Spiral Development Experience Report

Rami Razouk
The Aerospace Corporation

Outline

- Program Descriptions (general)
- Spiral Development Approach(es)
- Experiences
- Impediments
Key Drivers

- Portions of the software are “mission-critical”
- High performance requirements
- Some parts of the domain are well understood (but not necessarily by the development organization)
- Some unprecedented mission-unique algorithms
- Rigid schedules + Desire to control costs
I Risk assessment done at the beginning of each spiral increment.

Proposed several (1-3) binary gates per increment.

- 1 MSLOC of C/C++ code, organized into four increments (100K, 250K, 400K, 250K)
- Software requirements allocation done up front (prior to increment 1)
- Each increment of functionality corresponds to a spiral. 5 month - 1 year increments. Some overlap.
**Spiral Development B**

- Prototype mission data processing system
- 180 KSLOC of Ada - 6 increments over 9 months
- Increments 1, 2 & 3 focused on SW architecture and infrastructure
- Increment 6 initially planned with 0 SLOC

**Spiral Development C**

- ~ 1M SLOC C, C++, FORTRAN
- 6 increments, multiple spirals for each increment
- Increment duration is 6 - 8 months. Spirals probably on the order of 2 - 3 months.
- Risk assessment to be done as part of spirals
Spiral Developments D and E

- 1 MSLOC each - 3 large overlapping increments - no explicit mention of spirals
- Up front requirements analysis and architecture (prior to all increments)
- Risk assessment is a continuing parallel activity that overlaps increments 1, 2

Experiences

- A wide range of definitions of spiral development (#1-6)
- Generally poor understanding of the role of risk management as a key element of spiral development (#2, #3, #4)
- Very poor execution of risk management during spirals (schedule pressure) (#2, #3, #4)
Serious concern over the loss of focus on architecture (#1, #5)
A mismatch between HW and SW lifecycles (#6)
A strong tendency to do the easy things first (#3, #4)
  - This can be viewed as the right thing to do if you consider the highest risk to your program to be the need to prove that it will be useful

Overly constrained programs
  - cost, schedule, and functionality
Unwillingness to openly discuss risk involved in software development
  - shooting the messenger
Lack of understanding of the principles behind spiral development
  - monotonically decreasing risk