Evolutionary Prototyping

Evolutionary prototyping is a lifecycle model in which the system is developed in increments so that it can readily be modified in response to end-user and customer feedback.

Main Benefits	The ability to address risk early in the project, early feedback on whether the final system will be acceptance, and visible progress throughout the project.
Keys to Success	Using experienced developers, managing schedule and budget expectations, and managing the prototyping activity itself.
When to Use	Evolutionary prototyping works best when the customer is uncertain about what they want at the outset and their input must be solicited throughout the project to ascertain requirements.
Main Risks	The main risks associated with evolutionary prototyping are unrealistic schedule and budget expectations, inefficient use of prototyping, unrealistic system performance expectations, and poor design.

Overview

In evolutionary prototyping the system concept is developed as you move through the project. You begin by developing the most visible aspects of the system. You demonstrate that part of the system to the customer, and then continue to develop the prototype based on the feedback you receive. At some point, you and the customer agree that the prototype is "good enough," and you release the prototype as the final product.

It is probably best suited to business systems in which developers can have frequent, informal interactions with end-users. But it is also well suited to commercial, shrink-wrap, and systems projects as long as you can get end-users involved. The user interaction for these kinds of projects will generally need to be more structured and formal.

If evolutionary prototyping provides less control than you need or you already know fundamentally what you want the system to do, you can use evolutionary delivery or staged delivery instead.

CxOne Support

CxOne provides support for selecting of the most appropriate lifecycle for your project through the p*roject planning* materials. The project plan checklist, template, and guide provide a mechanism for selecting or defining the most suitable lifecycle.

Interactions with other Best Practices

Prototyping is an effective remedy for feature-creep when used with other practices.

Prototyping is also an effective defect-removal practice when combined with other practices. A combination of reviews, inspections, and testing produce a defect-removal strategy with the highest defect removal efficacy, lowest cost and shortest schedule. Adding prototyping to the mix produces a defect-removal strategy with the highest cumulative defect-removal efficacy (Pfleeger 1994a).

Further Reading

McConnell, Steve. *Rapid Development*. Redmond WA: Microsoft Press. 1996.

Connell, John and Linda Shafer. *Object-Oriented Rapid Prototyping*. Englewood Cliffs, N.J.: Yourdon Press, 1995.

Gordon, V. Scott and James M. Bieman. "Rapid Prototyping: Lessons Learned," *IEEE Software*, January 1995