Spiral Development: Experience, Principles, and Refinements

Barry Boehm, USC
LA SPIN Presentation
June 28, 2000

boehm@sunlight.usc.edu
http://sunlight.usc.edu/MBASE
Spiral Model and MBASE

- Spiral experience
  - Critical success factors
  - Invariants and variants
  - Stud poker analogy
- Spiral refinements
  - WinWin spiral
  - Life cycle anchor points
- MBASE
- Recent Developments
“Spiral Development Model:”
Candidate Definition

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. The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.
# Spiral Invariants and Variants - 1

- **Critical success factor examples**

<table>
<thead>
<tr>
<th>Invariants</th>
<th>Why Invariant</th>
<th>Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Concurrent rather than sequential determination of artifacts (OCD, Rqts, Design, Code, Plans) in each spiral cycle.</td>
<td>• Avoids premature sequential commitments to Rqts, Design, COTS, combination of cost/schedule performance - 1 sec. response time</td>
<td>1a. Relative amount of each artifact developed in each cycle.</td>
</tr>
<tr>
<td>2. Consideration in each cycle of critical-stakeholder objectives and constraints, product and process alternatives, risk identification and resolution, stakeholder review and commitment to proceed.</td>
<td>• Avoids commitment to stakeholder-unacceptable or overly risky alternatives.</td>
<td>2a. Choice of risk resolution techniques: prototyping, simulation, modeling, benchmarking, reference checking, etc.</td>
</tr>
<tr>
<td>3. Level of effort on each activity within each cycle driven by risk considerations.</td>
<td>• Avoids wasted effort in elaborating unsatisfactory alternatives.</td>
<td>2b. Level of effort on each activity within each cycle.</td>
</tr>
<tr>
<td></td>
<td>• Determines “how much is enough” of each activity: domain engr., prototyping, testing, CM, etc.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• - Pre-ship testing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Avoids overkill or belated risk resolution.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3a. Choice of methods used to pursue activities: MBASE/WinWin, Rational USDP, JAD, QFD, ESP, ...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3b. Degree of detail of artifacts produced in each cycle.</td>
<td></td>
</tr>
</tbody>
</table>
Spiral Invariant 1: Concurrent Determination of Key Artifacts (Ops Concept, Rqts, Design, Code, Plans)

• Why invariant
  – Avoids premature sequential commitments to system requirements, design, COTS, combination of cost/schedule/performance
    – 1 sec 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
Sequential Engineering Buries Risk

Arch. A: Custom many cache processors

Arch. B: Modified Client-Server

Response Time (sec)

Original Spec  After Prototyping
Waterfall Model Assumptions

1. The requirements are knowable in advance of implementation.
2. The requirements have no unresolved, high-risk implications
   - e.g., risks due to COTS choices, cost, schedule, performance, safety, security, user interfaces, organizational impacts
3. The nature of the requirements will not change very much
   - During development; during evolution
4. The requirements are compatible with all the key system stakeholders’ expectations
   - e.g., 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.
Spiral Invariant 2: Each cycle does objectives, constraints, alternatives, risks, review, commitment to proceed

• Why invariant
  – Avoids commitment to stakeholder-unacceptable or overly risky alternatives.
  – Avoids wasted effort in elaborating unsatisfactory alternatives.
    – 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
Windows-Only COTS Example: Digital Library Artifact Viewer

• Great prototype using ER Mapper
  – Tremendous resolution
  – Incremental-resolution artifact display
  – Powerful zoom and navigation features
• Only runs well on Windows
  – Mac, Unix user communities forced to wait
  – Overoptimistic assumptions on length of wait
• Eventual decision to drop ER Mapper
Models Excluded: Sequential Phases Without Key Stakeholders

- High risk of win-lose even with spiral phases
  - Win-lose evolves into lose-lose
- Key criteria for IPT members (AFI 63-123)
  - Representative, empowered, knowledgeable, collaborative, committed
Spiral Invariant 3: Level of Effort Driven by Risk Considerations

• Why invariant
  – Determines ‘how much is enough” of each activity: domain engr., prototyping, testing, CM, etc.
    – Pre-ship testing
    – Avoids overkill or belated risk resolution.

• 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
Pre-Ship Test Risk Exposure

Risk Exposure
RE = Size (Loss) \times \text{Prob (Loss)}

RE (total)
RE (Market share losses)
RE (defect losses)

Amount of testing; Time to market
### Spiral Invariants and Variants - 2

<table>
<thead>
<tr>
<th>Invariants</th>
<th>Why Invariant</th>
<th>Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Degree of detail of artifacts produced in each cycle driven by risk considerations.</td>
<td>• Determines “how much is enough” of each artifact (OCD, Rqts, Design, Code, Plans) in each cycle. • Avoids overkill or belated risk resolution</td>
<td>4a. Choice of artifact representations (SA/SD, UML, MBASE, formal specs, programming languages, etc.)</td>
</tr>
<tr>
<td>5. Managing stakeholder life-cycle commitments via the LCO, LCA, and IOC Anchor Point milestones (getting engaged, getting married, having your first child),</td>
<td>• Avoids analysis paralysis, unrealistic expectations, requirements creep, architectural drift, COTS shortfalls and incompatibilities, unsustainable architectures, traumatic cutovers, useless systems.</td>
<td>5a. Number of spiral cycles or increments between anchor points. 5b. Situation-specific merging of anchor point milestones.</td>
</tr>
<tr>
<td>6. Emphasis on system and life cycle activities and artifacts rather than software and initial development activities and artifacts.</td>
<td>• Avoids premature suboptimization on hardware, software, or initial development considerations.</td>
<td>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.</td>
</tr>
</tbody>
</table>
Spiral Invariant 4:
Degree of Detail Driven by Risk Considerations

• Why invariant
  – Determines “how much is enough” of each artifact (OCD, Rqts, Design, Code, Plans) in each cycle.
    • Screen layout rqts.
  – Avoids overkill or belated risk resolution.

• Variants
  – 4a. 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
Risk-Driven Specifications

• If it’s risky **not** to specify precisely, **Do**
  – Hardware-software interface
  – Prime-subcontractor interface

• If it’s risky **to** specify precisely, **Don’t**
  – GUI layout
  – COTS behavior
Spiral Invariant 5: Use of LCO, LCA, IOC, Anchor Point Milestones

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

• Variants
  5a. Number of spiral cycles or increments between anchor points.
  5b. Situation-specific merging of anchor point milestones
    • Can merge LCO and LCA when adopting an architecture from mature 4GL, product line

• Models excluded
  – Evolutionary or incremental development with no life cycle architecture
Life Cycle Anchor Points

• Common System/Software stakeholder commitment points
  – Defined in concert with Government, industry affiliates
  – Coordinated with the Rational Unified Process

• Life Cycle Objectives (LCO)
  – Stakeholders’ commitment to support architecting
  – Like getting engaged

• Life Cycle Architecture (LCA)
  – Stakeholders’ commitment to support full life cycle
  – Like getting married

• Initial Operational Capability (IOC)
  – Stakeholders’ commitment to support operations
  – Like having first child
# Win Win Spiral Anchor Points

(Risk-driven level of detail for each element)

<table>
<thead>
<tr>
<th>Milestone Element</th>
<th>Life Cycle Objectives (LCO)</th>
<th>Life Cycle Architecture (LCA)</th>
</tr>
</thead>
</table>
| **Definition of Operational Concept** | • Top-level system objectives and scope  
- System boundary  
- Environment parameters and assumptions  
- Evolution parameters  
- Operational concept  
- Operations and maintenance scenarios and parameters  
- Organizational life-cycle responsibilities (stakeholders) | • Elaboration of system objectives and scope of increment  
• Elaboration of operational concept by increment |
| **System Prototype(s)** | • Exercise key usage scenarios  
• Resolve critical risks | • Exercise range of usage scenarios  
• Resolve major outstanding risks |
| **Definition of System Requirements** | • Top-level functions, interfaces, quality attribute levels, including:  
- Growth vectors and priorities  
- Prototypes  
- Stakeholders’ concurrence on essentials | • Elaboration of functions, interfaces, quality attributes, and prototypes by increment  
- Identification of TBD’s (to-be-determined items)  
- Stakeholders’ concurrence on their priority concerns |
| **Definition of System and Software Architecture** | • Top-level definition of at least one feasible architecture  
- Physical and logical elements and relationships  
- Choices of COTS and reusable software elements  
- Identification of infeasible architecture options | • Choice of architecture and elaboration by increment  
- Physical and logical components, connectors, configurations, constraints  
- COTS, reuse choices  
- Domain-architecture and architectural style choices  
- Architecture evolution parameters |
| **Definition of Life-Cycle Plan** | • Identification of life-cycle stakeholders  
- Users, customers, developers, maintainers, interoperators, general public, others  
- Identification of life-cycle process model  
- Top-level stages, increments  
- Top-level WWWWWhH* by stage | • Elaboration of WWWWWhH* for Initial Operational Capability (IOC)  
- Partial elaboration, identification of key TBD’s for later increments |
| **Feasibility Rationale** | • Assurance of consistency among elements above  
- via analysis, measurement, prototyping, simulation,  
- Business case analysis for requirements, feasible architectures | • Assurance of consistency among elements above  
- etc. All major risks resolved or covered by risk management practices |

Initial Operational Capability (IOC)

- **Software preparation**
  - Operational and support software
  - Data preparation, COTS licenses
  - Operational readiness testing

- **Site preparation**
  - Facilities, equipment, supplies, vendor support

- **User, operator, and maintainer preparation**
  - Selection, teambuilding, training
Evolutionary Development Assumptions

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.
Spiral Model and Incremental Commitment: Stud Poker Analogy

- Evaluate alternative courses of action
  - Fold: save resources for other deals
  - Ante: buy at least one more round

- Using incomplete information
  - Hole cards: competitive situation
  - Rest of deck: chance of getting winner

- Anticipating future possibilities
  - Likelihood that next round will clarify outcome

- Commit incrementally rather than all at once
  - Challenge: DoD POM process makes this hard to do
Anchor Points and Rational RUP Phases

Engineering Stage

Manufacturing Stage

Inception  Elaboration  Construction  Transition

Feasibility Iterations  Architecture Iterations  Usable Iterations  Product Releases
Spiral Model Refinements

• Where do objectives, constraints, alternatives come from?
  – Win Win extensions

• Lack of intermediate milestones
  – Anchor Points: LCO, LCA, IOC
  – Concurrent-engineering spirals between anchor points

• Need to avoid model clashes, provide more specific guidance
  – MBASE
Spiral Invariant 6:
Emphasis on System and Life Cycle Activities and Artifacts

• Why invariant
  – Avoids premature suboptimization on hardware, software, or development considerations.
    • Scientific American

• 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
    • Insensitive to operational, performance, cost risks
Problems With Programming-Oriented Top-Down Development

“SCIENTIFIC AMERICAN” SUBSCRIPTION PROCESSING

Results:

- MORE TRIVIAL ERRORS
- GREATER DELAYS
- POOR EXCEPTION-HANDLING
- CUMBERSOME INPUT CONTROLS
- MORE LABOR-INTENSIVE
Summary: Hazardous Spiral Look-Alikes

- Incremental sequential waterfalls with significant COTS, user interface, or technology risks
- Sequential spiral phases with key stakeholders excluded from phases
- Risk-insensitive evolutionary or incremental development
- Evolutionary development with no life-cycle architecture
- Insistence on complete specs for COTS, user interface, or deferred-decision situations
- Purely logical object-oriented methods with operational, performance, or cost risks
- Impeccable spiral plan with no commitment to managing risks
Summary: Successful Spiral Examples

- Rapid commercial: C-Bridge’s RAPID process
- Large commercial: AT&T/Lucent/Telcordia spiral extensions
- Commercial hardware-software: Xerox Time-to-Market process
- Large aerospace: TRW CCPDS-R
- Variety of projects: Rational Unified Process, SPC Evolutionary Spiral Process, USC MBASE approach
Recent Developments

• Workshop contents available on SEI web site
  – SEI Tech Reports forthcoming

• Followon Workshop September 13-15, 2000
  – DUSD/S&T sponsor; n Washington DC
  – SEI lead organizer, in collaboration with USC

• New DoDD 5000-series acquisition directives due July-Aug
  – Emphasize evolutionary acquisition; spiral

• ASD/C3I Panel report on evolutionary acquisition
  – recommends use of spiral development
Hands-on Tutorials at USC, July 25-27

• **Easy WinWin: July 25-26**
  – Objective: Learn to initiate and conduct Easy WinWin requirements negotiations in your organization
  – Mode: Hands-on at individual workstation
  – Registration: Free to USC-CSE Affiliates; $500. otherwise

• **MBASE* Electronic Process Guide: July 27**
  – Objective: Learn to use MBASE and its EPG
  – Mode: Lecture in morning; hands-on in afternoon
  – Registration: Free to USC-CSE Affiliates: $250. otherwise

• Details at [http://sunset.usc.edu/upcoming-events](http://sunset.usc.edu/upcoming-events)

*MBASE: Model-Based (System) Architecting and Software Engineering
References

(MBASE material available at http://sunset.usc.edu/MBASE)


