A Process Programmer Looks at the Spiral Model

A Tribute to the Deep Insights of Prof. Barry W. Boehm

Leon J. Osterweil
University of Massachusetts Amherst

Institute of Software Chinese Academy of Sciences
Beijing, China
Barry the Culinary Adventurer
Barry Demolishes the Berlin Wall (1989)
Software Development Lifecycles

- How do we build software?
- How should we build software?
- Can we use these understandings to help us to do better?
- How to proceed to find out?

These are major practical and intellectual challenges
A Brief History of Lifecycle Models;
An observational phenomenological approach

- Life before the Spiral Model
- The Spiral Model
- What the Spiral Model Tells Us
  - And suggests for the future
In the Beginning there was The Waterfall Model

Then the Waterfall with some looping

- Feedback loops confined to successive stages
- “Build it twice”
  - Early prototyping

Diagram:
- Feasibility
- Requirements Specification
- Architecture
- Preliminary Design
- Detailed Design
- Code & Unit Test
- Integration Testing
- System Testing
Others emphasized the Importance of Evolutionary prototyping.
And there were advocates of “Throwaway” prototyping.
And Reuse Based Development Approaches

- Japanese “Software Factory” approaches
These Were All Unsatisfying
The Waterfall Leaves Questions

- Where does output go?
- What to do when reviews fail?
- What causes this rework?
- What portion of activity should be done?
- How do we break this cycle?
And Don’t We Need these Dataflow Edges too?
Other Lifecycles Leave Questions Too

- Which lifecycle to choose when?
- What are we trying to do with those prototypes?
- What is the nature of their iterations?
  - Some don’t show any iterations at all
- What is really going on with them?
Phenomenologically what do we learn from all of this?

- Iteration is a central issue in all of them
  - How
  - What
  - When

- Lack of clear focus on final product

Spiral Model focuses on these issues (and many more)
The Original Spiral Model
(thanks to Rick Adrion for this diagram)

determine, elaborate objectives

plan next phase

Review

The concepts of operations begin.

proto I
LC plan
test plan
integration plan
validation reqmts
design reqmts
acceptance test
development & verification

proto II
Concepts of operations
CW reqmts
design models
benchmarks
design

code

proto III
risk analysis
models

proto IV
risk analysis
design

risk analysis
cost (cumulative)

risk analysis
models
Many revisions, updatings, interpretations, and inspirations have followed. Both from Barry and from others.
Abstract Spiral Model: Size, Knowledge, Understanding (and cost) all grow with time
The Deep Insights

- Software Development is inherently ITERATIVE
  - And--Iterations build on each other
- Iterations build knowledge and artifacts:
  - Software Product grows incrementally
    - In size
    - In knowledge obtained
- But the main driver is Risk Reduction
It is a More Perceptive and Useful Updating of the Shewhart/Deming Cycle
It is a More Perceptive and Useful Updating of the Shewhart/Deming Cycle

Lack of focus on artifacts
Things That Could be Made Clearer

- Clearer focus on Product
- The role of context and history
- The Loop Invariant

An Articulate Process Language Can Help
Macroprocess/Microprocess Dichotomy
Process as Object

Input Artifacts

Resources:
- People
- Money
- Tools
- Time

Process

Outputs
- Artifacts
- Effects on the world

Other Behaviors
- Money used
- Time spent
- Errors committed
Macro-Process Focus

Resources: People, Money, Tools, Time

Input Artifacts

Process

Outputs
Artifacts
Effects on the world

Other Behaviors
Money used
Time spent
Errors committed

Common approaches: CMMI, ISO 9000, Six Sigma
Micro-Process Focus

Resources: People, Money, Tools, Time

Input Artifacts

Process

Outputs Artifacts
Effects on the world

Other Behaviors
Money used
Time spent
Errors committed

Needed approach: Define, analyze
Automate, precise process definitions
Spiral Model is a “Quasi-Micro-Process” View

Resources: People, Money, Tools, Time

Input Artifacts

Outputs
Artifacts
Effects on the world

Process

Other Behaviors
Money used
Time spent
Errors committed
Using Little-JIL Language Features to Discover More

- Scoping and Abstraction
  - To support history and context
- Recursion
  - More accurate than iteration
- Artifact specification
  - Focus on product evolution

Supports Clarity about the nature of Iteration and rework in SW Development
Requirements Step with Rework
Design Step with Rework
Clearer Depiction of Rework
Clearer Depiction of Rework

Invocation of step originally defined as substep of Requirements
Clearer Depiction of Rework

Invocation of step originally defined as substep of Requirements

Same exception thrown

Define new rqmt concept

Add new requirements elements

Complicated Fix Process

Declare and Define Rqmt

Inter-rqmts consistency check

rqmtReviewFailed
Clearer Depiction of Rework

- Define new reqmt concept
- Declare and Define Rqmt
- Add new requirements elements
- Inter-reqmts consistency check
- rqmtReviewFailed
- Complicated Fix Process

Invocation of step originally defined as substep of Requirements

Same exception thrown

Different invocation context -> different response
Using Little-JIL to define a Spiral Model Iteration
Becomes the Key to Understanding how “Rework” REALLY works in SW Development.
A Look at Some of the Details
Waterfall Process with Rework in Little-JIL
Waterfall Process with Rework in Little-JIL

The Traditional Waterfall Diagram
Waterfall Process with Rework in Little-JIL

The Traditional Waterfall Diagram

With Rework Details Provided
Waterfall Process with Rework in Little-JIL

The Traditional Waterfall Diagram

With Rework Details Provided

Loop Invariant
Where are the invariants in all of these iterations?
Which is where Boehm’s Incremental Commitment Model Fits
Spiral Model Also Elucidates Scrum

- Daily Sprint as Spiral Iteration (?)
- All four phases are clearly visible
- Reuse of history is clearly there too
  - Although Scrum advocates deemphasize this
- Artifact management details clarify intent of Scrum approach
Scrum Process in Little-JIL
The Scrum Process as a Spiral

- Develop Software
  - Iteration
    - Evaluate Alternatives, Identify and Resolve Risks
      - State
    - Develop and Verify Next Level Product
      - State
    - Plan Next Phases
      - State
  - State

- Sprint Planning Meeting
  - State

- Sprint
  - State

- Sprint Review
  - State

- Sprint Retrospective
Probably Should Be
A Scrum Sprint
Some Closing Observations

- The Spiral Model elucidates some deep truths about software development
- Created a transformation in thinking
- Proper focus on Iteration, Artifacts, Risk
- There is much more to be gleaned from studying the Spiral
- Complementary process notations can help
Thank you Barry!