Why We Need Empirical Information on Best Practices

Dr. Richard Turner
The George Washington University

Best practices are widely recommended as a way to improve organizational performance, especially in software-related endeavors. This article takes a skeptical view of the current way best practices are identified and prescribed. It identifies relevant information that is often missing from best practice discussions and recommends an alternative approach to gathering, evaluating, and applying that information.

In the history of software development and acquisition, one of the most often prescribed curatives for their continuing infirmities, aches, and agues is the identification and implementation of *best practices*. Of course, the notion of what defines a best practice is not clear. Some best practices, for example configuration or risk management, are actually disciplines seen as crucial to success. Other best practices are broad approaches or philosophies such as architecture-first development or Integrated Product and Process Development. A third type of best practices, peer reviews for example, are actually practices proven to be beneficial in a specific way. In reality, the term has been so broadly applied as to be nearly meaningless.

In spite of being *definitionally challenged*, best practices continue to arise – sometimes as ephemeral answers du jour and other times as lasting wisdom. They populate the lists and fill the books that we turn to for guidance. Unfortunately, we all too often find that the benefit is more in the eyes of the beholder than in any measurable result of implementing the enshrined practices. We ultimately do not know, beyond anecdotes and sales pitches, whether a practice will work for us. So, to move from faith toward science, we need to approach best practices in a skeptical but constructive manner. I believe that the best way to do this is through focused empirical studies and careful analysis that result in a validated assessment of the practice's cost and real benefit.

Some History

My earlier research into the adoption of best practices in defense acquisitions uncovered considerable recognition of the most widely referenced best practices, but very little real implementation [1]. There were good reasons for the unsuccessful implementation of even the highly recommended practices, and most had to do with lack of information.

I found that practices – best or otherwise – generally do not fall into the *one-size-fits-all* category, and it is not easy to evaluate how appropriate a practice is for a particular organization or program. Most practices also have hidden assumptions and conditions for use, and there is little available support for evaluation and selection. When a practice is chosen, there is often little information on how to implement it in the real world. Consequently, managers often find themselves acting on a *faith-and-gut* feel in deciding what practices to implement.

There are also the instances of best practices that are not. A case in point is the venerated heuristic that the larger a software module, the more likely it is to have defects. Surprisingly, empirical study at NASA's Software Engineering Laboratory showed that smaller modules actually increased the defect rate for a period, and that there existed a *sweet spot* where the module size corresponded to the fewest defects. The exact placement of the sweet spot depends on a number of characteristics about the software being developed, but Figure 1 illustrates the general finding, which is in direct conflict with the previously held *best practice*.

"... empirical study at NASA's Software Engineering Laboratory showed that smaller modules actually increased the defect rate for a period ..."

Applying Empiricism: Questions Needing Answers

Empiricism, in this context, can be thought of as a methodical approach to the gathering and analysis of data about a specific practice. It is applying, to the best of our ability, scientific principals to the evaluation and validation of practices with the goal of producing usable information. This is more than collecting anecdotes or drawing general conclusions from a few unstructured experiences.

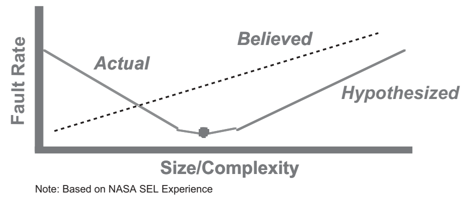
Empiricism should not be confused with quantitative analysis, since there are ways to methodically collect and meaningfully analyze qualitative data. Quantitative data is certainly a worthy goal, but in some cases it is very difficult to obtain. For that reason, we include several qualitative approaches, including workshops and expert opinion, under our empirical umbrella.

The primary purpose of methodical analysis of practices is to gather and maintain data to answer specific questions. Using this data, including knowledge gained from lessons learned in actually implementing practices, we can build tools to help select practices that are appropriate for a particular project. Let us look briefly at some questions to which empiricism can provide validated answers.

How Much Will It Really Cost?

It is usually risky to order from a menu without prices, so the first thing we need to know is the size of the bill. Let us consider some of the major costs we need to define and capture. How many hours of training are needed? Are there tools or other infrastructure required? Of course, these upfront costs might just be the tip of the iceberg. What are the costs of the effort and resources needed to actually apply the practice? Are there license fees or equipment maintenance associated with the infrastructure? Unexpected costs

Figure 1: *Notional Findings on Module Size versus Fault Rate*

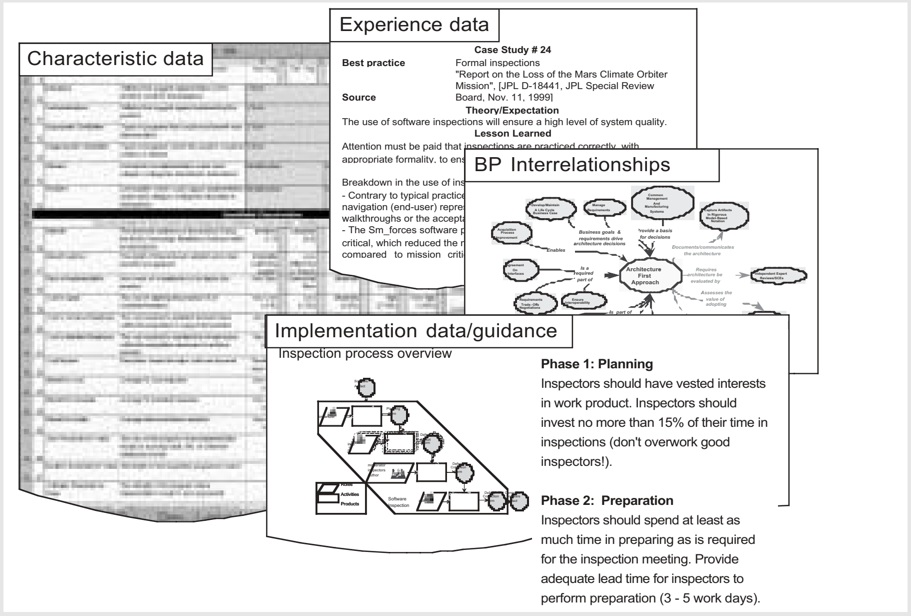


The Department of Defense Best Practices Clearinghouse — First Steps Toward Empiricism

The Office of the Under Secretary of Defense (Acquisition, Technology and Logistics) Defense Systems has initiated an activity to define an empirically-informed clearinghouse for software acquisition and development best practices. The Data Acquisition Center for Software (DACS), the Fraunhofer Center at the University of Maryland, and the Center for Software Engineering at The University of Southern California are specifying the infrastructure and processes required for a centralized, empirically-based resource for acquisition and development projects as shown in the Conceptual Best Practices Clearinghouse Data figure, below.

The clearinghouse is envisioned to maintain validated practice information, support user-driven selection of practices, provide step-wise implementation guidance, and track implementation results. Easy-to-use, informative tools will suggest appropriate practices based on goals, risks, life-cycle phase, and program environment. Support for evolving from basic to advanced practices could also be included. Web-based access tailored to user needs is planned, as well as an active infrastructure to link expertise and information providers to users via communities of practice, courses, workshops, publications, and shared pilot projects.

A user advisory group is being established to ensure that the products and tools to be provided will meet the needs of developers and acquirers. The clearinghouse team is seeking submission of best practices, implementation and results data, and lessons learned from development and acquisition organizations. Coordination with service and agency best practice and lessons learned repositories is underway. For more information, contact Kathleen Dangle of the Fraunhofer Institute at <kdangle@fc-md.umd.edu>, or Tom McGibbon of the DACS at <tom.mcgibbon@itt.com>.


Characteristic data
Experience data
Case Study # 24
Best practice Formal inspections
Source "Report on the Loss of the Mars Climate Orbiter Mission", [JPL D-18441, JPL Special Review Board, Nov. 11, 1999]
Theory/Expectation
The use of software inspections will ensure a high level of system quality.
Lesson Learned
Attention must be paid that inspections are practiced correctly, with appropriate formality, to ens
Breakdown in the use of ins
- Contrary to typical practice
navigation (end-user) repres
walkthroughs or the accepta
- The Sm_forces software p
critical, which reduced the r
compared to mission criti
BP Interrelationships
Develop/Maintain A Life Cycle Business Case
Manage Requirements
Common Management And Manufacturing Systems
Capture Artifacts In Rigorous Model-Based Notation
Acquisition Process Improvement
Business goals & requirements drive architecture decisions
Provide a basis for decisions
Documents/communicates the architecture
Enables
Agreement On Interfaces
Is a required part of
Architecture First Approach
Requires architecture be evaluated by
Independent Expert Reviews/SCEs
Assesses the value of adopting
Requirements Trade-Offs Negotiations
Ensure Interoperability
Is part of
Implementation data/guidance
Inspection process overview
Roles
Activities
Products
Software Inspection
Phase 1: Planning
Inspectors should have vested interests in work product. Inspectors should invest no more than 15% of their time in inspections (don't overwork good inspectors!).
Phase 2: Preparation
Inspectors should spend at least as much time in preparing as is required for the inspection meeting. Provide adequate lead time for inspectors to perform preparation (3 - 5 work days).

can doom any benefit that might be achieved. Maintaining information on how much a practice costs to implement is a key empirical characteristic.

What Is the Actual Benefit?

OK, we have an idea of the cost but really, how good is that best practice entree? What exactly do we expect from implementing the practice? Will it shorten the schedule, raise quality, or lower cost? If so, by how much? What specific risks could it mitigate? How are benefits measured? Sometimes there are hidden benefits or ones that surface late in the life cycle. On the other hand, even obvious benefits might need actions outside of the project's control to be fully realized.

For example, peer programming can provide higher quality and shorter development times, but successful implementation might require changes to corporate policy regarding reward structure, office space, or equipment allocations. Validating the benefit ensures that recommended practices have a fighting chance of helping programs that implement them. Benefits may not be explicitly captured in dollars, but the type, nature, and magnitude can be collected and analyzed.

What Is the Pedigree?

It is always good to know where the practice came from and who actually established it as a best practice. Is it technologically mature? Are there studies that suggest it works? Who has successfully implemented it? This is especially true when proprietary components such as tools or processes are part of the practice. *Caveat emptor* is a pretty good mantra for our empirical activities. Data on the number of implementations, breadth of application, and the level of consensus on the practice's value by experts are means to address pedigree empirically.

Is the Environment a Critical Success Factor?

Every practice does not apply to every type of project. Does the practice assume a particular type of development or acquisition environment? Does it only work for small projects? Was the best practice identified in an environment of stable requirements or is its primary benefit only realized in a situation of constant change? When in the product life cycle is it best applied? It might not be helpful to implement a requirements practice when the program is knee-deep in integration testing. What is the size or criticality threshold at which the practice begins to pay off? What is reasonable for the F-35 or a Missile Defense Agency component might not be appropriate for a commercial off-the-shelf-based, Web-enabled training application. Maintaining the characteristics of the environments where a practice has been implemented and the associated results is one way to capture this data.

How Long Before It Works?

Knowing the time it takes for a benefit to be realized is one of the subtlest questions to answer. Does the practice provide immediate benefit, or do its effects have to trickle down through the development or acquisition process for months (or years) before actually helping? Compare the benefit latency of peer reviews with that of a process improvement program. The first pays dividends immediately while the second takes months to show measurable value. Knowing the benefit latency also has an unfortunate down side

– if it will not help by the end of someone's watch, it may be more difficult to gain support for implementation. Benefit latency can be characterized based on lessons learned and experience reports.

Are There Other Barriers?

Practices are implemented by people, so they imply change. The project team's attitude, capabilities, and personality can raise all sorts of problems. Will they accept and adopt the practice or just go through the motions? As with any change, corporate culture also plays a part. Will management buy in or fight it every step of the way? How will the practice impact the organizational infrastructure? Knowing what barriers have historically manifested is a major advantage in planning successful implementation. Barriers can be identified from experience, rated as to impact, and organized into useful categories.

Can We Implement This?

Finally, we need to understand the probability of successfully bringing the practice to our particular program. Are the existing resources and authority sufficient to implement the practice? It is usually possible to implement something that affects the way a team works internally, but implementing something like Integrated Product and Process Development with all of the significant impacts on other stakeholders requires enormous resources and power. Is there sufficient time left in the project to achieve any benefit? Clear instructions as to how to implement the practice are also priceless. Knowing about available tools or consultants or classes can save the effort of making it up as you go.

There has to be an honest assessment of the implementation requirements and the ability to meet them, or its likely implementation will be incomplete or shoddy, and the project possibly worse off than before. Capturing the scope of control and other requirements for implementation is relatively straightforward and can support implementation guideline development.

Conclusions

You probably recognized that answering these questions could be extremely difficult. It will take a focused, ongoing effort to gather and maintain the data required to validate the effectiveness and costs of practices. This will be ongoing because the data, as well as the practices, will change continuously over time. We know that every practice has associated cost and benefit, maturity and pedigree, preferred environment, benefit latency, organizational barriers, and required competencies for successful implementation. We need to seize the opportunity to capture, analyze, and package the precious information of others' experiences. There is so much knowledge and experience being gained daily by Department of Defense programs that it is a travesty to let it simply vanish when the technology exists to make it useful and available.

Consider the impact to projects of a successful effort to empirically gather data and validate best practices. How marvelous to be able to pick vetted, proven practices that apply to our particular needs and resources, reasonably confident that the implementation will bring about predictable benefits. The reduction of rework and wasted effort could well dwarf the expenses associated with the validation effort. Above all, projects would have another means to increase their probability of success in an environment that has seen all too many failures.◆

Reference

1. Turner, Dr. Richard. "A Study of Best Practice Adoption by Defense Acquisition Programs." CROSSTALK, May 2002: 4-8.

About the Author

Richard Turner, D.Sc., is a research professor in Engineering Management and Systems Engineering at The George Washington University. He is currently supporting the Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics, Defense Systems. Turner is a co-author of "CMMI Distilled" and "Balancing Agility and Discipline: A Guide for the Perplexed." Turner has a Bachelor of Arts in mathematics from Huntingdon College, a Master of Science in computer science from the University of Louisiana at Lafayette, and a Doctor of Science from George Washington University.

George Washington University
1931 Jefferson Davis HWY
STE 104
Arlington, VA 22202
Phone: (703) 602-0851 ext. 124
E-mail: rich.turner.ctr@osd.mil

COMING EVENTS

April 19-22
2004 Systems and Software Technology Conference
Salt Lake City, UT
www.stc-online.org

May 11-13
Technet International
Washington, DC
www.technet2004.org

May 17-21
STAREAST
Orlando, FL
www.sqe.com/stareast/

May 23-28
26th International Conference on Software Engineering
Edinburgh, Scotland
www.jupiterevents.com

June 2-4
Sacmat 2004
Yorktown Heights, NY
www.sacmat.org

June 11-13
ACM Sigplan 2004 Conference on Language Compilers and Tools for Embedded Systems
Washington, DC
http://lctes04.flux.utah.edu

June 23-26
Agile Development Conference 2004
Salt Lake City, UT
www.agiledevelopment conference.com

June 27- July 2
USENIX Annual Technical Conference
Boston, MA
www.usenix.org/events/usenix04