Achieving CMMI Level 5 Improvements with MBASE and the CeBASE Method

Dr. Barry Boehm, Dr. Daniel Port, and Apurva Jain
University of Southern California

Dr. Victor Basili
University of Maryland

Tuesday, 30 April 2002
Track 1: 1:00 - 1:40
Ballroom A

Each branch of service in the Department of Defense has major initiatives to pursue more advanced software-intensive systems concepts involving network-centric warfare with self-adaptive networks and cooperating human and autonomous agents. The ability to balance discipline and flexibility is critically important to developing such highly dependable software-intensive systems in an environment of rapid change. Risk-management orientation enables users of Capability Maturity Model® Integration℠ (CMMI℠) to apply risk considerations to determine how much discipline and how much flexibility is enough in a given situation. The risk-driven nature of the spiral model and MBASE enables them to achieve a similar balance of discipline and flexibility. When these project-level approaches are combined with the organization-level approaches in the Experience Factory, the result is the unified Center for Empirically Based Software Engineering (CeBASE) method described in this article.

Recent events in Afghanistan have convincingly demonstrated the value of software and information technology in achieving military superiority. Each of the Department of Defense (DoD) services has major initiatives to pursue even more advanced software-intensive systems concepts involving network-centric warfare with self-adaptive networks and cooperating human and autonomous agents.

However, there are tremendous challenges in providing the software product and process capabilities necessary to realize these concepts. These challenges include security, scalability, interoperability, legacy systems transition, uncontrollable commercial off-the-shelf (COTS) products, and synchronizing dozens if not hundreds of independently evolving systems in a world of increasingly rapid change.

In particular, the processes for managing these complex systems of systems will require both highly disciplined methods to ensure dependable operations and highly flexible methods to adapt to change.

Fortunately, DoD's new 5000-series policies on evolutionary acquisition and spiral development provide the acquisition and program management framework to achieve this balance of discipline and flexibility. Also, the recent Capability Maturity Model® (CMM®) Integration℠ (CMMI℠) provides a development framework for integrating software and systems consideration with degrees of freedom for tailoring development processes to achieve appropriate balances of discipline and flexibility. However, these initiatives fall short of providing specific techniques for achieving and maintaining the right balance of discipline and flexibility for a particular program's evolving situation.

In our previous three CrossTalk articles, we provided some specific methods that programs can use to achieve this balance. In "Understanding the Spiral Model as a Tool For Evolutionary Acquisition [1]," we showed how appropriate use of the spiral model enables programs to achieve the flexibility needed for evolutionary acquisition, while applying risk-management principles to retain an appropriate level of program discipline.

"... the opportunity is here for other organizations to use the Experience Factory approach to achieve CMMI Level 5 benefits well before reaching Level 4."

In "Balancing Discipline and Flexibility with the Spiral Model and MBASE [2]," we showed how risk considerations could be used to realize appropriate but different process models for different program situations. We also elaborated on some of the specific practices in Model-Based (system) Architecting and Software Engineering (MBASE) such as the use of life-cycle anchor-point milestones to keep the program on track during its evolution.

In "Using the Spiral Model and MBASE to Generate New Acquisition Process Models: SAIV, CAIV and SCQAIV [3]," we showed how programs could use MBASE risk management techniques to avoid many overruns of fixed schedules and budgets. This is done by prioritizing desired features and inverting the development process to deliver the most important features within the available schedule or budget.

MBASE and the CeBASE Method

However, the spiral, MBASE, and Schedule as Independent Variable (SAIV) approaches all operate at the individual project level. This still leaves open the coverage of the counterpart CMMI organization-level process areas, particularly those of achieving continuous improvement of the organization's processes.

In this article, we show how MBASE has been integrated with the University of Maryland's (UMD) organization-level Quality Improvement Paradigm (QIP), Experience Factory (EF), and Goal-Question-Metric (GQM) approaches into a Center for Empirically Based Software Engineering (CeBASE) method, which successfully addresses these challenges. CeBASE is sponsored by the National Science Foundation, NASA, and the DoD, and jointly led by the UMD and the University of Southern California (USC).

As we explored the details of Maryland's QIP, EF, and GQM approaches and USC's MBASE approach, we found that they were expressing very similar principles and practices. The Spiral Model's initial focus on system objectives was consistent with the QIP's initial focus on organizational and project-specific goals expressed in

® Capability Maturity Model, CMM, Software Capability Maturity Model, and SW-CMM are registered in the U.S. Patent and Trademark Office.
℠ CMM Integration and CMMI are service marks of Carnegie Mellon University.

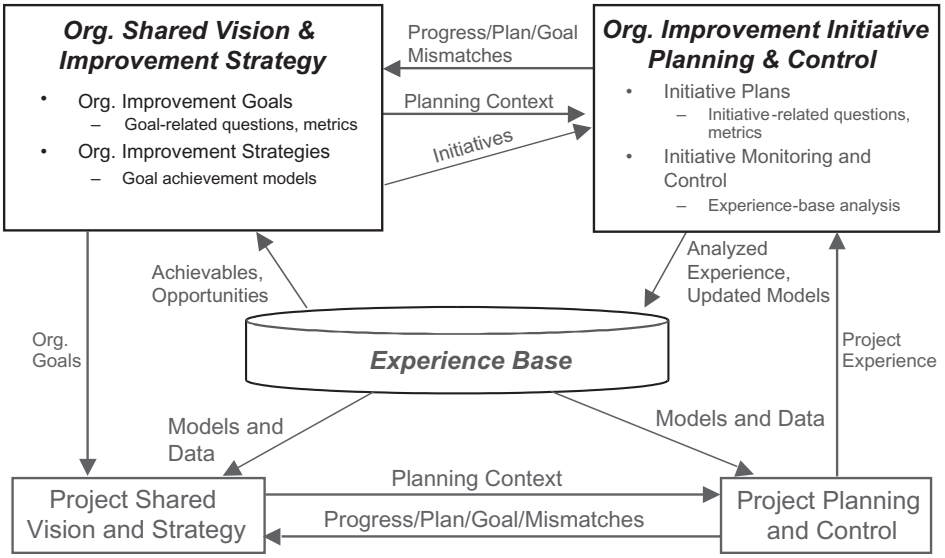


Figure 1: *Experience Factory Framework*

context using the GQM approach. The EF's focus on organizational learning to understand a system's operational stakeholders and their goals corresponds strongly with MBASE's stakeholder win-win approach to mutual stakeholder understanding and development of a shared system vision.

In the next section of this article, we summarize the key principles and practices of the QIP, the EF, and the GQM approaches; provide evidence of their successful application over 25 years of practice in the NASA Goddard-University of Maryland-Computer Science Corp. (CSC) Software Engineering Laboratory (SEL); and provide an example of their application at a systems as well as software level. In the section "The CeBASE method and CMMI," we present the CeBASE method and show how its process elements cover the process areas of the CMMI. In "Using the CeBASE Method," we show a version of the CeBASE method that has been successfully applied to more than 100 electronic services applications over six years' practice at USC.

Our conclusions include a diagram summarizing the process model distinctions among traditional approaches such as the Waterfall Model and Software Capability Maturity Model® (SW-CMM®); project-oriented approaches such as the spiral model, MBASE, and the Rational Unified Process (RUP); and integrated project/organization approaches such as the CMMI and CeBASE Method.

The QIP, GQM, and EF Approach

Framework and Methods

Since 1976, the UMD has been collaborating with NASA-Goddard and CSC on the SEL. The UMD and the SEL have developed and refined a series of closed-loop feedback processes that have resulted in significant improvements in software quality across more than 100 large software applications in the last 25 years.

The formulation of these feedback processes is called the QIP [4]. It uses six steps to provide an organized approach to continuous software quality improvement: 1) characterizing the organization, 2) setting goals, 3) choosing and instrumenting an appropriate process, 4) executing and monitoring the process, 5) analyzing the data to identify improvements, and 6) packaging the experience and improvements for future use.

The QIP makes use of the GQM approach, which is a mechanism for defining and evaluating a set of operational goals using measurement [5]. It ensures that your general goals are elaborated into specific questions and metrics for tracking progress and evaluating success, and that your people do not waste effort collecting and analyzing data weakly related to your goals.

The GQM approach can be applied at both the project level and the organization level. The EF [6] provides a consistent way of operating at both levels, as shown in Figure 1. Organization and project goals are determined by involving the relevant success-critical stakeholders in negotiating mutually satisfactory (win-win) and achievable goals.

For example, the organization may set a goal to reduce its projects' software cycle time by 50 percent. The initial implementing project may set goals and plans to have each project activity reduce its calendar time by 50 percent. As the project proceeds, its progress is monitored for progress/plan/goal mismatches, as shown at the bottom of Figure 1. While design, code, and test planning may finish in 50 percent less time, integration and test may start showing a 50 percent increase rather than decrease in duration. Analyzing this progress/plan/goal mismatch would determine the root cause to be delays due to inadequate test planning and preparation of test tools, test drivers, and test data. Further, shortening the test plan activity had produced no cycle timesaving, as test planning was not on the project's critical path.

The results of this analysis would be fed into the organization's experience base: Future cycle-time reduction strategies should focus on reducing the duration of critical path activities, and options for doing this include increasing the thoroughness and duration of noncritical-path activities. Overall then, as shown in the center of Figure 1, the EF analyzes and synthesizes such kinds of experience, acts as a repository for the experiences, and supplies relevant experience to the organization on demand. The EF packages experience by building informal and formal models and measures of various processes, products, and other forms of knowledge via people, documents, and automated support.

QIP, GQM, and EF in Practice

The application of the integrated set of these methods is referred to as the Experience Factory Organization, which resulted in a continuous improvement in software quality and cost reduction during the quarter-century life span of the SEL. When measured during three baseline periods in 1987, 1991, and 1995 (each representing about three years of development efforts), the demonstrated improvements included a 75 percent decrease in development defect rates from 1987 to 1991, and a 37 percent decrease from 1991 to 1995. We also observed a reduced development cost of 55 percent from 1987 to 1991 and of 42 percent from 1991 to 1995 [7, 8].

A more detailed example of improvement over time, in Figure 2, shows the defect rates in defects per thousand delivered lines of code (K-DLOC) for similar classes of projects at CSC during the application of the EF concepts. Over time, the defect models became well established and the range of variation (indicated by the upper and lower lines) narrowed, allowing managers to better manage quality [9]. Thus, the EF approach enabled the SEL portion of CSC to achieve SW-CMM Level 5 improvements well before CSC became a Level 5 organization in 1998. With the CMMI's emphasis on measurement and analysis as early as Level 2, the opportunity is here for other organizations to use the EF approach to achieve CMMI Level 5 benefits well before reaching Level 4.

Various EF concepts have been successfully applied in other organizations, including Daimler Chrysler, Robert Bosch, TRW, and Allianz.

Applying EF Concepts at the Systems Level

Systems-Level Goals and Questions

Many software organizations interpret the EF concepts at just the software level. They miss many opportunities to reap much more significant returns on investment at the systems level. For example, suppose that your software intensive project has a proposed goal for an improvement initiative to reduce the project's software defect rates. What should be your next step? Usually it would be to look down from the software goal into the software details. How will we define defect? What are the counting rules for overlapping defects? What are our current defect rates?

With EF at the systems level, your next step is to look upward and sideways from the software and ask system-level questions: Why do we want to reduce software defect rates? What system goals are being frustrated by software defects? Where are the frustrations the greatest?

For example, in an operational order-processing system, the answers may be that the software defects are causing 1) too much downtime in the operation's critical path, 2) too many defects in the system's operational plans, and 3) too many new-release operational problems.

These insights enable you to reformulate your improvement initiative goal to decrease the organization's software defect-related losses in operational cost effectiveness. Items one through three become initial high-payoff target sub-goals for the initiative. Given this new goal and context, what should be your next step?

Sub-Goal Level Questions, Models, and Metrics

Again, a good next step is to ask *why* the software defects are causing operational problems, often with the help of models. For example, Figure 3 shows a critical-path model for analyzing the order-processing downtime and delays caused by software defects. Analyzing this model may lead to several valuable insights, improvement strategies, and system payoffs:

1. Often, major sources of delay are additional manual processing delays caused by software or non-software problems, as with the Scientific American order processing system discussed in Boehm [10].
2. The logic for packaging and delivery scheduling can become quite complex when only part of an order is in stock. (It is generally okay to send a partial shipment at Amazon.com, however, not for jet engine repair spare parts.) Software defects can again cause considerable operational delays.
3. "Produce status reports" defects should not be on the operational critical path. This module was probably put on the critical path by a programmer's detailed design coupling and cohesion decision without considering its potential system effect, resulting in status-report defects causing order-delivery delays.
4. The overall legacy order-processing system may just be too slow and difficult to modify and should be replaced downstream by a new Web-based order-processing system. It is generally good to be asking *why not* as well and *why* questions.

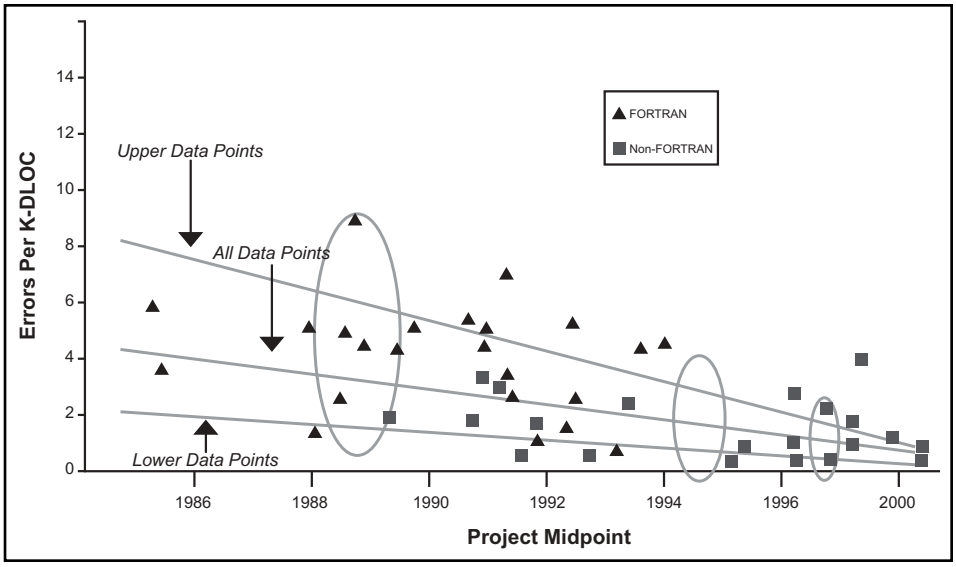


Figure 2: *Defect Rate Improvements in Software Engineering Laboratory/Computer Science Corporation Projects*

Putting It All Together

Each of these sub-goal-related initiatives needs to be monitored and controlled with respect to improvement-related metrics such as order-processing cycle time and user satisfaction. The results need to be integrated with other ongoing improvement initiatives to ensure synergy and integration with the overall organizational experience base discussed earlier in the "framework and methods" section. The sidebar on page 12 shows the resulting systems-level EF-GQM initiative steps.

These system-level EF-GQM approaches are already being practiced by leading-edge software organizations. Two of the recent Institute of Electrical and Electronics Engineers' Software Process Achievement Award winners, Advanced Information Services, Inc. (AIS) and Tinker Air Force Base, are good examples [11, 12].

AIS uses Balanced Scorecard techniques to integrate its software, systems, project, and organizational goals in such areas as customer satisfaction, financial performance, employee growth, process improvement, and organizational learning capability. Specifically, AIS periodically assesses its performance and rate of progress in these areas on a Balanced Scorecard form and uses the results to adjust its improvement efforts in each area. Tinker has used its software insights to stimulate systems-level initiatives with its counterpart hardware and test organizations to improve system-level cycle time and to deliver quality in such areas as B-2 Test Program Sets.

This kind of approach is what transitioning from the software CMM to the CMMI is all about. It requires software organizations to be more pro-active than reactive in interacting with the operational system stakeholders. It gets software people applying their necessary expertise to system issues. It results in much larger bottom-line payoffs for the operational system stakeholders. The next two sections discuss how the CeBASE method integrates software and system-level activities as well as project- and organizational-level activities and how its practices map to the process areas and practices in the CMMI.

Figure 3: *Order-Processing System Critical Path Model*

Systems-Level Experience Factory Goal-Question-Metric Initiative Steps

1. Identify a software-related improvement initiative goal.
2. Relate this to system-level goals: Ask questions about why the initiative is needed.
3. Use the results to identify the related system-level improvement initiative goal and high-payoff sub-goal initiatives.
4. Perform a systems-level root-cause analysis: Construct relevant models, ask questions about why the current-system shortfalls cause problems and whether or not to try alternative system approaches.
5. Identify the system improvement initiative's key stakeholders; achieve a shared vision of and commitment to the initiative goals and strategies.
6. Establish improvement initiative plans and progress metrics for each sub-goal initiative and the overall initiative.
7. Execute, monitor, and control the initiative plans with key stakeholder participation. Feed the resulting experiences into the organization's experience base for future benefits.

The CeBASE Method and the CMMI

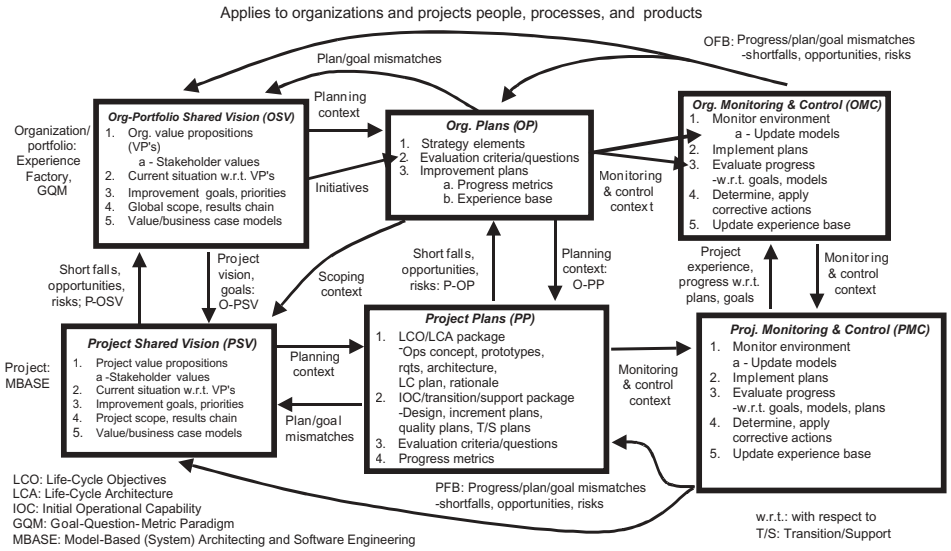
The CeBASE Method Framework

Overall, we found that both EF-GQM and MBASE could be integrated into a common CeBASE method. Its framework is organized around a trio of common strategic themes, shown by the vertical pairs in Figure 4. These three themes are the stakeholders' *shared vision* for the organization or project; risk-driven *plans* for process, product, and people; and continuous *monitoring and control*. As seen in Figure 4, these themes express both the operation of EF-GQM at the organizational level and the operation of MBASE-GQM at the project level. Within a large diverse organization, we may wish to consider a particular set of projects within a *portfolio* or product line of related products or services.

To start at the upper left of Figure 4, the organization's value propositions are often contained in an organizational mission statement. This will cover the organizational stakeholders' agreed-upon win conditions and will be expressed in terms of such Balanced Scorecard elements as customer satisfaction, financial performance, employee growth, process improvement, and organizational learning capability.

Improvement goals and priorities will come from Balanced Scorecard assessments. These might include such goals as reducing software development cycle time or reducing average order-delivery time. The specific quantitative goals, e.g., reduce software development cycle time by 50 percent, would be based on initial cost/value analyses. These are developed using such techniques as the DMR Consulting Group's Results Chains linking improvement initiatives to contributions and benefits-realized outcomes [13] and associated business-case models linking the value of benefits realized to the costs invested in the initiatives.

Figure 4: *The CeBASE Method Framework*



The CeBASE Method at the Organizational Level

The resulting organization (or portfolio) shared vision (OSV) (Figure 4) drives two sets of initiatives. Horizontally in Figure 4, it drives initiatives to improve software cycle time or to reduce order-delivery time across the organization. These initiatives will have strategy elements and their associated organization-level improvement plans (OP) such as reducing delays in order-delivery time due to software defects.

Following the GQM paradigm, the *goal* of reducing order-delivery time is related to organization plan *questions* such as, "What is our current record on delivery times?" "What distinguishes orders with significantly better or worse delivery times?" "What are the costs and benefits resulting from an improvement initiative?"

These questions are related to *metrics* such as overall order-delivery time, critical-path task times, costs and benefits of eliminating related software defects, and customer order-delivery satisfaction ratings. These are used to monitor and control organization-level progress (OMC), and to adjust the strategies and goals based on the organizational feedback of progress/plan/goal achievements and mismatches.

The CeBASE Method at the Project Level

Vertically in Figure 4, the OSV drives the nature of each project's shared vision (PSV) and its associated goals and priorities. Thus, for example, an organizational improvement goal to reduce software development cycle time by 50 percent will be reflected in the project's value propositions and improvement goals or the organization project shared vision (O-PSV) (see arrow in Figure 4). This will lead to project-level goals, models, questions, and metrics such as reducing the duration of each project task by 50 percent.

This in turn leads horizontally across the bottom of Figure 4 to project-level plans (PP), project monitoring and control activities (PMC), and to the determination and feedback of project-level progress/plan/goal mismatches. This mismatch feedback could be negative such as increased integration and test task durations or it could be positive. For example, the project might incorporate a new in-transit-visibility COTS package for order delivery tracking that both helps in delay diagnosis and improves customer satisfaction by answering questions about delivery delays.

This project feedback propagates upward to the organizational level along all three lines of traceability. The shortfalls or

opportunities with respect to organizational shared vision and goals are fed back along the project OSV (P-OSV) arrow at the left. The corresponding plan-context feedback occurs along the project OP (P-OP) arrow in the center, and the monitoring and control feedback at the right is used to update the organization's detailed experience base on best practices for achieving goals.

As a final observation, note that the content of the PP element consists of the Spiral/MBASE Life-Cycle Objectives (LCO), Life-Cycle Architecture (LCA), and Initial Operational Capability (IOC) anchorpoint milestone content that we discussed in our May 2001 and December 2001 CrossTalk articles [1, 2]. Thus, the Spiral/MBASE guidelines in those articles have become the project-level guidelines for the CeBASE method. A detailed example of these guidelines is shown next.

CeBASE Guidelines Example: Shared Vision

The CeBASE project-level and organization-level shared vision guidelines are quite similar. Their main difference is one of context: The project-level shared vision has the organization-level shared vision as context and shows traceability to it, but not vice versa. Figure 5 shows the table of contents and example text from the project-level shared vision guidelines. In the CeBASE method, it is the first item to be drafted by the project's Integrated Product Team of success-critical stakeholders or its equivalent. It sets the stage for subsequent Inception Phase prototyping and stakeholder win-win requirements negotiation.

The shared vision guidelines are adopted from best commercial practices in ways that apply to public service applications as well. The system capability "elevator" description comes from Geoffrey Moore's classic *Crossing the Chasm* [13]. The "Benefits Realized" and "Results Chain" sections are adapted from the DMR Consulting Group's Benefits Realization Approach [14]. The Results Chain identifies the full set of initiatives necessary to realize the proposed system's benefits; this also identifies the full set of success-critical stakeholders who should be involved in the system's definition. The current version of the guidelines is at <http://cebase.org/cebase method>.

Table of Contents

2. Shared Vision
- 2.1. System Capability Description
  - 2.1.1. Benefits Realized
  - 2.1.2. Results Chain
- 2.2. Key Stakeholders
- 2.3. System Boundary and Environment
- 2.4. Major Project Constraints
- 2.5. Top-Level Business Case
- 2.6. Inception Phase Plan and Required Resources
- 2.7. Initial Spiral Objectives, Constraints, Alternatives, and Risks

2.1 System Capability Description

A concise description of the system that can pass the "elevator test" described in Geoffrey Moore's *Crossing the Chasm* [13]. This would enable you to explain why the system should be built to an executive while riding up or down an elevator. It should take the following form:

- For (target customer)
- Who (statement of the need or opportunity)
- The (product name) is a (product category)
- That (statement of key benefit-that is, compelling reason to buy)
- Unlike (primary competitive alternative)
- Our product (statement of primary differentiation)

Here is an example for a corporate order-entry system: "Our sales people need a faster, more integrated order-entry system to increase sales and customer satisfaction. Our proposed Web order system would give us an e-commerce order-entry system similar to Amazon.com's that will fit the special needs of ordering mobile homes and their aftermarket components. Unlike the template-based system our main competitor bought, ours would be faster, more user friendly, and better integrated with our order fulfillment system."

Common Pitfalls:

- Not relating the need or opportunity to the goals in the organization's Shared Vision.
- Being too verbose about "our product" or its key benefits.

2.2 Key Stakeholders

Identify each stakeholder by their home organization, their authorized representative for project activities, and their relation to the Results Chain. The four classic stakeholders are the software/IT system's users, customers, developers and maintainers. Additional stakeholders may be system interfacers, subcontractors, suppliers, venture capitalists, independent testers, and the general public (where safety or information protection issues may be involved).

Common Pitfalls:

- Being too pushy or not pushy enough in getting your immediate clients to involve the other success-critical stakeholders. Often, this involves fairly delicate negotiations among operational organizations. If things are going slowly and you are on a tight schedule, seek the help of your higher-level managers.
- Accepting just anybody as an authorized stakeholder representative. You don't want the stakeholder organization to give you somebody they feel they can live without. Some good criteria for effective stakeholder representatives are that they be empowered, representative, knowledgeable, collaborative, and committed.

Figure 5: *Example CeBASE System-Level Shared Vision Content*

CeBASE Method Coverage of the CMMI

Example Mapping: Requirements Development

To test its coverage of critical issues, we have done a mapping of the CeBASE method onto the CMMI's 24 process areas using the CMMI summary tables in Ahern, et al. [15]. A mapping example is provided in a longer version of this paper available at <http://www.cebase.org>.

Overall, the mapping indicated that the CeBASE method covered the CMMI goals and practices well. It provided the CeBASE team with some action items to address missing elements covered in the CMMI. Most significantly, though, it identified items that we have found important to software and systems engineering that were missing in the CMMI. These included a business case justifying the need for required features, having a stakeholder win-win prioritization of requirements (for coping with new requirements and fixed budgets or schedules), coverage of project requirements (required platforms, resource constraints), level of service requirements (the -ilities), and evolution requirements (to avoid point-solution architectures).

Overall CeBASE Method Coverage of the CMMI

Overall, we found not only a strong correspondence but also an almost complete coverage of the CMMI's practices by the organizational and project components of the CeBASE method. We are extending the CeBASE method to cover the specific CMMI processes not currently covered. A summary of the percentage of the CMMI process areas covered by the CeBASE method is shown in Table 1 (see page 14). The "+" annotations in Table 1 indicate that the CeBASE method's coverage goes considerably beyond that of the CMMI. For example, it covers not just an organizational process focus but also an organizational product and people focus. The "-" annotations in Table 1 indicate that some areas in the CeBASE method still remain to be fleshed out, such as detailed guidelines for organizational training plans, although they are covered in principle.

The CeBASE method also provides a prescriptive approach for an organization to use in tailoring the CMMI's generic practices to its particular culture, environment, and value propositions. Thus, an e-commerce organization's value propositions (rapid time to market, rapid adaptation to change) will

Process Management • Organizational Process Focus: 100+ • Organizational Process Definition: 100+ • Organizational Training: 100- • Organizational Process Performance: 100- • Organizational Innovation and Deployment: 100+	**Engineering** • Requirements Management: 100 • Requirements Development: 100+ • Technical Solution: 60+ • Product Integration: 70+ • Verification: 70- • Validation: 80+
Project Management • Project Planning: 100 • Project Monitoring and Control: 100+ • Supplier Agreement Management: 50- • Integrated Project Management: 100- • Risk Management: 100 • Integrated Teaming: 100 • Quantitative Project Management: 70-	**Support** • Configuration Management: 70- • Process/Product Quality Assurance: 70- • Measurement and Analysis: 100- • Decision Analysis and Resolution: 100- • Organizational Environment for Integration: 80- • Causal Analysis and Resolution: 100

Note: All amounts are percentages.

Table 1: *CeBASE Method Coverage of CMMI*

cause it to adopt more flexible processes. However, such elements as the anchor-point milestones will balance this flexibility with sufficient discipline to keep the overall process under control. The value propositions of an organization developing safety-critical products or services will cause it to emphasize more rigorous specifications, processes, and practices, but in ways that enable it to cope with rapid change. Examples include capturing evolution requirements, designing systems to accommodate future change, building in buffer periods to synchronize and stabilize processes [16], or to adapt to potential schedule or budget slips by dropping lower-priority product features [3].

Another point worth emphasizing is that the EF component of the CeBASE method supports a continuous vs. staged approach to process improvement. You do not need to be a CMM Level 4 organization to begin realizing significant benefits from organizational innovation or causal analysis.

Using the CeBASE Method

Since 1996, we have been applying the EF and GQM approaches to improve the project-oriented MBASE aspects of the CeBASE method by using them to improve an annual series of USC electronic services projects. These are developed using annually improved MBASE guidelines by teams of five master's-level students as developers and staff members of USC's Information Services Division as clients (customers, users or user representatives, and maintainers). Each year, we have 15 to 20 teams execute the MBASE inception and elaboration phases in the fall semester to develop and validate life-cycle architecture packages for USC electronic services applications' candidates. The top six to 10 of these applications are then selected for spring semester teams who execute the MBASE construction and transition phases and deliver initial operational capability application systems.

Our shared vision for the USC Center for Software Engineering's research and education goals incorporates the win conditions of not only our students and their project clients, but also other stakeholders such as the center's staff and prospective technology users, represented by our 35 industry and government affiliate organizations [17]. Our questions and metrics include stakeholder critiques of each project and extensive instrumentation of the projects' effort, schedule, quality, productivity, and behavioral characteristics [18, 19, and 20].

Table 2 summarizes four years' experience to date in applying and refining CeBASE on an annual selection of real-client projects.

A few explanatory comments on Table 2 are in order. The number of LCA teams is larger than the number of IOC teams because USC's fall course is a core course for the USC master's of science degree in computer science and has a much larger enrollment than the spring course, which is only required for a few specialization areas. In 1996-97, the subset of projects to be continued in the spring was primarily those having students continuing from the fall course. After we found that most of the 1996-97 products went unused, we performed a critical success factor analysis and determined a set of spring project selection criteria (e.g., library commitment to product use, empowered clients) that increased the project adoption rate. Even then, unforeseen circumstances such as the inevitable changes in library infrastructure and organizational responsibilities have caused some applications' usage commitments to be overtaken by events. This is a frequent phenomenon for electronic services applications [21].

In general, the EF improvements on MBASE have effected a uniform improvement in outcome, but there are some anomalies. For example, the 12 percent of projects failing IOC in 1999-2000 were due to a team who botched their product transition when their client was unexpectedly called out of town during the transition period. Another example was our introduction of midcourse client briefings on core capability expectations in 1999-2000 as part of our SAIV process [3]. SAIV only guarantees the delivery of a highest-priority core capability set of features, with further features added as time is available. While this resulted in early client disappointments at LCA where client success scores dropped from 4.74 in 1998-1999 to 4.48 in 1999-2000, there was a dramatic increase in the clients' success score for the delivered product (4.3 in 1998-1999 to 4.75 in 1999-2000).

The 1998-1999 improvement in the "Failing LCO" criterion shown in Table 2 resulted primarily from our introduction of a simplifier and complicator (S&C) expectations management activity. This helped the developers to have a better understanding of the system and the stakeholders by leveraging an experience base of designs that help simplify the architecture (the *simplifiers*) and apply a risk-driven approach to the architectural areas that may cause significant complications (the *complicators*). Involving the clients in risk management activities throughout the process clearly contributed to their rating virtually all delivered applications as highly satisfactory.

Table 2: *Annual University of Southern California E-Services Project Outcomes*

	1996-1997	1997-1998	1998-1999	1999-2000
LCA Teams	15	16	19	22
Failing LCO	27%	25%	5%	5%
Failing LCA	0%	0%	0%	0%
LCA Client Score	4.46	4.67	4.74	4.48
IOC Teams	6	5	6	8
Failing IOC	16%	0%	0%	12%
IOC Client Score	n/a	4.15	4.3	4.75
IOC Regularly Used	16%	60%	50%	62%

Conclusions

Figure 6 summarizes the distinctions among maturity models such as the SW-

CMM and the CMMI; process models such as the waterfall model; and process model generators such as MBASE, RUP, and the CeBASE method. It shows where each model fits with respect to organizational focus (project vs. organization), application focus (software vs. system), and operational focus (practice vs. assessment).

From Figure 6, we can see that the SW-CMM covers both project and organization considerations, but it has shortfalls in both applications focus (software, not systems) and operational focus (assessment, not practice). We can also see that solutions focused on redressing one of the two shortfall dimensions will still have shortfalls of their own in another dimension. Thus, the CMMI redresses the systems shortfall in the SW-CMM, but it still has the shortfall of providing explicit guidelines for assessment but not for project practices. While MBASE and RUP provide explicit guidelines for project practices, they do not provide counterparts for an organization's practices. However, the combination of CMMI and the CeBASE method covers all aspects of operational, organizational, and application focus.

In terms of future software-intensive system challenges, the ability to balance discipline and flexibility is critically important to the development of highly dependable software-intensive systems in an environment of rapid change. The CMMI's risk-management orientation enables its users to apply risk considerations to determine how much discipline and how much flexibility is enough in a given situation. The risk-driven nature of the spiral model and MBASE enables them to achieve a similar balance of discipline and flexibility. When these project-level approaches are combined with the organization-level approaches in the EF, the result is the unified CeBASE method summarized in the section "The CeBASE Method and the CMMI." It currently implements most of the CMMI, is being extended to cover the full CMMI, and has a strong track record of continuous process improvement at USC's and UMD's Software Engineering Laboratories and industry adapters elsewhere.◆

Operational Focus: Assessment / Practice		**Organizational Focus** Project	Organization
Application Focus	Software	Software CMM / Waterfall, Incremental	Software CMM / Early EF, GQM
	Systems	CMMI / Spiral, MBASE, RUP	CMMI / CeBASE Method

Figure 6: *Process Model Coverage Distinctions*

Acknowledgements

We would like to acknowledge the support of the National Science Foundation in establishing CeBASE, the DoD Software Intensive Systems Directorate in supporting its application to DoD projects and organizations, and the affiliates of the USC Center for Software Engineering and the University of Maryland's software engineering program for their contributions to MBASE and CeBASE.

References

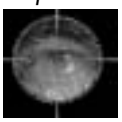
1. Boehm, B., and W. Hansen. "Understanding the Spiral Model as a Tool for Evolutionary Acquisition." CrossTalk May 2001.
2. Boehm, B., and D. Port. "Balancing Discipline and Flexibility with the Spiral Model and MBASE." CrossTalk Dec. 2001.
3. Boehm, B., D. Port, L. Huang, and A. W. Brown. "Using the Spiral Model and MBASE to Generate New Acquisition Process Models: SAIV, CAIV, and SCQAIV." CrossTalk Jan. 2002.
4. Basili, Victor R., and Gianluigi Caldiera. "Improve Software Quality by Reusing Knowledge and Experience." Sloan Management Review 37.1 (1995).
5. Basili, Victor R., Gianluigi Caldeira, and H. D. Rombach. "The Goal Question Metric Approach." Encyclopedia of Software Engineering. Ed. J. Marciniak. Wiley, 1994.
6. Basili, Victor R., Gianluigi Caldeira, and H. D. Rombach. "The Experience Factory." Encyclopedia of Software Engineering. Ed. J. Marciniak. Wiley, 1994.
7. Basili, Victor R., Gianluigi Caldiera, Frank McGarry, Rose Pajersky, Gerald Page, and Sharon Waligora. "The Software Engineering Laboratory – An Operational Software Experience Factory." 14th International Conference on Software Engineering. May 1992.
8. Basili, Victor R., Marvin Zelkowitz, Frank McGarry, Jerry Page, Sharon Waligora, and Rose Pajerski. "Special Report: SEL's Software Process-Improvement Program." IEEE Software 12.6 (1995): 83-87.
9. McGarry, F. "What Is A Level 5 Organization? Lessons from 10 Years of Process Improvement Experiences at CSC." Proceedings of the Twenty-Sixth NASA Software Engineering Workshop. Nov. 2001.
10. Boehm, B. Software Engineering Economics. Prentice Hall, 1981.
11. Ferguson, P., et al. "Software Process Improvement Works!" Advanced Information Services, Inc. CMU/SEI-99-TR-027, Nov. 1999.
12. Butler, K., and W. Lipke, "Software Process Achievement at Tinker Air Force Base, Oklahoma." CMU/SEI-2000-TR-014, Sept. 2000.
13. Moore, Geoffrey. Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers. New York: Harper Business, 1991: 161.
14. Thorp, J., and DMR Consulting Group. The Information Paradox, McGraw Hill, 1998.
15. Ahern, D., A. Clouse, and R. Turner. CMMI Distilled. Addison Wesley, 2001.
16. Cusumano, Michael A., and Richard W. Selby. Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People. New York: Simon & Schuster, 1996.
17. Boehm, B., A. Egyed, D. Port, A. Shah, J. Kwan, and R. Madachy. "A Stakeholder Win-Win Approach to Software Engineering Education." Annuals of Software Engineering 6 (1998): 295-321.
18. Egyed, A., and B. Boehm. "Comparing Software System Requirements Negotiation Patterns." Systems Engineering 2.1 (1999): 1-14.
19. Port, D., and B. Boehm. "Introducing Risk Management Techniques Within Project-Based Software Engineering Courses." Computer Science Education 2002 (to appear).
20. Majchrzak, A., and C. Beath. "A Framework for Studying Learning and Participation in Software Development Projects." Management Informational Systems Quarterly, under review.
21. Boehm, B., "Project Termination Doesn't Equal Project Failure." IEEE Computer Sept. 2000: 94-96.

COMING EVENTS

May 13-17
Software Testing Analysis and Review (STAREAST 2002)

Orlando, FL
www.sqe.com/stareast

June 3-6
Combat Identification Systems Conference



Colorado Springs, CO
www.usasymposium.com

July 18-20
Shareware Industry Conference
St. Louis, MO
www.sic.org

July 22-25
Joint Advanced Weapons Systems Sensors, Simulation, and Support Symposium (JAWS S3)
Colorado Springs, CO
www.jawswg.hill.af.mil

August 19-22
The Second Software Product Line Conference
San Diego, CA
www.sei.cmu.edu/SPLC2/

September 9-13
International Conference on Practical Software Quality Techniques and International Conference on Practical Software Testing Techniques 2002 North
St. Paul, MN
www.softdim.cim/psqt/

April 28-May 1, 2003
Software Technology Conference 2003

Salt Lake City, UT
www.stc-online.org

About the Authors

Barry Boehm, Ph.D., is the TRW professor of software engineering and director of the Center for Software Engineering at the University of Southern California. He was previously in technical and management positions at General Dynamics, Rand Corp., TRW, Defense Advanced Research Projects Agency, and the Office of the Secretary of Defense as the director of Defense Research and Engineering Software and Computer Technology Office. Dr. Boehm originated the spiral model, the Constructive Cost Model, and the stakeholder win-win approach to software management and requirements negotiation.

University of Southern California
Center for Software Engineering
Los Angeles, CA 90089-0781
Phone: (213) 740-8163
Fax: (213) 740-4927
E-mail: boehm@sunset.usc.edu

Daniel Port, Ph.D., is a research assistant professor of Computer Science and an associate of the Center for Software Engineering at the University of Southern California (USC). Dr. Port's previous positions were assistant professor of Computer Science at Columbia University, director of Technology at the USC Annenburg Center EC2 Technology Incubator, co-founder of Tech Tactics, Inc., and a project lead and technology trainer for NeXT Computers, Inc. He received a doctorate from the Massachusetts Institute of Technology in applied mathematics with an emphasis on theoretical computer science in 1994 and a bachelor's degree in mathematics from the University of California in Los Angeles.

University of Southern California
Center for Software Engineering
Los Angeles, CA 90089-0781
Phone: (213) 740-7275
Fax: (213)740-4927
E-mail: dport@sunset.usc.edu

Apurva Jain is a doctoral student at the University of Southern California's Center for Software Engineering. Previously he was a project manager at SpruceSoft Inc. His research interests are software process management and pervasive computing. He received a bachelor's degree from Curtin University of Technology, Perth, Australia and a professional honors diploma from Informatics, Singapore.

University of Southern California
Center for Software Engineering
941 W. 37th Place, SAL 329
Los Angeles, CA 90089-0781
Phone: (213) 740-6505
Fax: (213) 740-4927
E-mail: apurvaj@sunset.usc.edu

Victor R. Basili, Ph.D., is a professor of Computer Science at the University of Maryland, the Executive Director of the Fraunhofer Center, Maryland, and one of the founders and principals in the Software Engineering Laboratory. He works on measuring, evaluating, and improving the software development process and product and has consulted for many organizations. Dr. Basili is a recipient of a 1989 NASA Group Achievement Award, a 1990 NASA/GSFC Productivity Improvement and Quality Enhancement Award, the 1997 Award for Outstanding Achievement in Mathematics and Computer Science by the Washington Academy of Sciences, and the 2000 Outstanding Research Award from ACM Special Interest Group on Software Engineering.

Computer Science Department
4111 AV Williams Building
University of Maryland
College Park, MD 20742
Phone: (301) 405-2668
Fax: (301) 405-2691
E-mail: basili@cs.umd.edu