JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2003**

Vol. 2, No. 4, July-August 2003

The Best Practice Promise and Myth

Mahesh H. Dodani, IBM Software Group, U.S.A.

1 BEST PRACTICES

"In theory, there is no difference between theory and practice. But, in practice, there is." - Jan L.A. van de Snepscheut

The quest and use of best practices has always been an integral part of any software engineering practice. Best practices promise quality, consistency, efficiency and flexibility in engineering software systems. On the other hand, best practices can also be abused or misused – once a best practice has been established, it can be used by its proponents to force people to follow it blindly. So, what is a best practice? Are there any best practices in (OO) software engineering? How does something become a best practice, and how does it stay as a best practice?

What are the characteristics of a best practice? There are some obvious characteristics derived from the name:

- A proven practice in the given context. In OO software engineering a best practice can be a coding practice, a design pattern, a reference architecture, or a business pattern.
- Among the proven practices, it is the best at achieving some result. In OO software engineering, it is the coding practice that is most efficient, the design pattern that facilitates ease of change, the reference architecture that adapts to differing requirements, and the business pattern that enables composite processes.
- It is well documented, used widely by the community, and continually evolving. In OO software engineering the documentation is done through standards and common templates and distributed through web sites, books and conferences. The community are the tens of thousands of OO practitioners, researchers, and innovators. The best example of a continually evolving best practice are design patterns http://c2.com/cgi/wiki?DesignPatterns – which has evolved from the Gang-Of-Four patterns which is in wide use within the community to many related practices including learning patterns, architectural patterns, and e-business patterns.

However, the major difficulty in describing best practices is the ability to measure the results. How does one measure a proven practice, that is best in its class, is well

Cite this column as follows: Mahesh Dodani: "The Best Practices Promise and Myth", in *Journal of Object Technology*, vol. 2, no. 4, July-August 2003, pp. 65-68.
http://www.jot.fm/issues/issue_2003_07/column5

documented, used by the community and continually evolving? The only example of such a measure is the Capability Maturity Model (http://www.sei.cmu.edu/cmm/) for software which describes the best practices underlying software process maturity. The five maturity levels are intended to help software organizations measure and improve their software processes:

1. Initial: The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2. Repeatable: Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3. Defined: The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
4. Managed: Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5. Optimizing: Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

An organization can measure the maturity level of their process based on the above characteristics, and know how to move themselves to the next maturity level. Note that this approach has evolved and matured itself over the last twenty years!

2 BEST PRACTICE PROMISES

The most compelling example of best practice in OO software engineering is embodied in the patterns movement. Even though people had been designing OO systems for reuse since the inception of OO languages and systems in the 1980s, it was many years later that these design best practices and experience found a vehicle for being well documented and disseminated in the community. The Gang-Of-Four design patterns were published in 1994 and catalogued 23 design patterns that defined the best practices of building reusable systems. The key aspects of these patterns were that they were already in "common" use by experienced designers in building reusable OO systems and they were the "best" in the class as their reusability and flexibility had been proven. The problem was that these best practices were part of folklore, and known only to the experts who used them and disseminated them to their teams and the small community that they touched. The design patterns catalog paved the way for these design patterns to have a common way to be described, disseminated, and used. This is a very important step in establishing a best practice, as it now can become part of the community, and have a chance of evolving to address the ever changing landscape of OO software engineering.

These design patterns are now ingrained within the OO community, and are being refined, extended, and used in all aspects of designing OO systems, frameworks, and tools. As an example, consider the Java 2 Enterprise Edition (J2EE) technology and its component-based model which is the de facto standard to enable development of secure, robust and interoperable business applications. The J2EE component model specifies platform architecture, application components, containers, set of standard J2EE services, interoperability requirements and platform contracts (APIs, SPIs, network protocols and deployment descriptors). J2EE has used and evolved the original design patterns to make the framework flexible and reusable, and has defined its own catalog of design patterns to showcase the best practices in J2EE solutions to common problems as described in http://java.sun.com/blueprints/patterns/catalog.html.

As another example of the evolution of the design patterns, consider the IBM Patterns for e-business (http://www-106.ibm.com/developerworks/patterns/), which define a set of proven, reusable architectures that can drive the design, development, implementation and extension of e-business applications. [Please see my previous article introducing this best practice http://www.jot.fm/issues/issue_2003_03/column3.] Since e-business has been around for such a short time, people were skeptical that best practices could already be defined on how to drive business requirements into scalable and flexible e-business applications. However, on closer inspection, these patterns have all of the characteristics of best practices: they were culled from actual experience, have been proven in practice, are the best in their class, have been well documented, and have been disseminated to and in use by the community.

3 BEST PRACTICE MYTHS

The promise and benefits of having best practices have led many companies to identify and establish them at all levels of an organization. In their haste to gain this advantage, people tend to abuse/misuse the concept of best practice. Realizing the power of a "best practice" moniker, some people take advantage of myths to establish a best practice on something that isn't, and then use it to bludgeon everyone into following or using it. The following paragraphs identify some common myths of best practices.

I Work Therefore I Am Myth: The most common myth around best practices is that if there is a solution to a problem (especially if the problem has been difficult to solve), then this solution should qualify automatically for a best practice. Since this is usually the first solution to the problem, it generates enough euphoria among the small community affected by the problem that there is a tendency to declare it a best practice. Of course, the creators of the solution may use this euphoria to their advantage to claim their solution to be a best practice, and therefore ensure that the community starts using it immediately. The main issue with this "best practice" is maturity. The solution has not matured into a practice, and further into a best practice.

Guilty by Association Myth: A practice that uses other best practices does not automatically become a best practice. This myth is most prevalent in the software

engineering arena, where we rely on best practices as defined in coding practices, patterns, frameworks to guide us and facilitate creating new practices. However, just because a best practice is used in creating a (new) practice does not automatically qualify it as a best practice. It may increase its chances of maturing and succeeding as a best practice. However, it has to go through the same maturing steps, including the new practice being culled from experience, proven best results from using the practice, and being documented and used by the community. Even in the patterns world, it is important that we are careful in how to use it. The GOF authors appropriately warn us on how "not" to use them: "Design patterns should not be applied indiscriminately. Often they achieve flexibility and variability by intoroducing additional levels of indirection, and that can complicate a design and/or cost you some performance. A design pattern should only be applied when the flexibility it affords is actually needed." http://www.awprofessional.com/catalog/product.asp?product_id={E1CD5BE7-E008-481B-8D0C-8E80CE9978F9}.

Open Standards Myth: The complexity of current e-business applications and systems requires a set of standards, common templates, and open environments to facilitate interoperability and flexibility. However, the fact that a practice uses (or relies on) open standards does not automatically make it a best practice. In the OO world, we know of many examples of people using standards (e.g. UML, XML, etc.) and still produce very poor results. In fact, the antipattern movement has been established within the OO community to identify these common problems with solutions, and then show how to refactor the solution to get rid of the problem. An appropriate antipattern catalog can be found in http://c2.com/cgi/wiki?AntiPatternsCatalog.

In conclusion, best practices are the cornerstone of effective and efficient software engineering as it culls the experiences and proven practices from the field and facilitates its use and evolution within the community. However, it is important that you recognize the characteristics of best practices and ensure that they are in place before using it.

About the author

Mahesh Dodani is an e-business architect with IBM Software Group. His primary interests are in enabling individuals and organizations to tackle complex e-business industry solutions. He can be reached at dodani@us.ibm.com.