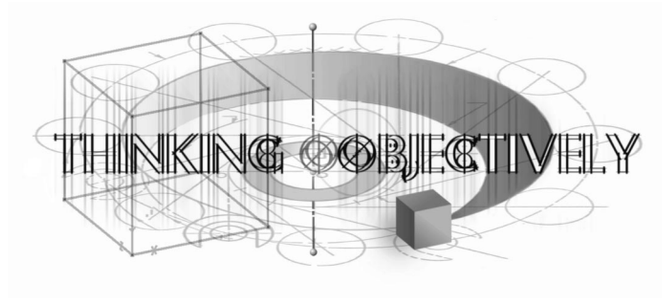

Mohamed E. Fayad and Mauri Laitinen

Process Assessment Considered Wasteful

The first step in any software process improvement program must be assessment. Assessment is supposed to give an organization a sense of where it stands in terms of software production skills. While we believe assessments can be valuable, and in some cases necessary, there are many instances in which assessment is at best wasteful and at worst counterproductive.

Assessment Background

In most assessment models, the organization evaluates its development capability against a set of "best practices" that are supposed to be found in effective organizations. The number of practices, their mastery, and their level of integration into the development organization determine the organization's assessment score. There are a large number of process improvement initiatives, of which, the best known are the Software Engineering Institute's Capability Maturity Model (SEI CMM), SPICE, the U.S. Department of Defense SDCE, ISO 9000, and ISO/IEC 12207. Some programs allow self assessment while others require outside certification.

The SEI CMM is one of the best-known and most widely discussed software process improvement model. It defines five levels of organizational maturity, from initial or chaotic to optimizing. Each increasing maturity level, starting at level 2, has associated with it a set of key process areas. For example, level 2 includes, among other things, requirements management and project planning as key areas. Level 3 includes training and peer reviews. Levels 4 and 5 include software quality management and defect prevention, respectively. Each level also includes the process areas of its lower levels. The SEI also specifies the methods by which organizations can assess themselves against the CMM or use an outside agency to perform the assessment.

Problems With Assessment

While all of the mentioned process improvement initiatives are valuable, and we can hardly argue with the need for process-oriented software improvement, we believe there are a number of problems with assessments that limit or misdirect attention. We use the CMM as our prime example although many of the points apply as much to other assessment schemes.

1. The most obvious reason to question the value of assessment is that for organizations just starting, it can be a waste of money and time. Few level 1 software organizations believe they are at level 3 or above. Assessments are very expensive, and in an immature organization the results are likely to be meaningless. For example, if a department does not use configuration management to keep track of product versions, there is no process to assess. Until configuration management is put in place, and used for a while, assessment of this facet of development will have significant cost but will not even serve as a baseline for further measurement.

PAUL WATSON

Time and effort could be better spent in acquiring and implementing a configuration management system.

2. All assessment models are artificially derived, and while all describe generally agreed-upon practices, the list itself is artificial. When an organization compares itself to an assessment model, it is comparing a real-world organization's practices against an idealized list.

The first quibble with the idealized list is that it is a one-size-fits-all representation. A 10-person software startup uses the same criteria as a giant defense contractor developing large systems for decades. More importantly, the assessment is the same for the huge corporation and its subcontractors. This is only reasonable if the effort in assessment does not exceed the development effort.

3. A more serious objection is that the idealized list of practices has not been proven to work. Although individual practices within the assessment model are of unquestionable value, the models' lists of practices and their order of adoption has not been shown to be either necessary or sufficient to produce a marked organizational improvement. Moreover, an organization experienced in software development may have a set of practices that help it produce good software but do not map well into the model processes. For example, in SEI's CMM level 3, project performance is supposed to be tracked by key activity. If tracking is done instead by a person or sub-group, it may be effective but doesn't match the assessment criteria.

Even more interesting is the possibility that other factors may be involved. Candidate factors include improvements in software tools, the "startup effect" in which new initiatives get much more highly qualified and motivated people than standard projects, and the idea of "heroic efforts" [1].

4. The CMM levels are unevenly weighted in terms of membership, cost achievement, and value. For example, level 3 seems to be the first acceptable plateau for a software development to aspire to. Next year, the U.S. Department of Defense will require CMM level 3 certification to qualify for contracts. This suggests that level 2 is nothing more than an "at least we're not level 1" sort of designation. Regardless of the SEI's original intent, levels 1 and 2 are now negative designations. By contrast, moving from level 4 to level 5 is a news-making event since it happens so rarely.

In the decade since the CMM was first published, most development organizations are still at level 1, the bottom level, and only a handful of organizations in the world are accepted to be at level 5. Most of the world's best-known commercial software is produced by organizations at or below level 3. These facts suggest that level 1 needs more differentiation and that level 5 has little relevance to most software development. A group of experienced software developers who work with few formal processes are lumped together with monkeys pounding at terminals in level 1. While such a characterization may please process purists, it is not especially useful for assessing the capabilities of the majority of development groups in the world.

At the other end of the scale, what is the value of attaining CMM level 5? If getting there is so difficult and rare, does it have meaning for most development organizations? Perhaps it is similar to the difference between joggers and Olympians. The jogger runs for pleasure and exercise. While it might be enjoyable to fantasize about winning a world-class event, the talent, time, and dedication it takes to run at that level make the comparison meaningless. This is not to belittle the Olympic contender: competitive running is just a different activity of interest to and attainable by the very few who value its particular payoff. Assessing joggers against Olympians, then, is a meaningless activity.

So, should CMM level 5 be a meaningful goal for most organizations, or should it be the particular goal of those few groups who value whatever reward it brings? If it is limited, then why include it in the assessment? And if level 3 is all that is needed to get contracts, is the incentive there to move beyond it?

If we extend this question of rewards, perhaps there should be different assessments for different groups. Software groups that work on life-critical systems might find value in the more stringent requirements of the upper levels. Groups working on the more common and less-critical systems might benefit from a greater differentiation in the lower assessment scales.

5. Moving to levels 4 and 5 sounds worthwhile, but there is little empirical evidence to justify the move. As stated before, while some of the assessment practices

correlate with more effective development, there is no evidence that all the processes or the order of their acquisition is right. In fact, for the CMM there is some evidence, both direct and indirect, to the contrary.

As reported by El Emam and Briand, many organizations find the early adoption of certain processes, such as change control and code reviews, more effective than adopting them in the recommended sequence [2]. Bamberger reports that when she works with clients, she helps them look more at the essence of the CMM's ideas rather than the explicit maturity levels involved, so they can get control of their software projects [3]. At lower levels then, getting a start in controlling projects is more important than orthodox progression through levels of maturity. The indirect data comes from Herbsleb, et al., who report approximately 44% of survey respondents had little or limited success addressing the CMM assessment recommendations [4].

While not conclusive, it is clear that adopting the CMM recommended practices is not especially easy, and for some it may be completely impractical (although we can't tell from the limited data). If the recommendations are hard to adopt, then it may be that the current CMM assessment has not identified the optimal software process improvement (SPI) characteristics. And if it is difficult to progress at the lower levels, it may help explain why progress to the upper levels is so rare.

If the SPI practices recommended aren't optimal, then they should not be cast in the assessment as the important activities. It makes them harder to change as more is learned. To make an analogy, with modern skiing instruction and equipment, most people start skiing in one day. Twenty years ago, the number of people who skied after one day was much smaller, and the number of people who adopted the sport was likewise diminished. If the techniques and equipment had been fixed as the way to ski, far less progress would have been made.

6. According to Herbsleb, et al., it takes about two years per level to move between CMM 1 and CMM 3 and the cost of software process improvement ranges from almost $500 to $2,000 per person per year [4]. There is no meaningful data for the higher levels, but given the ambitiousness of the goals and the fact that each level includes all the activities of the previous levels, it seems unlikely that moving to levels 4 and 5, even if possible for all organizations, will take much less time or money.

7. We think there are also questionable assumptions about the ability of people to master the continually increasing levels and quantities of new technology. For an organization that is truly chaotic and has little exposure to newer technology, the learning curve is steep indeed. A group getting its first exposure to object-oriented development or framework technology will have trouble simultaneously assimilating much of the background of software process improvement.

Additionally, training adds to the cost of each new technology acquired. New group members have a higher learning curve and consequently higher training cost at each new level. Moreover, regardless of the rosy picture suggested in the CMM level 5 actions, technology transfer means changing technology, and change means reduced effectiveness during adoption.

8. Most SPI studies are based on the analysis of one or a few data points. Herbsleb, et al., point out that we have only a few data points about the actual cost of SPI programs [4]. That article's representation of improvements such as productivity gain per year, time to market, post-release defects (reduction per year) and business value ratio are each based on so few data points that the positive results can only be thought of as tantalizing possibilities rather than established facts.

9. Small- and medium-sized companies have been put off by the presentation of software assessment and SPI. There is no question assessment can be costly, and for a small, lean organization, the overhead involved in meeting and verifying CMM or ISO 9000 criteria can be prohibitive. ISO 9000, for example, lists dozens of conditions and types of documentation required for certification. For these groups, if ISO 9000 certification is required, they may elect to buy an off-the-shelf solution that meets the letter but not the spirit of the requirements. As Bamberger notes, the way CMM is presented often makes smaller organizations believe it offers them no value [3]. Moreover, smaller organizations cannot afford the two- to three-year duration it normally takes to reach CMM level 3. If CMM certification becomes required for contract awards to subcontractors, then rather than fostering

improvement, it will more likely become a meaningless regulation.

Conclusions

Process improvement, then, must be tailored to the organization and with the goal of improving systems, but doing assessment to measure against CMM standards distracts from such a goal. The best part of assessment with respect to various standards is that a smart organization can use the assessment as a framework to evaluate how projects are done. And by conscious analysis rather than slavish adherence, the organization can plan and take steps that will improve its operation. We are also concerned that the focus on the existence of processes, at least in the way that many people apply CMM and ISO 9000 assessments, tends to over-value the technique at the expense of the goal. Processes are tools that help with solutions rather than solutions themselves [5]. C

References

1. Bach, J. Enough about process: What we need are heroes. *IEEE Softw. 12*, 2 (Feb. 1995), 96–98.
2. El Emam, K. and Briand, L. Costs and benefits of software process improvement. In *Better Software Practice for Business Benefit: Principles and Experience*. IEEE Computer Society Press, Washington, DC, 1997.
3. Bamberger J. Essense of the capability maturity model. *IEEE Comput. 30*, 6 (Jun. 1997), 112–114.
4. Herbsleb, J., et. al. Software quality and thecapability maturity model. *Commun. ACM 40*, 6, (Jun. 1997), 30–40.
5. Fayad, M.E. and Laitinen, M. *Transition to Object-Oriented Software Development*. Wiley, New York, to appear.

Mohamed E. Fayad (fayad@cs.unr.edu) is an associate professor at the University of Nevada.

Mauri Laitinen (mdl@sierra.net) is a principal in Laitinen Consulting at Lake Tahoe, Calif.

ACM 0002-0782/97/1100 $3.50