Software Engineering: Fact & Fancy

Brian Lawrence
Coyote Valley Software
brian@coyotevalley.com
www.coyotevalley.com
(408) 578-9661

SSQA
March 2006

A Little Joke

Abraham Lincoln:

"If you call a tail a leg, how many legs does a dog have?"

"Four!

Just because you call a tail a leg, that doesn't make it one."

Routine vs. Non-Routine Tasks

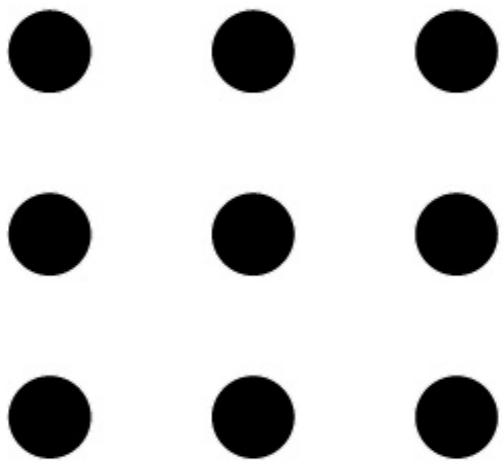
- *Routine tasks* are ones where we know how to do them before starting, and know that for a specific amount of effort, we will get a corresponding amount of progress.
- *Non-Routine tasks* are ones where while we may know how to do them, we don't exactly know how much time or effort we will need to expend to complete them.

A Brief Case Study

- Form into groups of no more than four people. Pretend you are interviewing to work on a project as a group.
- From among the group, choose one person to serve as an observer.
- Observers – Come up to the front and I'll give you your instructions.

A Brief Case Study
- Routine Task

- On a piece of paper, draw nine dots in 3 rows of 3, so that the dots are evenly spaced and both vertically and horizontally symmetric. E.g.:



A Brief Case Study - Non-Routine Task

- Connect the nine dots using as few contiguous straight lines as possible.
- The group with the fewest lines wins*.

* This problem has been solved many ways. To be competitive, you probably need to use fewer than four lines.

A Brief Case Study

One of several solutions.

SE Activities

Charters	Routine
Life Cycles	Routine
Inspection	Both
Requirements	Non-Routine
Architecture	Non-Routine
Project Mgmt	Can be Both

SE Activities

Risk Mgmt	Non-Routine
CM	Routine
Testing	Non-Routine
QA	Routine
Appraisals	Routine
Retrospectives	Can be Both

Fact or Fancy?

Software Engineering projects can be repeatable.

- This is a fundamental premise of well-known, widely-accepted SE process models (e.g. CMM).
- Fancy!
- Non-routine tasks cannot be made repeatable.

Fact or Fancy?

Software project sponsors will always want precise and accurate predictions for cost and schedule.

- Fact!
- Unfortunately, they can't always get precise and accurate predictions.
- Unpredictable projects can still be successful.

Fact or Fancy?

Software Engineering is primarily technological in nature.

- Fancy!
- Most of the challenges are social, not technical.
- This notion is NOT widely supported by the SE literature.

Fact or Fancy?

There are some very solid practices in Software Engineering.

- Fact!
- There are several SE practices I would personally always recommend in any software project.
- They may take many forms.

Fact or Fancy?

Measurement practices in Software Engineering are safe and effective.

- Fancy!
- In my experience, most measures have done more harm than good.
- There are plenty of people promoting measurement who are apparently blissfully oblivious of the real effects of measurement.

Fact or Fancy?

Software Engineering is fact, a true engineering discipline.

- Fancy!
- At best, SE is not much more than a *Craft*.
- Much of the SE literature likes to pretend that it is true engineering.
- See Mary Shaw's explanation.

References

Adams, James, *Conceptual Blockbusting*, Addison-Wesley, 1986

DeMarco, Tom, “Mad About Measurement,” *Why Does Software Cost So Much?*, Dorset House, 1995

Kaner, Cem, “Rethinking Software Measurement,” *STQE*, March 2000, Vol. 2, No. 2.

Glass, Robert, *Software Runaways*, Prentice-Hall, 1998

Shaw, Mary, “Three Patterns that help explain the development of Software Engineering,” 1997

Readings

- Brooks, Fred, *The Mythical Man-Month*, 20th Anniversary ed. Addison-Wesley, 1995.
- DeMarco, Tom, and Tim Lister, *Waltzing With Bears*, Dorset House, 2003
- Gause, Don, and Jerry Weinberg, *Exploring Requirements*, Dorset House, 1989.
- Kaner, Cem, Jack Falk, and Hung Nguyen, *Testing Computer Software*, Wiley, 1993
- Kerth, Norm, *Project Retrospectives*, Dorset House 2001.
- McConnell, Steve. *Rapid Development*. Microsoft Press. 1996.
- Rechtin, Eberhart, and Mark Maier, *The Art of Systems Architecting*, CRC Press, 1997
- Royce, Winston W. “Managing the Development of Large Software Systems,” *IEEE Software*. (April 1970): 118-127.
- III, “Immunizing Your Project Against Foreseeable Failure,” *STQE Magazine,* Jan. 2000.
- Shaw, Mary and David Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall. 1996
- Wiegers, Karl, *Peer Reviews in Software*, Addison-Wesley, 2001
- Weinberg, Jerry, *Quality Software Management,* 4 Volume Trilogy, Dorset House, 1991-1997

Acknowledgement

The following colleagues were cheerfully volunteered to serve as reviewers of this presentation:

- Rick Brenner – *Chaco Canyon Consulting*
- III - *Systemodels*
- Judah Mogilensky – *Process Enhancement Partners*
- SuZ Garcia – *Software Engineering Institute*