

## References

- Crosby, P.B.: *Quality is Still Free*, 1996. McGraw-Hill, ISBN 0070145326. Modernized version of P.B. Crosby: *Quality is Free*, 1980. McGraw-Hill. Reissue 1992 ISBN 0451625854. Five stages of Management Maturity. Zero Defects. Absolutes of Quality. Attitude.
- Crosby, P.B.: *Quality Without Tears*, 1984. McGraw-Hill, ISBN 0070145113. Practical quality principles.
- Deming, W.E. *Out of the Crisis*, 1986. MIT, ISBN 0911379010. Landmark book on quality
- Gilb, T. *Principles of Software Engineering Management*, 1988. Addison-Wesley Pub Co, ISBN: 0201192462. Including an explanation of the "Evolutionary Delivery" method.
- Gilb, T. *Competitive Engineering*, 2005. Elsevier, ISBN: 0750665076. A Handbook for Systems Engineering, Requirements Engineering and Software Engineering, using Planguage.
- Larman, Craig and Victor Basili "Iterative and Incremental Development: A Brief History" IEEE Computer, June 2003.
- Larman, C. *Agile and Iterative Development*, 2004. Addison-Wesley Pub Co, ISBN: 0131111558. A Managers Guide: Agile principles, and comparing XP, Scrum, RUP and Evo.
- Malotiaux, N.R. *Evolutionary Project Management Methods*, 2001. <http://www.malotiaux.nl/nrm/pdf/MxEvo.pdf> Project management issues and first experiences with Evolutionary Project Management .
- Malotiaux, N.R. *How Quality is Assured by Evolutionary Methods*, 2004. PNSQC 2004 Proceedings. Also downloadable as a booklet: <http://www.malotiaux.nl/nrm/pdf/Booklet2.pdf> Newer experience after introducing Evo in some 25 projects in 9 organizations.
- Malotiaux, N.R. *Optimizing the Contribution of Testing to Project Success*, 2005. PNSQC 2005 Proceedings. Also downloadable as a booklet: <http://www.malotiaux.nl/nrm/pdf/EvoTesting.pdf> How to organize the testing process the Evo way.
- McConnell, Steve, *Rapid Development: Taming Wild Software Schedules*, 1996. Microsoft Press, ISBN 1-55615-900-5
- Prowell, S.J. et al., *Cleanroom Software Engineering, Technology and process*. Addison-Wesley, 1999. ISBN 0201854805



Upgrading Your  
Toolbox: Adapting  
and Adopting Tools  
& Practices  
Oct. 10-12, 2005  
Portland, Oregon

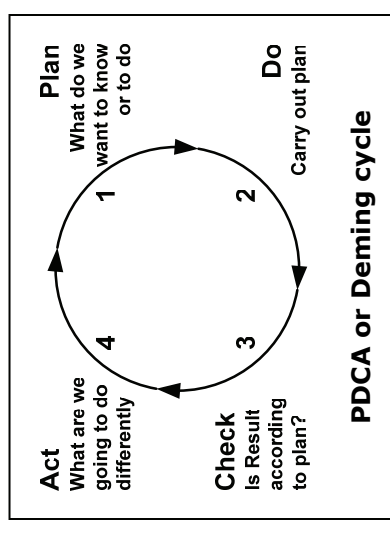
## Why Contemplate the Evolutionary Approach for Project Success?

Sponsored by PNSQC and the Rose City Software Process Improvement Network

PACIFIC NW  
SOFTWARE  
QUALITY  
CONFERENCE

## EVO

EVO is short for Evolutionary Delivery or Evolutionary Project-Management. As is typical of agile methods, EVO emphasizes short cycles and stakeholder involvement. Using EVO, you rapidly, frequently, and deliberately go through the Plan-Do-Check-Act (PDCA) learning cycle for the product, project and process, continuously thinking "what should we do, in which order, to which level of detail for now." EVO integrates planning, requirements, and risk management to create result management. It's actively induced evolution because we don't wait for evolution to happen, we make it happen.



## Product—Project—Process

EVO allows us to organize the project in a way that we discover and implement the optimum result at the lowest cost. This implies optimizing the effectiveness and efficiency of discovery and implementation. It also means that we change our estimation practice from optimistic to realistic, so that we can predict the future more accurately. We dynamically keep our plans up to date in order to maintain control over the result. In EVO we do not only design the product, we also design the project.

Because it is continually being improved as a process, EVO is made up of the best set of methods we know at a given time. If we find a better way, we change to the better way. Not only do we employ PDCA on the product and project activities, we also constantly and dynamically apply the PDCA cycle to the methods we use. If another method seems better, we try it. We experiment. But we deliberately Check and Act: if the new method is better, we change to the better method. If the new method is not better, we revert to the last known best method. Methods, processes and procedures are there to help us. If they don't, we discard or shelve them. Consequently, EVO practice may be different across projects and organizations because of differences in culture or experience.

## EVO's History

Evolutionary software and systems development methods have been used for decades. The term Evolutionary Development, or EVO, was invented by Tom Gilb in 1976. Early descriptions of evolutionary delivery, then called "incremental delivery," are described by Harlan Mills in 1971 and by F.P. Brooks in his famous "No silver bullet" article in 1987. A practical elaboration of the theory was written by Tom Gilb in *Principles of Software Engineering Management* in 1988 and *Competitive Engineering* in 2005 where Gilb emphasizes the requirements elicitation, quantification, and prioritization aspects of EVO, Niels Malotiaux added practical techniques for organizing EVO projects in 2001 through 2005.

## Comparison of Common Software Development Lifecycles<sup>1</sup>

Lifecycle Model Capability	Pure Waterfall	Code-and-Fix	Spiral	Modified Waterfalls	Evolutionary Prototyping	Staged Delivery	Evolutionary Delivery	Design-to-Schedule	Design-to-Tools	Commercial Off-the-Shelf Software	Evolutionary Delivery (EVO) <sup>2</sup>	Extreme Programming (XP) <sup>3</sup>
Works with poorly understood requirements	Poor	Poor	Excellent	Fair to excellent	Excellent	Poor	Fair to excellent	Poor to fair	Fair	Excellent	Excellent	Excellent
Works with poorly understood architecture	Poor	Poor	Excellent	Fair to excellent	Poor to fair	Poor	Poor	Poor	Poor to excellent	Poor to excellent	Excellent	Excellent
Produces highly reliable system	Excellent	Poor	Excellent	Excellent	Fair	Excellent	Fair to excellent	Fair	Poor to excellent	Poor to excellent	Excellent	Excellent
Produces system with large growth envelope	Excellent	Poor to fair	Excellent	Excellent	Excellent	Excellent	Excellent	Fair to excellent	Poor	N/A	Excellent	Excellent
Manages risks	Poor	Poor	Excellent	Fair	Fair	Fair	Fair	Fair to excellent	Poor to fair	N/A	Excellent	Poor to fair
Can be constrained to a predefined schedule	Fair	Poor	Fair	Fair	Poor	Fair	Fair	Excellent	Excellent	Excellent	Excellent	Fair
Has low overhead	Poor	Excellent	Fair	Excellent	Fair	Fair	Fair	Fair	Fair to excellent	Excellent	Excellent	Fair
Allows for midcourse corrections	Poor	Poor to excellent	Fair	Fair	Excellent	Poor	Fair to excellent	Poor to fair	Excellent	Poor	Excellent	Excellent
Provides customer with progress visibility	Poor	Fair	Excellent	Fair	Excellent	Fair	Excellent	Fair	Excellent	N/A	Excellent	Excellent
Provides management with progress visibility	Fair	Poor	Excellent	Fair to excellent	Fair	Excellent	Excellent	Excellent	Excellent	N/A	Excellent	Fair
Requires little manager or developer sophistication	Fair	Excellent	Poor	Poor to fair	Poor	Fair	Fair	Poor	Fair	Fair	Fair	Poor

<sup>1</sup> Adapted from *Rapid Development: Taming Wild Software Schedules* by Steve McConnell (Microsoft Press, 1996), pp. 156-157

<sup>2</sup> Column contributed by Niels Malotaux.

<sup>3</sup> Column contributed by Jim Shore