

CHAPTER 1

INTRODUCTION

Now more than ever, companies want to deliver products and services better, faster, and cheaper. At the same time, in the high-technology environment of the twenty-first century, nearly all organizations have found themselves building increasingly complex products and services. It is unusual today for a single organization to develop all the components that compose a complex product or service. More commonly, some components are built in-house and some are acquired; then all the components are integrated into the final product or service. Organizations must be able to manage and control this complex development and maintenance process.

The problems these organizations address today involve enterprise-wide solutions that require an integrated approach. Effective management of organizational assets is critical to business success. In essence, these organizations are product and service developers that need a way to manage their development activities as part of achieving their business objectives.

In the current marketplace, maturity models, standards, methodologies, and guidelines exist that can help an organization improve the way it does business. However, most available improvement approaches focus on a specific part of the business and do not take a systemic approach to the problems that most organizations are facing. By focusing on improving one area of a business, these models have unfortunately perpetuated the stovepipes and barriers that exist in organizations.

CMMI for Development (CMMI-DEV) provides an opportunity to avoid or eliminate these stovepipes and barriers. CMMI for Development consists of best practices that address development activities applied to products and services. It addresses practices that cover the product's lifecycle from conception through delivery and maintenance.

The emphasis is on the work necessary to build and maintain the total product.

CMMI-DEV contains 22 process areas. Of those process areas, 16 are core process areas, 1 is a shared process area, and 5 are development specific process areas.¹

All CMMI-DEV model practices focus on the activities of the developer organization. Five process areas focus on practices specific to development: addressing requirements development, technical solution, product integration, verification, and validation.

About Process Improvement

In its research to help organizations to develop and maintain quality products and services, the Software Engineering Institute (SEI) has found several dimensions that an organization can focus on to improve its business. Figure 1.1 illustrates the three critical dimensions that organizations typically focus on: people, procedures and methods, and tools and equipment.

What holds everything together? It is the processes used in your organization. Processes allow you to align the way you do business.

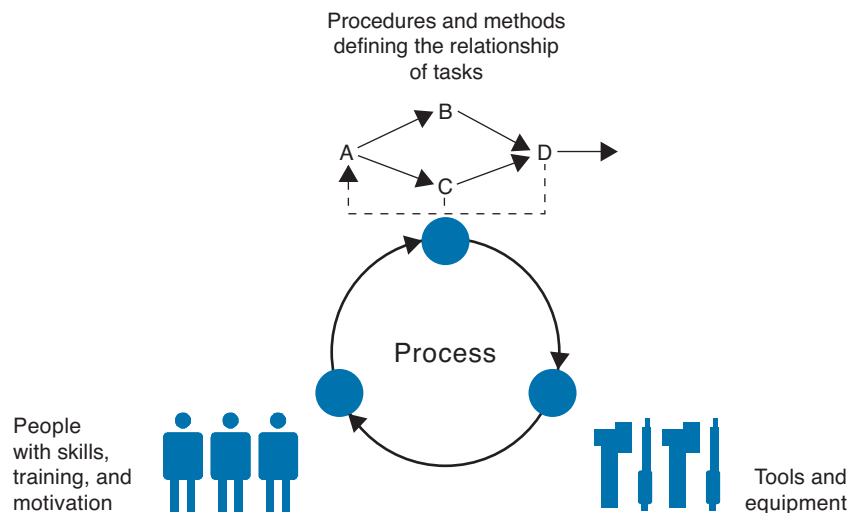


FIGURE 1.1
The Three Critical Dimensions

1. A core process area is a process area that is common to all CMMI models. A shared process area is shared by at least two CMMI models, but not all of them.

They allow you to address scalability and provide a way to incorporate knowledge of how to do things better. Processes allow you to leverage your resources and to examine business trends.

This is not to say that people and technology are not important. We are living in a world where technology is changing at an incredible speed. Similarly, people typically work for many companies throughout their careers. We live in a dynamic world. A focus on process provides the infrastructure and stability necessary to deal with an ever-changing world and to maximize the productivity of people and the use of technology to be competitive.

Manufacturing has long recognized the importance of process effectiveness and efficiency. Today, many organizations in manufacturing and service industries recognize the importance of quality processes. Process helps an organization's workforce to meet business objectives by helping them to work smarter, not harder, and with improved consistency. Effective processes also provide a vehicle for introducing and using new technology in a way that best meets the business objectives of the organization.

Looking Ahead

by Watts Humphrey

Nearly 25 years ago when we first started the maturity model work that led to the CMM and CMMI, we took a diagnostic approach. What are the characteristics of organizations that performed good software work? We were following the principle that Tolstoy stated in *Anna Karenina*.²

"Happy families are all alike; every unhappy family is unhappy in its own way."

In applying the Tolstoy principle to software organizations, we looked for those markers that characterized effective software operations and then formed these characteristics into a maturity model. The logic for the maturity model was that, to do good work, organizations must do everything right. However, since no one could possibly fix all of their problems at once, our strategy was to determine what organizations were doing and compare that to what they

2. Leo Tolstoy, *Anna Karenina*, Modern Library, New York.

should be doing to be a “happy family.” Then, using the model as a guide, they should start fixing the omissions and mistakes in the order defined by the model.

Keep Doing the Good Things

Unfortunately, we were not sufficiently clear with our initial guidance, and some people felt that, if they were only trying to get to maturity level 2, they should not do things that were at levels 3, 4, and 5. That is *not* what we meant. Organizations should continue doing all of the “good” things they now are doing and only focus on the problem areas. The objective at level 2 is to address those level 2 things that are missing or inadequate, and fix them first. Then the organization should consider addressing the level 3, 4, and 5 gaps. Again, they should not stop doing any things that work.

The Logic for the Maturity Level 2

The logic that we followed in establishing the maturity levels was as follows. First, organizations cannot do good work, and they certainly cannot improve, if they are in a perpetual state of crisis. Therefore, the first improvement efforts should focus on those things that, if done poorly or not at all, will result in crises. These are planning, configuration management, requirements management, sub-contract management, quality assurance, and the like.

The Logic for Maturity Level 3

Second, after the crises are largely controlled and the organization is plan-driven rather than crisis-driven, the next step is learning. How can people learn from each other rather than having to learn from their own mistakes? Again, from Tolstoy’s principle, there is an infinite number of ways to fail, so improving by reacting to failures is a never-ending and fruitless struggle. The key is to find out what works best in the organization and to spread that across all groups. Therefore, maturity level 3 focuses on learning-oriented things like process definition, training, and defined ways to make decisions and evaluate alternatives.

The Logic for Maturity Levels 4 and 5

At level 4, the focus turns to quantitative management and quality control, and at level 5, the efforts include continuous improvement, technological innovations, and defect prevention. Unfortunately, we never included an explicit requirement in CMM or CMMI that high-maturity organizations must look outside of their own laboratories

to identify best industry practices. Then, after they find these practices, they should measure, evaluate, and prototype them to see if these practices would help improve their operations. This seemed like such an obvious step that we never explicitly required it. However, judging from the slow rate of adoption of new and well-proven software and systems development innovations, such a requirement should have been included in the model.

Continuous Improvement

At this point, of the many organizations that have been evaluated at CMMI level 5, too many have essentially stopped working on improvement. Their objective was to get to level 5 and they are there, so why should they keep improving? This is both an unfortunate and an unacceptable attitude. It is unfortunate because there are many new concepts and methods that these organizations would find beneficial, and it is unacceptable because the essence of level 5 is continuous improvement.

Unfortunately, many organizations have become entangled in the weeds of process improvement and have lost sight of the forest and even of the trees. Six Sigma is a powerful and enormously helpful statistically based method for improvement, but it is easy for people to become so enamored with these sophisticated methods that they lose sight of the objective. This is a mistake. The key is priorities and what will help organizations to improve their business performance. This is where external benchmarking and internal performance measures are needed. Use them to establish improvement priorities and then focus your improvement muscle on those areas that will substantially improve business performance.

Next Steps

While CMMI has been enormously successful, we have learned a great deal in the last 25 years, and there are now important new concepts and methods that were not available when we started. The key new concept concerns **knowledge work**. Peter Drucker, the leading management thinker of the twentieth century, defined knowledge work as work that is done with ideas and concepts rather than with things. While a great deal of today's technical work is knowledge work, large-scale knowledge work is a relatively new phenomenon. Except for software, until recently, the really big projects all concerned hardware systems. Now, however, software work pervades most parts of modern systems, and even the work of hardware designers more closely resembles that of software developers than traditional hardware bread boarding and manufacturing.

To properly manage this new kind of work, new methods are needed, and Drucker enunciated the key new management concept. This is that knowledge work can't be managed with traditional methods; the knowledge workers must manage themselves.³ The logic behind Drucker's view is compelling, but the story is too extensive to cover in this short perspective. However, there is a growing number of publications that describe knowledge work and how and why new management methods are needed.⁴

In summary, the key point is that software and complex systems development projects are large-scale knowledge work, and the reason such projects have long been troubled is that they have not been managed with suitable methods. The first method that has been designed to follow Drucker's knowledge-management principles is the Team Software Process (TSP), but there will almost certainly be more such methods in the future.

Using the TSP to guide software and systems development projects turns out to be highly effective, and TSP projects are typically delivered on schedule, within budget, and with substantially improved quality and productivity.⁵ To assist CMMI users in continuously improving their performance, the SEI has defined a new CMMI-based strategy and a family of practices to guide them in evaluating and piloting these methods. This method is called CMMI-AIM (Accelerated Improvement Method), and it is currently in use by a growing number of organizations.

Conclusions

As we continue refining our processes and methods to address the needs and practices of creative teams and people, new opportunities will keep showing up for broadening the scope of our processes and including new methods and technologies as they become available. Because many of these advances will be new to most users, users will need specific guidance on what these new methods are and how to best use them. The SEI strategy has been to provide this guidance for each new family of methods as it becomes available and is proven in practice.

3. Peter Drucker, Knowledge-Worker Productivity: the Biggest Challenge, *California Management Review*, Winter 1999, 41, 2, ABI/INFORM Global.

4. Watts S. Humphrey, "Why Can't We Manage Large Projects?" *CrossTalk*, July/August 2010, pp. 4–7; and Watts S. Humphrey and James W. Over, *Leadership, Teamwork, and Trust: Building a Competitive Software Capability*, Reading, MA: Addison Wesley, 2011.

5. Noopur Davis and Julia Mullaney, Team Software Process (TSP) in Practice, SEI Technical Report CMU/SEI-2003-TR-014, September 2003.

Two examples of such new methods are CMMI- and TSP-related guidance on how to develop secure systems and on how to architect complex systems. As we look to the future, there will be many more opportunities for improving the performance of our systems and software engineering work. The key is to couple these methods into a coherent improvement framework such as TSP-CMMI and to provide the explicit guidance organizations need to obtain the potential benefits of these new methods. To avoid chasing the latest fads, however, organizations should measure their own operations, evaluate where they stand relative to their leading peers and competitors, and focus on those improvements that will measurably improve their business performance.

About Capability Maturity Models

A Capability Maturity Model (CMM), including CMMI, is a simplified representation of the world. CMMs contain the essential elements of effective processes. These elements are based on the concepts developed by Crosby, Deming, Juran, and Humphrey.

In the 1930s, Walter Shewhart began work in process improvement with his principles of statistical quality control [Shewhart 1931]. These principles were refined by W. Edwards Deming [Deming 1986], Phillip Crosby [Crosby 1979], and Joseph Juran [Juran 1988]. Watts Humphrey, Ron Radice, and others extended these principles further and began applying them to software in their work at IBM (International Business Machines) and the SEI [Humphrey 1989]. Humphrey's book, *Managing the Software Process*, provides a description of the basic principles and concepts on which many of the Capability Maturity Models (CMMs) are based.

The SEI has taken the process management premise, "the quality of a system or product is highly influenced by the quality of the process used to develop and maintain it," and defined CMMs that embody this premise. The belief in this premise is seen worldwide in quality movements, as evidenced by the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) body of standards.

CMMs focus on improving processes in an organization. They contain the essential elements of effective processes for one or more disciplines and describe an evolutionary improvement path from ad hoc, immature processes to disciplined, mature processes with improved quality and effectiveness.

Like other CMMs, CMMI models provide guidance to use when developing processes. CMMI models are not processes or process descriptions. The actual processes used in an organization depend on many factors, including application domains and organization structure and size. In particular, the process areas of a CMMI model typically do not map one to one with the processes used in your organization.

The SEI created the first CMM designed for software organizations and published it in a book, *The Capability Maturity Model: Guidelines for Improving the Software Process* [SEI 1995].

Today, CMMI is an application of the principles introduced almost a century ago to this never-ending cycle of process improvement. The value of this process improvement approach has been confirmed over time. Organizations have experienced increased productivity and quality, improved cycle time, and more accurate and predictable schedules and budgets [Gibson 2006].

Evolution of CMMI

The CMM Integration project was formed to sort out the problem of using multiple CMMs. The combination of selected models into a single improvement framework was intended for use by organizations in their pursuit of enterprise-wide process improvement.

Developing a set of integrated models involved more than simply combining existing model materials. Using processes that promote consensus, the CMMI Product Team built a framework that accommodates multiple constellations.

The first model to be developed was the CMMI for Development model (then simply called “CMMI”). Figure 1.2 illustrates the models that led to CMMI Version 1.3.

Initially, CMMI was one model that combined three source models: the *Capability Maturity Model for Software* (SW-CMM) v2.0 draft C, the *Systems Engineering Capability Model* (SECM) [EIA 2002a], and the *Integrated Product Development Capability Maturity Model* (IPD-CMM) v0.98.

These three source models were selected because of their successful adoption or promising approach to improving processes in an organization.

The first CMMI model (V1.02) was designed for use by development organizations in their pursuit of enterprise-wide process improvement. It was released in 2000. Two years later Version 1.1 was released and four years after that, Version 1.2 was released.

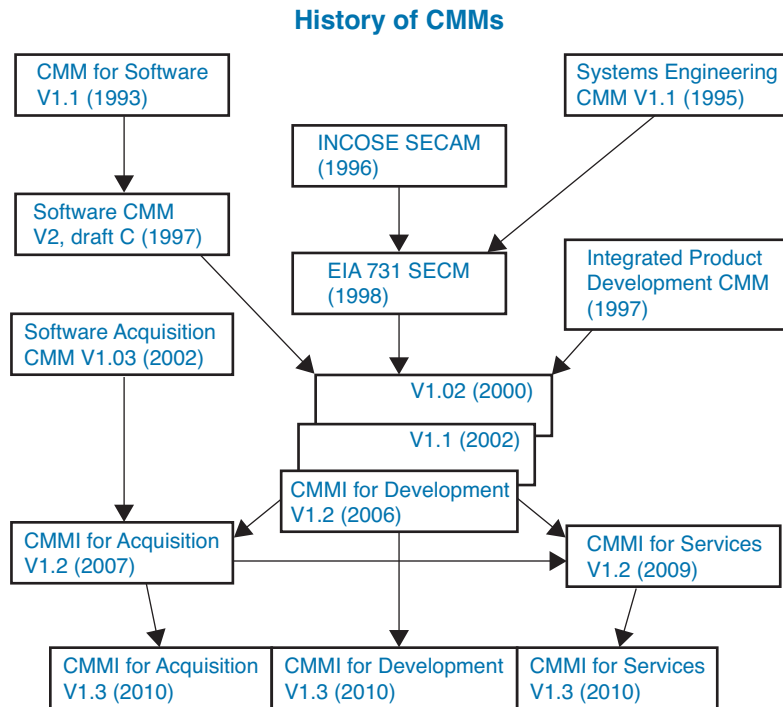


FIGURE 1.2
The History of CMMs⁶

By the time that Version 1.2 was released, two other CMMI models were being planned. Because of this planned expansion, the name of the first CMMI model had to change to become CMMI for Development and the concept of constellations was created.

The CMMI for Acquisition model was released in 2007. Since it built on the CMMI for Development Version 1.2 model, it also was named Version 1.2. Two years later the CMMI for Services model was released. It built on the other two models and also was named Version 1.2.

In 2008 plans were drawn to begin developing Version 1.3, which would ensure consistency among all three models and improve high maturity material in all of the models. Version 1.3 of CMMI for Acquisition [Gallagher 2011, SEI 2010b], CMMI for Development [Chrissis 2011, SEI 2010c], and CMMI for Services [Forrester 2011, SEI 2010a] were released in November 2010.

6. EIA 731 SECM is the Electronic Industries Alliance standard 731, or the Systems Engineering Capability Model. INCOSE SECAM is International Council on Systems Engineering Systems Engineering Capability Assessment Model [EIA 2002a].

CMMI: Integration and Improvement Continues

by Bob Rassa

CMMI is almost 15 years old, and has clearly become the world-wide de facto standard for process improvement in the development of systems, including systems engineering, software engineering, design engineering, subcontractor management, and program management. Since the release of CMMI V1.2 (for Development) almost 5 years ago, CMMI has embraced process improvement for Acquisition as well as the delivery of Services.

The full product suite of CMMI-DEV, CMMI-ACQ, and CMMI-SVC covers the complete spectrum of process improvement for the entire business, including commercial and defense industry, governments, and even military organizations. After the initial release of CMMI in November 2000, well over 1,000 Class A appraisals were reported in just four years—very successful numbers by our measures at that time; whereas recently almost 1,400 Class A appraisals were conducted in 2009 alone—quite a significant improvement.

As of January 2006, more than 45,000 individuals had received Introduction to CMMI training. As of July 2010, that number has exceeded more than 117,000 students.

CMMI-DEV has been translated into Japanese, Chinese, French, German, Spanish, and Portuguese. Translation of CMMI-SVC into Arabic is beginning. The success in CMMI recognition and adoption worldwide is undeniable.

The CMMI V1.2 architecture was altered slightly to accommodate two additional CMMI constellations, which we designated CMMI-ACQ (CMMI for Acquisition) and CMMI-SVC (CMMI for Services). CMMI V1.3 focuses on providing some degree of simplification as well as adding more integrity to the overall product suite. V1.3 model improvements have a heavy concentration on the high maturity aspects embodied in levels 4 and 5, in both the model structure as well as the appraisal method.

We learned that there were certain ambiguities within the V1.2 product suite, and the areas affected are now clarified in V1.3 to achieve greater consistency in overall model deployment and appraisal conduct of CMMI. The criteria that are used in the appraisal audit process, which was implemented in 2008, have now been incorporated in the product suite where appropriate. We have also provided clarification on the sampling of “focus programs” in

the appraised organization to reduce the complexity and time involved in conducting Class A appraisals, thereby reducing the cost of implementing CMMI.

It has been noted by some that CMMI is only for large organizations, but the data tells a different story. In fact, a large number of small organizations have been appraised and have told us that they reap benefits of CMMI far beyond the investment. A comprehensive Benefits of CMMI report is now on the website of the designated CMMI Steward, the Software Engineering Institute of Carnegie Mellon University (<http://www.sei.cmu.edu/cmmi>). This report, essentially a compendium of real benefits provided by users, clearly shows positive effects such as reduced defects on delivery, reduced time to identify defects, and more. The data tells us that CMMI is truly state-of-the-art in-process improvement, and the substantive benefits reported confirm this.

However, to be truly effective, CMMI must be applied conscientiously within the organization. When we started the initial development of CMMI, it was well-publicized that its purpose was to integrate the divergent maturity models that existed at the time. We soon realized that the real purpose that should have been communicated as the ultimate benefit of CMMI was that this integrated model would integrate the design and management disciplines in terms of both process and performance.

To achieve this ultimate benefit, care is needed to ensure that integrated processes are put into place within the organization, that such processes are implemented across the enterprise on all new programs and projects, and that such implementation is done in a thorough manner to assure that new programs start out on the right foot.

This book provides the latest expert and detailed guidance for effective CMMI implementation. It covers all the specifics of V1.3 and addresses nuances of interpretation as well as expert advice useful to the new and experienced practitioner.

Hundreds of process improvement experts have contributed to the overall CMMI development and update, and many of them contributed their expertise to this volume for the benefit of the worldwide user community. We trust you will enjoy their work and find it useful as you continue your journey along the path of continuous process improvement.

Remember, great designers and great managers will still likely fail without a proven process framework, and this is what CMMI provides.

CMMI Framework

The CMMI Framework provides the structure needed to produce CMMI models, training, and appraisal components. To allow the use of multiple models within the CMMI Framework, model components are classified as either common to all CMMI models or applicable to a specific model. The common material is called the “CMMI Model Foundation” or “CMF.”

The components of the CMF are part of every model generated from the CMMI Framework. Those components are combined with material applicable to an area of interest (e.g., acquisition, development, services) to produce a model.

A “constellation” is defined as a collection of CMMI components that are used to construct models, training materials, and appraisal related documents for an area of interest (e.g., acquisition, development, services). The Development constellation’s model is called “CMMI for Development” or “CMMI-DEV.”

The Architecture of the CMMI Framework

by Roger Bate

with a postscript by Mike Konrad

Over the years, as the CMMI Product Suite has been used in disparate industries and organizations, it became apparent that CMMI could be applied to all kinds of product development, especially if the terminology was kept general for similar practices.

A further revelation was that the process and project management practices of the model are suitable for a wide range of activities besides product development. This discovery led me to propose that we should enable the expansion of CMMI, including the extension of the scope of CMMI, by creating a new architecture for the CMMI Framework.

This new architecture would accommodate other areas of interest (e.g., Services and Acquisition). I was musing one day about the valuable best practices that were contained in models. I began to think of them as the *stars* of process improvement. I pushed this metaphor a little further to call the collection of components that would be useful in building a model, its training materials, and appraisal documents for an area of interest a *constellation*. This was the beginning of the architecture that was eventually created.

There are two primary objectives for the CMMI Framework architecture:

- Enable the coverage of selected areas of interest to make useful and effective processes.
- Promote maximum commonality of goals and practices across models, training materials, and appraisal methods.

These objectives pull in opposite directions; therefore, the architecture was designed as a bit of a compromise.

The CMMI Framework will be used in the future to accommodate additional content that the user community indicates is desirable. The framework contains components used to construct models and their corresponding training and appraisal materials. The framework is organized so that the models constructed will benefit from common terminology and common practices that have proven to be valuable in previous models.

The CMMI Framework is a collection of all model components, training material components, and appraisal components. These components are organized into groupings, called constellations, which facilitate construction of approved models and preserve the legacy of existing CMM and CMMI models.

In the framework, there are constellations of components that are used to construct models in an area of interest (e.g., Acquisition, Development, and Services). Also in the framework, there is a CMMI model foundation. This foundation is a skeleton model that contains the core model components in a CMMI model structure. The content of the CMMI model foundation is apropos to all areas of interest addressed by the constellations. A CMMI model for a constellation is constructed by inserting additional model components into the CMMI model foundation.

Because the CMMI architecture is designed to encourage preserving as much common material as is reasonable in a multiple constellation environment, the framework contains and controls all CMMI material that can be used to produce any constellation or model.

CMMI models have a defined structure. This structure is designed to provide familiar placement of model components of various constellations and versions. If you look at the structure of a process area, you'll see components including Process Area Name, Category, Maturity Level, Purpose, Introductory Notes, References, and Specific Goals. You will also find that every process area in this

model (i.e., CMMI for Development) and all other CMMI models produced from the CMMI Framework have the same structure. This feature helps you to understand quickly where to look for information in any CMMI model.

One of the benefits of having a common architecture and a large portion of common content in the various models is that the effort required to write models, train users, and appraise organizations is greatly reduced. The ability to add model components to the common process areas permits them to expand their scope of coverage to a greater variety of needs. In addition, whole new process areas may be added to provide greater coverage of different areas of interest in the constellations.

CMMI models have a great deal of well-tested content that can be used to guide the creation of high performance processes. The CMMI architecture permits that valuable content to continue to work in different areas of interest, while allowing for innovation and agility in responding to new needs.

You can see that CMMI has grown beyond the *star* practices of the three original source models to *constellations*. This expansion into the galaxy is only possible with a well-thought-out and designed architecture to support it. The CMMI architecture has been designed to provide such support and will grow as needed to continue into the future.

Postscript

Roger passed away in 2009, about two years after this perspective was written for the second edition of this book. Roger's grand vision of constellations and a flexible architecture that established a deep level of commonality among CMMI models was largely realized in the sequential release of the three V1.2 CMMI models published between 2006 and 2009.

One of many anecdotes that reveals Roger's transcending vision and greatness and reflects on the early days of CMMI is when I was contacted by friend and colleague Tomoo Matsubara-san in 1997. He asked me to provide a short opinion piece for his Soapbox column in *IEEE Software* magazine. At the time, we at the SEI were confronted with consternation from many Software CMM transition partners because they felt that we were abandoning the Software CMM product line. Instead, we were pursuing the convergence of the software engineering and systems engineering process improvement communities by creating a product line that both could use.

We hoped such a convergence would not only eliminate the need to separately maintain two models and associated product lines (those for the Software CMM and the EIA 731 Systems Engineering Capability Model), but also would encourage synergism within organizations addressing system-wide performance issues. To me, the Soapbox column provided an opportunity to communicate what we were hoping to accomplish with CMMI, so I suggested to Roger that he write something for Matsubara-san's column on why software engineering would benefit from a "systems engineering" perspective.

This chain of events led to Roger's Soapbox piece, "Do Systems Engineering? Who, Me?"⁷ This anecdote typifies Roger's brilliance, ability to span multiple disciplines, and talent for motivating multiple stakeholders with a unifying vision and a shared mission to change the organization's behavior and achieve superior performance.

As mentioned, Roger's vision was largely realized with the release of the Version 1.2 CMMI Product Suite. For V1.3, however, we confronted a challenge that required some modification to Roger's initial vision for the CMMI architecture.

The challenge was the term "project," which commonly refers to an endeavor whose purpose and end are marked by the delivery of something tangible (e.g., a product). This term appeared throughout many of the process areas (PAs) and was generally a good fit for CMMI-ACQ and CMMI-DEV but not for CMMI-SVC. After researching the problem, we devised a solution: for V1.3, we would allow the CMF to vary across constellations in a very limited way. CMF material in CMMI-DEV and CMMI-ACQ could continue to refer to "projects," whereas in CMMI-SVC, the reference would instead be to "work," "work activities," "services," or similar. In this way, the CMMI user community could continue to use common English terms in a familiar and consistent way while benefitting from a sharing of CMMI terminology and best practices (for 17 PAs) across a broad range of process improvement communities. (This idea was researched by exploratory implementation, a survey, and pilots; and it worked well.)

V1.3 benefits from many synergies. The sequential release of new constellations for V1.2 between 2006 and 2009 provided the opportunity to evaluate what CMF should be—one context at a time. However, in V1.3 development, model material originally proposed

7. Roger R. Bate, "Do Systems Engineering? Who, Me?" *IEEE Software*, vol. 15, no. 4, pp. 65–66, July/Aug. 1998, doi:10.1109/52.687947.

for one area of interest was sometimes extended to all three. Examples include the concept of quality attributes; prioritization of requirements; ontology for products and services; a richer set of examples for measurement and analysis; a sharper focus on what is essential to team performance (i.e., team composition, empowerment, and operational discipline); and alignment of high maturity practices with business objectives enabled by analytics. The V1.3 user may be unaware of these synergies or their sources, but may derive benefit from them.

With V1.3, we've only begun to explore ideas for the future of CMMI. Beyond V1.3, there are lots of promising directions forward, which is only the case because of Roger's original pioneering vision for CMMI. With Roger's passing, his Chief Architect mantle has been passed on to someone many years his junior (me) and I can assure our readers that well into his mid-80s, Roger remained a truly brilliant man, retaining his clear thinking and ear for nuance while maintaining a broad perspective when rendering an opinion; and so I recognize almost as much as anyone how big a space he has left behind. Although he was the one who conceived of the term constellations as a way of thinking about CMMI architecture, to us who knew him well, he was the real star of CMMI.

CMMI for Development

CMMI for Development is a reference model that covers activities for developing both products and services. Organizations from many industries, including aerospace, banking, computer hardware, software, defense, automobile manufacturing, and telecommunications, use CMMI for Development.

CMMI for Development contains practices that cover project management, process management, systems engineering, hardware engineering, software engineering, and other supporting processes used in development and maintenance.

Use professional judgment and common sense to interpret the model for your organization. That is, although the process areas described in this model depict behaviors considered best practices for most users, process areas and practices should be interpreted using an in-depth knowledge of CMMI-DEV, your organizational constraints, and your business environment.