraints, and alternatives for the stakeholders.

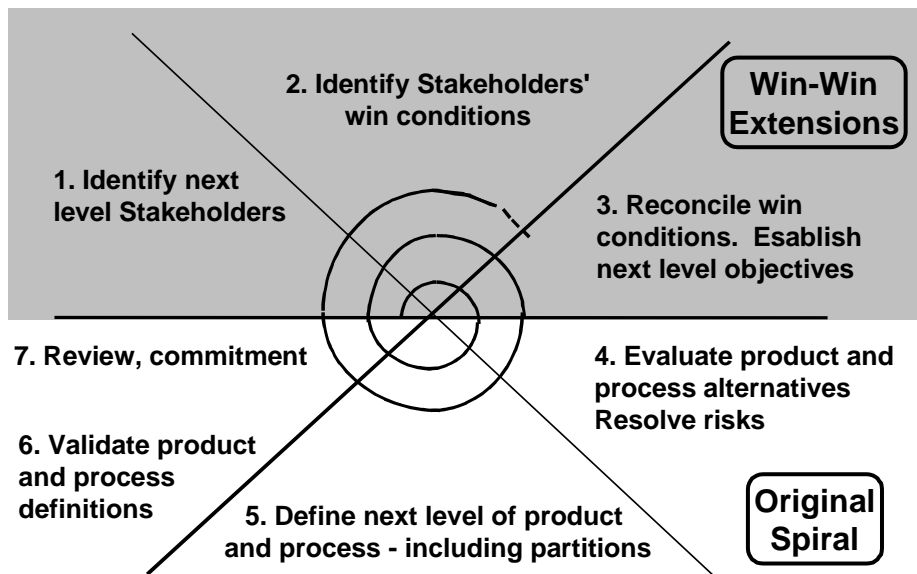


Figure 7: The WinWin Spiral Model

In particular, as illustrated in the figure, the nine-step Theory W process translates into the following spiral model extensions (numbered as in the figure):

1. Determine Objectives. Identify the system life-cycle stakeholders and their win conditions. Establish initial system boundaries and external interfaces.
 2. Determine Constraints. Determine the conditions under which the system would produce win-lose or lose-lose outcomes for some stakeholders.
 3. Identify and Evaluate Alternatives. Solicit suggestions from stakeholders. Evaluate them with respect to stakeholders' win conditions. Synthesize and negotiate candidate win-win alternatives. Analyze, assess, and resolve win-lose or lose-lose risks.
- Commit. Record Commitments, and areas to be left flexible, in the project's design record and life cycle plans.
- 4-7. Cycle Through the Spiral. Elaborate the win conditions, evaluate and screen alternatives, resolve risks, accumulate appropriate commitments, and develop and execute downstream plans.

MBASE Electronic Process Guide

The Model-Based (System) Architecting and Software Engineering (MBASE) approach [Boehm 99a, Boehm 99b, Boehm 00b], provides more detailed definitions of the anchor point milestone elements [Boehm 00b], and a process guide for deriving them.

The MBASE Electronic Process Guide (EPG) [Mehta 99] was developed using the Electronic Process Guide support tool provided by the SEI [Kellner 98]. It uses Microsoft Access to store the process elements, using an Activities-Artifacts-Agents model, and translates the results into hyperlinked HTML for web-based access. Four sorts of windows appear: diagrams, outlines, descriptions, and templates. Figure 8 shows the top-level outline of activities, artifacts, and agents. Figure 9 shows the diagram for the Inception phase of the process. A click on any element of that diagram brings up a structured description of that element. In the Figure, the "Risk Driven Analysis" section was clicked to bring up the outline and description shown in Figure 10.

The top section of the outline window in Figure 11 places Risk Driven Analysis within its context of the Inception Phase, parallel to the Elaboration Phase, with both being part of the MBASE 577a Process (used for Computer Science course 577a at USC). The other activities of the Inception Phase are also listed and can be fetched by clicking their names in the outline. Below the process outline is the outline of the Risk Driven Analysis description shown to the right. Clicking there on the "Outputs" item scrolls the description to its Outputs section, which is shown in Figure 10. Clicking an artifact name like "Operational Concept Description" brings up another structured description; in this case one containing a reference to an active template document for the artifact. Clicking in the fields of this document lets the user enter the values and descriptions appropriate to his or her own particular project.



Figure 8 EPG Top-Level Outline of Activities, Artifacts, and Agents

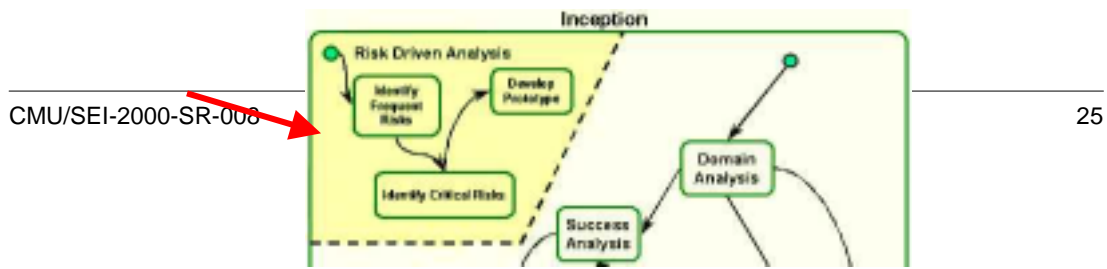


Figure 9 EPG Diagram of the Inception Phase

Activities:

- MBASE 577a Process
 - Inception Phase
 - Risk-driven Analysis**
 - Identify Critical Risk
 - Identify Frequent Risk
 - Develop Prototype
 - Domain Analysis
 - Success Analysis
 - Product Analysis
 - Process Analysis
 - Property Analysis
 - LOO Review and Consideration
 - Elaboration Phase
 - Record Project Effort
- Methods for MBASE

Activity: Risk-driven Analysis

- Overview
- Purpose
- Decomposition
- Description
- Tools and Techniques
- Pitfalls
- Inputs
- Outputs
- Behavior
- Effort Guidelines

Risk-driven Analysis

Overview

System analysis driven by risk

Purpose

The objectives of this activity are:

- To identify the most critical risk factors in the project
- To assess the impact of the risks associated with the project
- To prepare a risk mitigation plan for the most critical risks

Decomposition

The activity Risk-driven Analysis is decomposed into the following:

- Identify Frequent Risks
- Develop Prototype
- Identify Critical Risks

Description

MBASE is a risk-driven process framework and every process based on MBASE adopts an early risk assessment and resolution approach. During Inception critical system risks should be identified and resolved based on their impact on the system life-cycle

Tools and Techniques

Software Risk Taxonomy, Project Simplifiers and Complicators

Pitfalls

The common pitfalls during this activity are:

- Not identifying the major technological risks during Inception

Figure 10 EPG Outline and description of Risk Driven Analysis

Participating Agent	support risk management as performing agents
Performing Agent	Identify and resolve proje

Inputs

The following artifacts are inputs to the Risk-driv

Artifact	Source
System and Software Architecture Description	Pr

Outputs

The following artifacts are outputs to the Risk-d

Artifact	Source
Feasibility Rationale Description	Project
Operational Concept Description	Project

Behavior

Figure 11 The "Outputs" section of the description on the right of Figure 10.

4 Summary

This paper has presented a preliminary definition of the spiral development model and characterized the model further by presenting a set of six “invariant” attributes. That is, six properties which every spiral development process must incorporate. These are listed in Table 2 along with a notion of why they are necessary and a few characteristics, called “variants,” that may vary from one spiral process model to another.

Numerous process models with development structured in a cyclic series of efforts can appear to be spiral models and yet violate one or more invariants and subsequently fail. These are listed in Table 3, together with a reference to further discussion in the text.

The second part of the paper was devoted to the anchor point milestones of Invariant 5. These milestones—Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operating Capability (IOC)—provide concrete artifacts to drive the project toward completion. They also provide for comparison, evaluation, and planning between projects. The discussion concluded with the WinWin spiral model and MBASE, which assist in the use of the anchor point milestones.

This paper is intended as a sufficient characterization of the Spiral Development Model to distinguish it from other project process models. Although not within itself a full description and user's guide for the spiral development model, it is a suitable basis for such further works.

Table 2: *Invariants of Spiral Processes: Name, Rationale, and Variants*

Invariant {& example}	Why invariant	Variants
1. Concurrent rather than sequential determination of key artifacts--Ops Concept, Requirements, Plans, Design, Code--in each spiral cycle {One- Second Response Time}	Avoids premature sequential commitments	1a. Relative amount of each artifact developed in each cycle 1b. Number of concurrent mini-cycles in each cycle
2. Each cycle considers critical stakeholder objectives and constraints, product and process alternatives, risk identification and resolution, stakeholder review, and commitment to proceed {Windows-only COTS}	Avoids commitment to alternatives that are risky or unacceptable to stakeholders Avoids wasting effort on unusable alternatives	2a. Choice of risk resolution techniques: prototyping, simulation, modeling, benchmarking, reference checking, etc. 2b. Level of effort on each activity within each cycle
3. Level of effort on each activity within each cycle driven by risk considerations {Pre-ship Testing}	Avoids too little or too much of each activity Avoids overkill or belated risk resolution	3a. Choice of methods used to pursue activities: MBASE/WinWin, Rational RUP, JAD, QFD, ESP, . . . 3b. Degree of detail of artifacts produced in each cycle
4. Degree of detail of artifacts produced in each cycle driven by risk considerations {GUI layouts}	Avoids too little or too much of each artifact Avoids overkill or belated risk resolution	4a. Choice of artifact representations (SA/SD, UML, MBASE, formal specs, programming languages, etc.)
5. Managing stakeholder life cycle commitments via the LCO, LCA, and IOC anchor point milestones {Stud Poker Analogy}	Avoids analysis paralysis, unrealistic expectations, requirements creep, architectural drift, COTS shortfalls or incompatibilities, unsustainable architectures, traumatic cutovers, useless systems	5a. Number of spiral cycles or increments between anchor points 5b. Situation-specific merging of anchor point milestones
6. Emphasis on system and life cycle activities and artifacts rather than software and initial development activities and artifacts {Order Processing}	Avoids premature suboptimization on hardware, software, or development considerations	6a. Relative amount of hardware and software determined in each cycle 6b. Relative amount of capability in each life cycle increment 6c. Degree of productization (alpha, beta, shrink-wrap, etc.) of each life cycle increment

Table 3: Hazardous Spiral Look-Alikes

Hazardous Spiral Look-Alike	Invariant	Examples
Incremental sequential waterfalls with significant COTS, user interface, or technology risks	1	One Second Response Time Violation of Waterfall Assumptions
Sequential spiral phases with key stakeholders excluded from phases	2	Windows-only COTS Excluding Key Stakeholders
Risk-insensitive evolutionary or incremental development	3	Pre-ship Testing Risk Insensitivity Section 3.3: Risks of Evolutionary Development
Impeccable spiral plan with no commitment to managing risks	3	(special case of the above)
Insistence on complete specifications for COTS, user interface, or deferred-decision situations	4	Risk of Precise Specification
Evolutionary development with no life-cycle architecture	5	Section 3.3: Risks of Evolutionary Development
Purely logical object-oriented methods with operational, performance, or cost risks	6	Logic-Only OO Designs

References

- [AT&T 93] AT&T, *Best Current Practices: Software Architecture Validation*. Murray Hill, NJ: AT&T, 1993.
- [Bernstein 00] Bernstein, L. "Automation of Provisioning," *Proceedings, USC-SEI Spiral Experience Workshop*. Los Angeles, CA, Feb. 2000.
<http://www.sei.cmu.edu/cbs/spiral2000/Bernstein>
- [Boehm 81] Boehm, B. *Software Engineering Economics*. New York, NY: Prentice Hall, 1981.
- [Boehm 88] Boehm, B. "A Spiral Model of Software Development and Enhancement." *Computer* (May 1988): 61-72.
- [Boehm 89a] Boehm, B. *Software Risk Management*. IEEE Computer Society Press, 1989.
- [Boehm 89b] Boehm, B. & Ross, R. "Theory W Software Project Management: Principles and Examples." *IEEE Trans. Software Engr.* (Jul. 1989).
- [Boehm 94] Boehm, B. & Bose, P. "A Collaborative Spiral Software Process Model Based on Theory W." *Proceedings, ICSP 3*, IEEE, Reston, VA, Oct. 1994
- [Boehm 96] Boehm, B. "Anchoring the Software Process." *IEEE Software* 13, 4 (July 1996): 73-82
- [Boehm 97] Boehm, B. "Developing Multimedia Applications with the WinWin Spiral Model." *Proceedings, ESEC/FSE 97*. New York, NY: Springer Verlag, 1997.
- [Boehm 98] Boehm, B. "Using the Win Win Spiral Model: A Case Study." *IEEE Computer* (July 1998): 33-44.
- [Boehm 99a] Boehm, B. & Port, D. "Escaping the Software Tar Pit: Model Clashes and How to Avoid Them." *ACM Software Engineering Notes* (Jan. 1999): 36-48.
- [Boehm 99b] Boehm, B. & Port, D. "When Models Collide: Lessons from Software Systems Analysis." *IEEE IT Professional* (Jan./Feb. 1999): 49-56.
- [Boehm 00a] Boehm, B. "Unifying Software Engineering and Systems Engineering." *IEEE Computer* (March 2000): 114-116.

- [Boehm 00b]** Boehm, B; Abi-Antoun, M; Brown, A.W; Mehta, N. & Port, D. "Guidelines for the LCO and LCA Deliverables for MBASE." *USC-CSE*, Mar. 2000. http://sunset.usc.edu/classes/cs577b_2000/EP/07/MBASE_Guidelines_for_CS577v0_2.pdf
- [Bostelaar 00]** Bostelaar, T. "TRW Spiral Development Experience on Command & Control Product Lines Program," *Proceedings, USC-SEI Spiral Experience Workshop*. Los Angeles, CA., Feb. 2000. <http://www.sei.cmu.edu/cbs/spiral2000/Bostelaar>
- [Carr 93]** Carr, M.; Kondra, S.; Monarch, I.; Ulrich, F. & Walker, C. *Taxonomy-Based Risk Identification* (CMU/SEI-93-TR-06 ADA 266992) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1993 <http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.006.html>.
- [Charette 89]** Charette, R.N. *Software Engineering Risk Analysis and Management*. New York, NY: McGraw Hill, 1989.
- [Clark 95]** Clark, B. & Boehm, B. (eds.) "Knowledge Summary: Focused Workshop on COCOMO 2.0," *USC-CSE*, May 16-18, 1995.
- [Cusumano 95]** Cusumano, M. & Selby, R. *Microsoft Secrets*. New York, NY: Free Press, 1995
- [DeMillo 00]** DeMillo, R. "Continual Improvement: Spiral Software Development," *Proceedings, USC-SEI Spiral Experience Workshop*. Los Angeles, CA, Feb. 2000. <http://www.sei.cmu.edu/cbs/spiral2000/DeMillo>
- [Forsberg 96]** Forsberg, K; Mooz, H. & Cotterman, H. *Visualizing Project Management*. New York, NY: Wiley Publishers, 1996.
- [Garlan 95]** Garlan, D.; Allen, R. & Ockerbloom, J. "Architectural Mismatch: Why Reuse Is So Hard," *IEEE Software* (Nov 1995): 17-26.
- [Hall 98]** Hall, E. *Managing Risk: Methods for Software Systems Development*. Reading, MA: Addison Wesley, 1998.
- [Hantos 00]** Hantos, P. "From Spiral to Anchored Processes: A Wild Ride in Lifecycle Architecting," *Proceedings, USC-SEI Spiral Experience Workshop*. Los Angeles, CA, Feb. 2000. <http://www.sei.cmu.edu/cbs/spiral2000/Hantos>
- [Jacobson 99]** Jacobson, I; Booch, G. & Rumbaugh, J. *The Unified Software Development Process*. Reading, MA: Addison-Wesley, 1999.

- [Kellner 98]** “Process Guides: Effective Guidance for Process Participants,” *Proceedings of the 5th International Conference on the Software Process: Computer Supported Organizational Work*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1998.
- [Kitaoka 00]** Kitaoka, B. “Yesterday, Today & Tomorrow: Implementations of the Development Lifecycles,” *Proceedings, USC-SEI Spiral Experience Workshop*. Los Angeles, CA, Feb 2000.
<http://www.sei.cmu.edu/cbs/spiral2000/Kitaoka>
- [Kruchten 98]** Kruchten, P. *The Rational Unified Process*. Reading, MA: Addison-Wesley, 1998.
- [Leinbach 00]** Leinbach, C. “E-Business and Spiral Development,” *Proceedings, USC-SEI Spiral Experience Workshop*. Los Angeles, CA, Feb 2000.
<http://www.sei.cmu.edu/cbs/spiral2000/Leinbach>
- [Mehta 99]** Mehta, N. *MBASE Electronic Process Guide*. USC-CSE, Los Angeles, CA: Oct 1999. <http://sunset.usc.edu/research/MBASE/EPG>
- [Royce 98]** Royce, W. *Software Project Management: A Unified Framework*. Reading, MA: Addison Wesley, 1998.
- [SPC 94]** Software Productivity Consortium, *Process Engineering with the Evolutionary Spiral Process Model* (SPC-93098-CMC, Version 01.00.06). Herndon, Virginia, 1994
- [Thorp 98]** Thorp, J. *The Information Paradox*. New York, NY: McGraw Hill, 1998.
- [USAF 00]** U.S. Air Force. “Evolutionary Acquisition for C2 Systems.” *Air Force Instruction* (January 1, 2000) 63-123.

Acronyms

AC2ISRC	Aerospace Command and Control, Intelligence, Surveillance, and Reconnaissance Command (Air Force)
AFOTEC	Air Force Operational Test and Evaluation Center
ASC	Aeronautical Systems Center
C2ISR	Command and Control, Intelligence, Surveillance, and Reconnaissance
CCPDS-R	Command center processing and display system replacement
CECOM	US Army Communications-Electronics Command
CIO	Chief Information Officer
CMU	Carnegie Mellon University, home of SEI
COTS	Commercial-Off-The-Shelf
CSE	Center for Software Engineering, USC
DAU	Defense Acquisition University
DCMC	Defense Contract Management Command
DoD	Department of Defense
DSMC	Defense Systems Management College
ESC	Electronic Systems Command (Air Force)
ESP	Evolutionary Spiral Process (SPC)
FAA	Federal Aviation Agency
FFRDC	Federally Funded Research and Development Center
GUI	Graphical User Interface
IKIWISI	I'll know it when I see it
INCOSE	International Council on Systems Engineering Air Force Operational Test and Evaluation Center
IOC	Initial Operating Capability
IPT	Integrated Product Team
JAD	Joint Applications Development
LCA	Life Cycle Architecture
LCO	Life Cycle Objectives
MBASE	Model-Based Architecting and Software Engineering (CSE)
MITRE	MITRE (an FFRDC)

OO	Object-Oriented
OOA&D	Object-Oriented Analysis and Design
OSD	Office of the Secretary of Defense
OUSD/AR	Office of the Under Secretary of Defense / Acquisition Reform
PMI	Program Management Institute
POM	Program Objectives Memorandum
QFD	Quality Function Deployment
ROI	Return on investment
RUP	Rational Unified Process (Rational)
SAF/AQ	Secretary of the Air Force/Acquisition
SAIC	Science Applications International Corporation
SA/SD	Structured Analysis/Structured Design
SDM	Spiral Development Model
SEI	Software Engineering Institute, CMU
SMC	Air Force Space and Missile Systems Center
SPAWAR	Space and Naval Warfare Systems Command
SPC	Software Productivity Consortium
TBD	To Be Determined
UML	Unified Modeling Language
USC	University of Southern California, home of CSE
USD/AT&L	Under Secretary of Defense/Acquisition, Technology, and Logistics
WWWWWHH	Why, What, When, Who, Where, How, How much

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (LEAVE BLANK)	2. REPORT DATE July 2000	3. REPORT TYPE AND DATES COVERED Final		
4. TITLE AND SUBTITLE Spiral Development: Experience, Principles and Refinements		5. FUNDING NUMBERS C — F19628-95-C-0003		
6. AUTHOR(S) Barry Boehm, Wilfred J Hansen, editor				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2000-SR-008	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12.A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS			12.B DISTRIBUTION CODE	
13. abstract (maximum 200 words) Spiral development is a family of software development processes characterized by repeatedly iterating a set of elemental development processes and managing risk so it is actively being reduced. This paper characterizes spiral development by enumerating a few "invariant" properties that any such process must exhibit. For each, a set of "variants" is also presented, demonstrating a range of process definitions in the spiral development family. Each invariant excludes one or more "hazardous spiral look-alike" models, which are also outlined. This report also shows how the spiral model can be used for a more cost-effective incremental commitment of funds, via an analogy of the spiral model to stud poker. An important and relatively recent innovation to the spiral model has been the introduction of anchor point milestones. The latter part of the paper describes and discusses these.				
14. SUBJECT TERMS Spiral development model, invariant, variant, anchor point milestones, software development, project management, risk management, cyclic phases, iterative process			15. NUMBER OF PAGES 36	
16. PRICE CODE				
7. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102