Brady Field

A Spiral Model of Software Development and Enhancement

**Important points**

“The major distinguishing feature of the spiral model is that it creates a risk-driven approach to the software process rather than a document-driven or code-driven process. It incorporates many of the strengths of other models and resolves many of their difficulties.” [301]

“After a number of fixes, the code became so poorly structured that subsequent fixes were very expensive. This underscored the need for a design phase prior to coding.” [302]

“The waterfall model’s approach helped eliminate many difficulties previously encountered on software projects. The waterfall model has become the basis for most software acquisition standards in government and industry. Some of its initial difficulties have been addressed by adding extensions to cover incremental development, parallel developments, program families, accommodation of evolutionary changes, formal software development and verification, and stagewise validation and risk analysis.” [303]

“The spiral model of the software process (see Figure 2) has been evolving for several years, based on experience with various refinements of the waterfall model as applied to large government projects.” [304]

“An important feature of the spiral model, as with most other models, is that each cycle is completed by a review involving the primary people or organizations concerned with the product.” [305]

“The spiral model was used in the definition and development of the TRW Software Productivity System (TRW-SPS), and integrated software engineering environment.” [306]

“Design rationale information is of paramount importance in assessing the potential reusability of software components on future projects.” [307]

“Overall, risk-driven documents, particularly specifications and plans, are important features of the spiral model.” [308]

“The primary advantage of the spiral model is that its range of options accommodates the good features of existing software process models, while its risk-driven approach avoids many of their difficulties.” [309]

“The spiral model places a great deal of reliance on the ability of software developers to identify and manage sources of project risk.” [310]

“In general, the spiral model process steps need further elaboration to ensure that all software development participants are operating in a consistent context.” [311]

“Partial implementations of the spiral model, such as the Risk Management Plan, are compatible with most current process models and are very helpful in overcoming major sources of project risk.” [312]

**Disagreements**

“The evolutionary development model is ideally matched to a fourth-generation language application and well matched to situations in which users say, ‘I can’t tell you what I want, but I’ll know it when I see it.’” [303]

I have a few problems with this statement. First of all, it implies that the evolutionary model is a bad fit for any generation of language prior to the fourth generation. I would argue that third generation applications are more plentiful than fourth generation applications. Second, the model chosen is not directly linked to the language the application will be written in. One cannot reasonably argue that the evolutionary model is suited for some languages and not for others. Third, no model will compensate for the users not knowing what they want. Even the magically elusive automated software synthesis model requires the user knowing what they want beforehand.

**Questions**

I have no questions. The reading was straightforward.