David Lambertson

**A Spiral Model of Software Development and Enhancement Summary**

**Important Points from Article**

* “The spiral model presented in this article is one candidate for improving the software process model situation. The major distinguishing feature of the spiral model is that it creates a risk-driven approach to the software process rather than a primarily document-driven or code-driven process.” [301]
* “The primary functions of a software process model are to determine the order of stages involved in software development and evolution and to establish the transition of criteria for progressing from one stage to the next.” [301]
* “The waterfall model has become the basis for most software acquisition standards in government and industry.” [303]
* “Document-driven standards have pushed many projects to write elaborate specifications of poorly understood user interfaces and decision-supported functions, followed by the design and development of large quantities of unusable code.” [303]
* “The above concerns led to the formulation of the evolutionary development model, whose stages consist of expanding increments of an operational experience.” [303]
* “The transform model assumes the existence of a capability to automatically convert a formal specification of a software product into a program satisfying the specification.” [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 software projects. As will be discussed, the spiral model can accommodate most previous models as special cases and further provides guidance as to which combination of previous models best fits a given software situation.” [304]
* “This risk-driven [sub setting] of the spiral model steps allows the model to accommodate any appropriate mixture of a specification-oriented, automatic transformation-oriented, or other approach to software development.” [305]
* “These templates are useful not only for organizing project activities, but also as a residual design-rationale record.” [307]
* “The RTT [requirements traceability tool] establishes the traceability between itemized software requirements specifications, design elements, code elements and test cases.” [308]
* “In areas involving a high risk if the design turned out to be wrong, the design was carried down to the detailed design level, usually with the aid of rapid prototyping.” [308]
* “In areas involving a moderate risk if the design was wrong, the design was carried down to a preliminary-design level.” [308]
* “In areas involving a low risk if the design was wrong, very little design elaboration was done.” [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]

**Things I Didn't Agree With**

“For example, a design produced by an expert may be implemented by non-experts. In this case, the expert who does not need a great deal of detailed documentation, must produce enough additional documentation to keep the non-experts from going astray.” [310] I do not believe with this statement by Barry Boehm. I believe that the expert would be the manager and would oversee the non-experts, who will implement the design. He would be able to help and guide the non-experts and not need to create additional documentation for those non-experts.

**Things I Did Not Understand**

One part of this article I did not understand was when Boehm talked about “A detailed design go-back.” [308] I didn’t understand how he talked about artifacts in the programming process.