CHAPTER 20

Organizational Maturity: The Elephant Affecting Productivity

Bill Curtis, CAST Software, USA

The maturity of an organization's software development environment impacts the productivity of its developers and their teams [5]. Consequently, organizational attributes should be measured and factored into estimates of cost, schedule, and quality. This chapter presents an evolutionary model of organizational maturity, how the model can guide productivity and quality improvements, and how its practices can be adapted to evolving development methods.

Background

While working on improving software development at IBM in the 1980s, Watts Humphrey took Phil Crosby's course on quality management that included a maturity model for improving quality practices [1]. Crosby's model listed five stages of improvement through which a collection of quality practices should progress. While traveling home, Humphrey realized that Crosby's model would not work because it resembled approaches used for decades with little sustainable success. He realized past improvement efforts died when managers and developers sacrificed improved practices under the duress of unachievable development schedules. Until he fixed the primary problems facing projects, productivity improvements and quality practices had little chance to succeed.

During the late 1980s, Humphrey developed an initial formulation of his Process Maturity Framework [6] in the Software Engineering Institute at Carnegie Mellon University. In the early 1990s Mark Paulk, Charles Weber, and I transformed this framework into the Capability Maturity Model for Software (CMM) [10]. Since then the CMM has guided successful productivity and quality improvement programs in many software organizations globally. An organization's maturity level is appraised in process assessments led by authorized lead assessors.

Analyzing data from CMM-based improvement programs in 14 companies, James Herbsleb and his colleagues [5] found a median annual productivity improvement of 35 percent, ranging from 9 percent to 67 percent across companies. Accompanying this improvement was a median 22 percent increase in defects found prior to testing, a median reduction of 39 percent in field incidents, and a median reduction in delivery time of 19 percent. Based on cost savings during development, these improvement programs achieved a median return on investment of 5 to 1. How were these results achieved?

The Process Maturity Framework

The Process Maturity Framework has evolved over the past 30 years while sustaining its basic structure. As described in Table 20-1, this framework consists of five maturity levels, each representing a plateau of organizational capability in software development on which more advanced practices can be built. Humphrey believed that to improve productivity, impediments to sound development practices should be removed in a specific order. For instance, level 1 describes organizations with inconsistent or missing development practices. Too often crisis-driven projects rely on heroic efforts from developers who work nights and weekends to meet ridiculous schedules. Until project commitments and baselines can be stabilized, developers are trapped into working too fast, making mistakes, and having little time to correct them.

Table 20-1. *Process Maturity Framework*

Maturity Level	Attributes
Level 5 – Innovating <i>CMMI – Optimizing</i>	<ul style="list-style-type: none"> • Performance gaps needing innovative improvements identified • Innovative technologies and practices continually investigated • Experiments conducted to evaluate innovation effectiveness • Successful innovations deployed as standard practices
Level 4 – Optimized <i>CMMI – Quantitatively Managed</i>	<ul style="list-style-type: none"> • Projects managed using in-process measures and statistics • Causes of variation are managed to improve predictability • Root causes of quality problems are analyzed and eliminated • Standardized processes enable reuse and lean practices
Level 3 – Standardized <i>CMMI – Defined</i>	<ul style="list-style-type: none"> • Development processes standardized from successful practices • Standard processes and measures tailored to project conditions • Project artifacts and measures are retained, and lessons shared • Organization-wide training is implemented
Level 2 – Stabilized <i>CMMI – Managed</i>	<ul style="list-style-type: none"> • Managers balance commitments with resources and schedule • Changes to requirements and product baselines are managed • Measures are implemented for planning and managing projects • Developers can repeat sound practices in stable environments
Level 1 – Inconsistent <i>CMMI – Initial</i>	<ul style="list-style-type: none"> • Development practices are inconsistent and often missing • Commitments are often not balanced with resources and time • Poor control over changes to requirements or product baselines • Many projects depend on unsustainable heroic effort

The path to improvement begins when project managers or team leaders stabilize the project environment by planning and controlling commitments, in addition to establishing baseline and change controls on requirements and deliverable products. Only when development schedules are achievable and product baselines stable can developers work in an orderly, professional manner. Achieving level 2 does not force consistent methods and practices across the organization. Rather, each project adopts the practices and measures needed to create achievable plans and rebalance commitments when the inevitable requirements or project changes occur. When unachievable commitments are demanded by higher management or customers,

level 2 managers and team leaders learn to say “no” or diplomatically negotiate altered and achievable commitments.

Once projects are stable, the standard development processes and measures that characterize level 3 can be synthesized across the organization from practices and measures that have proven successful on projects. Implementation guidelines are developed from past experience to tailor practices for different project conditions. Standard practices transform a team/project culture at level 2 into an organizational culture at level 3 that enables an economy of scale. CMM lead assessors often report that standard processes are most frequently defended by developers because they improved productivity and quality and made transitioning between projects much easier.

Once standardized processes and measures have been implemented, projects can use more granular in-process measures to manage the performance of development practices and the quality of their products across the development cycle. Process analytics that characterize level 4 are used to optimize performance, reduce variation, enable earlier adjustments to unexpected issues, and improve prediction of project outcomes. Standardized development practices establish a foundation on which other productivity improvements such as component reuse and lean practices can be implemented [7].

Even when optimized to their full capability, processes may not achieve the productivity and quality levels required in a competitive environment or for demanding requirements. Consequently, organization must identify and evaluate innovations in technology, processes, workforce practices, etc., that can dramatically improve productivity and quality outcomes beyond existing performance levels. At level 5, the organization moves into a continuous innovation loop driven by specific targets for improvement that will change over time.

The Process Maturity Framework can be applied to individual processes—the so-called continuous approach. However, this framework is most effective when applied as a unique guidebook for organizational change and development. If the organization does not change, individual best practices typically will not survive the stress of crisis-driven challenges. This approach is consistent with observations on organizational systems in exceptionally successful businesses described in Jim Collin’s books *Built to Last* and *Good to Great*.

The Impact of Maturity on Productivity and Quality

One of the earliest and best empirical studies of a maturity-based process improvement program was reported by Raytheon [2, 4, 8]. Raytheon’s time reporting system collected data in effort categories drawn from a cost of quality model designed to show how improvements in product quality increased productivity and reduced costs. This model divided effort into four categories:

- Original design and development work
- Rework to correct defects and retest the system
- Effort devoted to first-run testing and other quality assurance activities
- Effort in training, improvement, and process assurance to prevent quality problems

Over the course of their improvement program (Table 20-2), Raytheon reported that the percentage of original development work increased from only a third of the effort at level 1 to just over half at level 2, two-thirds at level 3, and three-quarters at level 4. At the same time, rework was cut in half at level 2 and declined by a factor of almost 7 at level 4. As they achieved level 4, Raytheon reported that productivity had grown by a factor of 4 from the level 1 baseline.

Table 20-2. *Raytheon’s Distribution of Work Effort by CMM Level*

Year	CMM Level	Percent of total effort				Productivity growth
		Original work	Rework	First-run tests	Prevention	
1988	1	34%	41%	15%	7%	baseline
1990	2	55%	18%	13%	12%	1.5 X
1992	3	66%	11%	23%		2.5 X
1994	4	76%	6%	18%		4.0 X

Note 1: Table 20-2 was synthesized from data reported in Dion [2], Haley [4], and Lyndon [8]. *Note 2:* Effort for first-run tests and prevention were collapsed into one category in 1992. *Note 3:* Productivity growth is in factors compared to the 1988 baseline.

As evident in these data, productivity was heavily affected by the amount of rework. The proportion of rework is usually high prior to initiating an improvement program, with reports of 41 percent at Raytheon, 30 percent at TRW [14], 40 percent at NASA [15], and 33 percent at Hewlett Packard [3]. Stabilizing baselines and commitments enabled developers to work in a more disciplined, professional manner, reducing mistakes and rework and thereby improving productivity. The amount of initial testing stayed the roughly the same, while the retesting required after fixing mistakes declined. The extra effort devoted to the improvement program (prevention) was more than offset by reduced rework. Accompanying productivity growth was a 40 percent reduction in development costs per line of code by level 3.

The size of Raytheon's productivity growth in moving from level 3 to level 4 is difficult to explain from quantitative management practices alone. Further investigation revealed a reuse program that reduced the effort required to develop systems. Corroborating results on the productivity impact of reuse at level 4 were reported by Omron [11] and Boeing Computer Services [13]. Standardized processes at level 3 appear to create the necessary foundation of rigorous development practices and trusted quality outcomes needed to convince developers it is quicker to reuse existing components than develop new ones.

Updating Maturity Practices for an Agile-DevOps Environment

In the early 2000s the U.S. Department of Defense and aerospace community expanded the CMM to include system engineering practices. The new architecture of the Capability Maturity Model Integration (CMMI) dramatically increased the number of practices and reflected the ethos of large defense programs. In the opinion of many, including some authors of the original CMM, CMMI was bloated and required excessive practices for many software development environments that occasionally bordered on bureaucracy. At the same time, the rapid iterations of agile methods were replacing lengthy development practices that were insufficient to handle the pace of change affecting most businesses.

In theory, agile methods solve the level 1 commitment problem by freezing the number stories to be developed at the beginning of a sprint. New stories can only be added during the planning of a subsequent sprint. Consequently, it was disconcerting to hear developers at the Agile Alliance conferences in 2011 and 2012 complain about

stories being added during the middle of sprints at the request of marketing or business units. These in-sprint additions created the same rework-inducing schedule pressures that had plagued low maturity waterfall projects. Enforcing controls on commitments is a critical attribute of level 2 to protect developers from chaotic circumstances that degrade the productivity and quality of their work.

In a session at the Agile Alliance Conference in 2012, Jeff Sutherland, one of the creators of the Scrum method, commented that perhaps as many as 70 percent of the companies he visited were performing scrumbut. “We are doing Scrum, *buut* we don’t do daily builds, *buut* we don’t do daily standups, *buut* we don’t do...” As Jeff observed, they clearly weren’t doing Scrum. When performed rigorously across an organization’s development teams, Scrum and other agile or DevOps methods can provide the benefits of standardized processes characteristic of a level 3 capability. However, when these methods lack discipline, development teams are exposed to the typical level 1 problems of uncontrolled baselines and commitments, as well as patchy development practices that sap their productivity.

In 2015 Fannie Mae, a provider of liquidity for mortgages in the U.S. housing market, initiated a disciplined agile-DevOps transformation across their entire IT organization [12]. The transformation involved replacing traditional waterfall processes with short agile sprints and installing a DevOps tool chain with integrated analytics. Although they did not use CMMI, their improvement program mirrored a maturity progression from stabilizing changes on projects (level 2) to synthesizing standard practices, tools, and measures across the organization (level 3). Productivity was measured using Automated Function Points [11] delivered per unit of time and was tracked to monitor progress and evaluate practices.

After the transformation was deployed organization-wide, Fannie Mae found that the density of defects in applications had decreased by typically 30 percent to 48 percent. Productivity gains attributed to the transformation had to be calculated by collating data across several sprints whose combined duration and effort were comparable to previous waterfall release cycles (the baseline). The initial sprints were often less productive while the team adjusted to short-cycle development methods. However, when combined with results from several succeeding sprints, the average productivity was found to have increased by an average of 28 percent across applications compared to the waterfall baseline.

Summary

Improvement programs based on the Process Maturity Framework have improved productivity in software development organizations globally. Practices are implemented in evolutionary stages, each of which creates a foundation for more sophisticated practices at the next maturity level. Although development methods evolve over time, many of the problems that reduce their effectiveness are similar across generations. Thus, the maturity progression of Stabilize–Standardize–Optimize–Innovate provides an approach to improving productivity that is relevant to agile-DevOps transformations.

Key Ideas

The following are the key ideas from the chapter:

- Immature, undisciplined development practices can severely constrain productivity.
- Staged evolutionary improvements in an organizations' development practices can dramatically increase productivity.
- Modern development practices can suffer from weaknesses that hindered the productivity of earlier development methods.

References

- [1] Crosby, P. (1979). *Quality Is Free*. New York: McGraw-Hill.
- [2] Dion, R. (1993). Process improvement and the corporate balance sheet. *IEEE Software*, 10 (4), 28–35.
- [3] Duncker, R. (1992). *Proceedings of the 25th Annual Conference of the Singapore Computer Society*. Singapore: November 1992.
- [4] Haley, T., Ireland, B., Wojtaszek, E., Nash, D., & Dion, R. (1995). *Raytheon Electronic Systems Experience in Software Process Improvement (Tech. Rep. CMU/SEI-95-TR-017)*. Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

- [5] Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997). Software Quality and the Capability Maturity Model. *Communications of the ACM*, 40 (6), 30–40.
- [6] Humphrey, W. S. (1989). *Managing the Software Process*. Reading, MA: Addison-Wesley.
- [7] Liker, J. K. (2004). *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer*. New York: McGraw-Hill.
- [8] Lydon, T. (1995). Productivity drivers: Process and capital. *In Proceedings of the 1995 SEPG Conference*. Pittsburgh: Software Engineering Institute, Carnegie Mellon University.
- [9] Object Management Group (2014). *Automated Function Points*. www.omg.org/spec/AFP.
- [10] Paulk, M. C., Weber, C. V., Curtis, B., & Chrissis, M. B. (1995). *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA: Addison-Wesley.
- [11] Sakamoto, K., Kishida, K., & Nakakoji, K. (1996). Cultural adaptation of the CMM. In Fuggetta, A. & Wolf, A. (Eds.), *Software Process*. Chichester, UK: Wiley, 137–154.
- [12] Snyder, B. & Curtis, B. (2018). Using analytics to drive improvement during an Agile- DevOps transformation. *IEEE Software*, 35 (1), 78–83.
- [13] Vu, J. D. (1996). Software process improvement: A business case. *In Proceedings of the European SEPG Conference*. Milton Keynes, UK: European Software Process Improvement Foundation.
- [14] Barry W. Boehm (1987). Improving Software Productivity. *IEEE Computer*. 20(9): 43-57.
- [15] Frank McGarry (1987). Results from the Software Engineering Laboratory. *Proceedings of the Twelfth Annual Software Engineering Workshop*. Greenbelt, MD: NASA.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.