Fitzgerald, B. and O'Kane, T. (1999) 'A Longitudinal Study of Software Process Improvement', *IEEE Software*, 16(3), p.37-45.

Case Study

Studying a software process improvement effort over time reveals the factors associated with its success. This case study shows how Motorola's Cellular Infrastructure Group progressed to CMM level 4, and examines what is needed to optimize its software development process.

A Longitudinal Study of Software Process Improvement

Brian Fitzgerald, University College Cork, Ireland
Tom O'Kane, Motorola

Given the extent to which software underpins all our everyday activities, software development has become a critical issue for modern organizations, and indeed for all of society. The "software crisis," a term first coined more than 30 years ago, continues unabated, however—software still takes too long to develop, costs too much, and does not work very well when eventually delivered. Some argue that this is because the software process in many organizations is undisciplined, chaotic, and completely unpredictable.[1] In other organizational areas, increased process maturity has led to concomitant improvements in efficiency. Therefore, increasing software process maturity is an obvious and logical step in addressing the software crisis. The most comprehensive and effectively realized model of process maturity is the Capability Maturity Model from the Software Engineering Institute.[1]

While the CMM was initially applied to government and military software development, its use is now spreading to all industry sectors. However, a large-scale comprehensive study of organizations that have undergone CMM evaluation reveals that only two percent of organizations have achieved level 4 or 5; nearly 62 percent are at level 1.[2] Also, the process of achieving level 3 takes four years on average,[2] which shows that progression through CMM maturity levels is both time-consuming and difficult.

FROM THE TRENCHES: Wolfgang B. Strigel, editor • wstrigel@spc.ca

0740-7459/99/$10.00 © 1999 IEEE

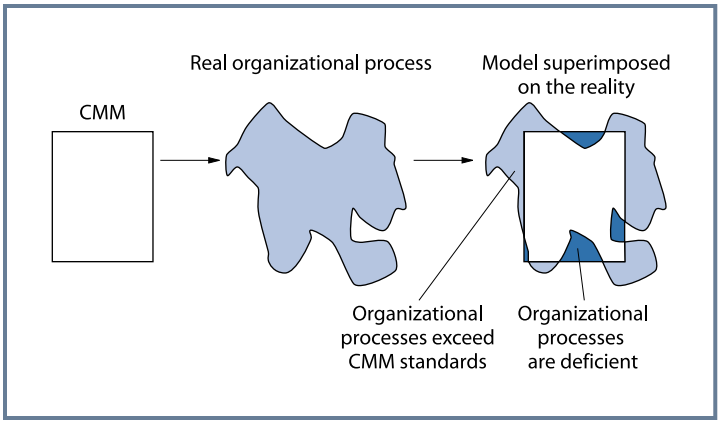

Figure 1. Application of CMM to the organizational reality of an organizational process.

This case study examines how Motorola's Cellular Infrastructure Group at Cork, Ireland achieved each of the CMM maturity levels up to level 4 between 1993 and 1997. CMM assessment findings, particularly the critical success factors, are a rich source of information for the organization. We identify and discuss in detail the CSFs that CIG's CMM assessment team associated with the achievement of each maturity level, as well as those CSFs that were not achieved—these represent opportunities for further improvement. Given the ever-increasing interest in software process improvement in general, and the CMM in particular, the lessons learned by the Motorola organization are likely to be of significant interest to others.

Defining and Achieving Success

The CMM is a five-layer model describing an evolutionary path from ad hoc, chaotic processes to mature, disciplined software processes. The five maturity levels describe successive foundations for continuous process improvement and define an ordinal scale for measuring the maturity of an organization's software process. The CMM consists of 18 key process areas categorized across the five maturity levels. These KPAs can be split further into management, organizational, and engineering categories (see Table 1).

The CSF method, first applied to the information systems field in 1979,[3] defines specific things that must go right for a business to flourish and maintain its competitive edge. The concept has been widely applied in the IS field,[4] but not so much in the specific area of software development. Some notable examples of CSF research in software development include studies of strategic information systems development,[5] executive information systems development,[6] and the development of IBM's OS400 system.[7] The latter is perhaps the most relevant to our work; the 13 CSFs identified in that study overlap quite well with the CSFs we identified at the Cork facility (listed in Table 1).

However, while these previous studies identify a number of CSFs, some of which are broadly similar, none addresses the achievement of a particular measurable standard of success. In our study, the CSFs identified are closely coupled to the achievement of CMM maturity levels. We also extend the application of the CSF concept to software process improvement.

The relationship between KPAs and CSFs requires elaboration. While Motorola satisfied the relevant KPAs to achieve each CMM level, KPAs actually represent a binary concept in that just exceeding the threshold level on each factor is "better" (at least in CMM terms) than scoring very well on several KPAs but failing to reach the threshold on one. Thus, no strong incentive exists for an organization to greatly exceed any KPA. We believe, however, that the focus should always be on improving the software process, not merely achieving a score, as richness in any aspect of process improvement is rendered somewhat sterile by reverting to a single score. The CMM assessment team at Motorola noted this richness, and therefore successful process improvement can be viewed in terms of CSFs rather than KPAs. CSFs can be seen as situated exemplars that help push out the boundaries of process improvement, which cannot be realized by merely achieving a threshold level on a set of KPAs.

Figure 1 depicts this issue graphically. Illustrating how the CMM model is superimposed onto the organizational reality lets us identify deficiencies in the organization's process. However, this is only within the frame of reference of the CMM and only to the degree to which the CMM itself is correct. This is, of course, how the CMM is supposed to be used.

In reality, however, this is a simplistic view. An organization may in fact be enacting processes identified by the CMM but far exceeding its boundaries. The organization may also be enacting processes not captured explicitly by the CMM, such as those for data security and disaster recovery. Even though the organization is following a process that is at variance with the CMM, it may be still correct as the CMM itself is not perfect. In Figure 1, the areas that are deficient in the organization's process, vis-à-vis the CMM, are represented with waved shading.

In effect, the totality of process in the real organization is much richer than that of the CMM. Unfortunately, some may have overlooked this and,

Table 1
Key Process Areas and Critical Success Factors by Maturity Level

CMM Level	KPA: Management	KPA: Organizational	KPA: Engineering	CSFs at Motorola, Cork
1. **Initial**	Ad-hoc processes			
2. **Repeatable**	Requirements management Software project planning Software project tracking and oversight Software subcontract management Software configuration management			**CSF1**: Dedicated project planning and tracking roles. **CSF2**: Formalized procedures for managing subcontractors. **CSF3:** Institutionalized software quality assurance program. **CSF4:** Software configuration management automated and intrinsic to the software life cycle
3. **Defined**	Integrated software management Intergroup coordination	Organization process focus Organization process definition Training program	Software product engineering Peer reviews	**CSF1**: Comprehensive process definition and tailoring technique. **CSF2:** Strong and pervasive culture for software process improvement. **CSF3**: Good training program with emphasis on leading-edge technologies. **CSF4**: Strong cooperation both within Motorola and with external organizations. **CSF5**: Peer review of all software products.
4. **Managed**	Quantitative process management		Software quality management	**CSF1**: Data-driven culture for management of products, projects, and subcontractors.
5. **Optimized**		Technology change management Process change management	Defect prevention	**CSF1**: Technology change program fully deployed across whole organization. **CSF2:** Organizational process improvement culture sufficiently proactive and involved in strategic goal setting. **CSF3**: Defect prevention program coordinated across the whole organization.

consequently, confine process improvement activity within the model's boundaries. Another more disquieting reason may be that commercial considerations force an organization to strictly confine itself to the model because it is pursuing a CMM score—it then takes a minimalist approach instead of fully engaging in software process improvement. In effect the CMM is regarded as the reality, which of course is wrong since all models are wrong to a greater or lesser degree.

Applying the CSF concept

Motorola, a major mobile telecommunications systems provider, has over 300 engineers creating large, complex, expensive switching and communications infrastructure systems in Cork, Ireland. Clients are typically very large telecommunications providers who use these systems to support their mobile phone networks. Users (individuals who make mobile telephone calls) take the underlying system completely for granted and expect total reliability. With several significant and reputable competitors in the marketplace, Motorola systems must be reliable. Also, the telecommunications technology area is constantly evolving, with new products and services continually offered. Systems are constantly being adapted to incorporate interfaces to these new developments.

The systems themselves are developed using common languages such as C and C++. Technical

personnel, who typically have a background in engineering or computer science, work in a formalized development environment where the design, implementation, and testing phases are clearly differentiated. All projects follow a methodology tailored precisely to the development process, and Motorola uses special test laboratory facilities to test each system function rigorously. Given the system requirements and the competitive marketplace, errors and downtime must be kept to a minimum. When errors do occur, all fixes undergo rigorous testing before a system is released to customers.

Maintaining the integrity of all software work products is essential for any software organization.

The development process is explicitly documented, and is evolving as the company follows its program for continuous process improvement, which it expects will ultimately lead to an improved CMM rating. Motorola places much emphasis on satisfying the concepts of the CMM and has created a specialist group—the Software Engineering Process Group—at the Cork facility to ensure compliance with the CMM criteria. The SEPG collects and analyzes metrics on the development process and posts this information on internal notice boards for developers to read.

The pioneers of the CSF method state that its application requires a thorough understanding of "the industry, the specific company, and the job being performed."[8] Our research took place longitudinally over a five-year period, and one of us manages software process improvement at the Cork facility. Our overall research strategy was based on *action research*,[9] a method that merges research and practice while accepting that pragmatic organizational realities take precedence over the artifices of any research method. The method consists of a spiral of planning, acting, observing, and reflecting cycles, which is compatible with a software process improvement program as the same steps are essentially followed. Thus, the method was ideally suited to the research objective.

Critical success factors: CMM level 2

CMM level 1 is the default for all organizations—it doesn't have to be achieved as such. The software process at this level is characterized as chaotic and ad hoc, and any successes are primarily due to the heroic efforts of talented individuals. Thus, the first significant level of the CMM, in that it represents a hurdle to be achieved, is level 2, the *repeatable* level. The KPAs of CMM level 2 (see Table 1) focus on bringing discipline and basic management control to the software process. Project management is institutionalized and projects begin to meet cost and schedule deadlines and satisfy requirements.

The CIG at Cork has a long track record in software process improvement and was initially assessed at CMM level 2 prior to 1993, based on an internal self-assessment.

CSF 1: Dedicated project planning and tracking roles. Producing software is an expensive business. Making optimal use of available resources, both human and computer, is vital if software development is to make economic sense. Project planning and tracking are therefore essential.

At CIG, full-time project planning and tracking personnel are assigned to each project and work as part of the software product engineering team. Starting with the requirements group's estimates for all feature deliverables, they estimate project risks based on historical data from previous projects, then factor in other known-overhead activities such as training, software process work, and holidays. Finally, they allocate engineering team resources and compile a work breakdown structure for the project. Once the project plan is in place, they track progress using such techniques as earned-value analysis. Their principal aim is to ensure the project budget is maintained. Regular project meetings are convened to assess progress, and progress reports and a history of the project are maintained. These then provide feedback for the engineering teams.

CSF 2: Formalized procedures for managing subcontractors. Selecting and managing software subcontractors requires great care and discipline—failure to do so effectively has the potential to create havoc and put the software organization in a vulnerable position.

CIG has formalized a set of solid and proven procedures to deal with the selection, verification, tracking, and performance measurement of software subcontractors. A dedicated subcontract manager ensures that the software produced by subcontractors at least matches the quality of the software produced by CIG itself in terms of defect densities, timeliness of delivery, and quality of development processes. All subcontractors must pass a formal and rigorous assessment to qualify for selection, and once selected their performance is closely moni-

tored to ensure quality standards are maintained.

CSF 3: Institutionalized software quality assurance. It is essential to ensure the software organization's processes, procedures, and standards are complied with if previous project successes are to be repeated. This is usually achieved through compliance audits where the auditor maintains an independent, objective view of the software work products and activities.

At Cork, the software quality department ensures software product quality by auditing all work products for compliance with the defined software process. The quality department has sign-off responsibility for all customer deliverables, conducts regular reviews with the project groups, and is the conduit for customer feedback reports and customer satisfaction indices.

CSF 4: Software configuration management is an intrinsic, automated part of the software life cycle. Maintaining the integrity of all software work products is essential for any software organization. CIG staff assigned to SCM use specialized tools for integrated file management. Automated SCM is a key feature deployed throughout CIG and affects all aspects of the software life cycle. The primary task is to ensure that version control is exercised over all the components that make up the final software deliverable plus any other artifacts developed by the organization.

Critical success factors: CMM level 3

CMM level 3 ("defined") extends the scope of process improvement to organizational issues. An appropriate infrastructure must be established to ensure that both software engineering processes and the related management processes are fully institutionalized. CIG developed a technique to explicitly define the software process and established sound management practices to enhance it.

CSF 1: Comprehensive process definition and tailoring. Having a defined process is absolutely essential for progressing to higher levels of process maturity. However, in practice, this can be a cumbersome and difficult task. Prompt (PROcess MaPping and Tailoring) is a method developed by Tom O'Kane for defining software process maps at Motorola. It is based on a simple but rigorous set of symbols and rules, and allows for a concise graphical depiction of the software process (see the boxed text, "The Prompt Notation," on pp. 43-44 for more details). The technique was developed at Cork but is now in widespread use throughout Motorola, and its use has spread to other companies.

The Prompt technique makes possible a high level of accuracy and permits tailoring the process to the contingencies of each project. The process map can reflect the process as actually applied rather than documenting what it should be. Prompt maps are also task-oriented—rather than describing jobs performed by individuals, they provide a holistic, graphical representation of the entire software workflow.

CSF 2: Strong and pervasive culture for software process improvement. At CIG, ownership of the software process, plus the responsibility for its maturation and continuous improvement, pervades all organizational levels. While the software process improvement effort focuses on facilitating and managing change in the organization, the overriding belief at CIG is that process ownership and development are best placed with those closest to the process, who experience its bottlenecks and inefficiencies. Ownership of the software processes therefore does not reside with the process management function, which is responsible for providing the necessary resources and support infrastructure to enable the process to evolve in a controlled manner. To realize benefits such as cycle-time reduction, those managing the improvement program need to create awareness among the general engineering community of the need to change.

Process ownership and development are best placed with those closest to the process.

CSF 3: Good training program with emphasis on leading-edge technologies. Motorola regards employee training as critical. At CIG this is driven through the development of individual training plans for each employee. New employees receive intensive training on both the technical and nontechnical aspects of their positions. Thereafter each employee receives a minimum of five days of training per year. The training department develops the annual training plan for the facility based on individual training plans. It also maintains an extensive suite of Web-based and PC-based self-directed

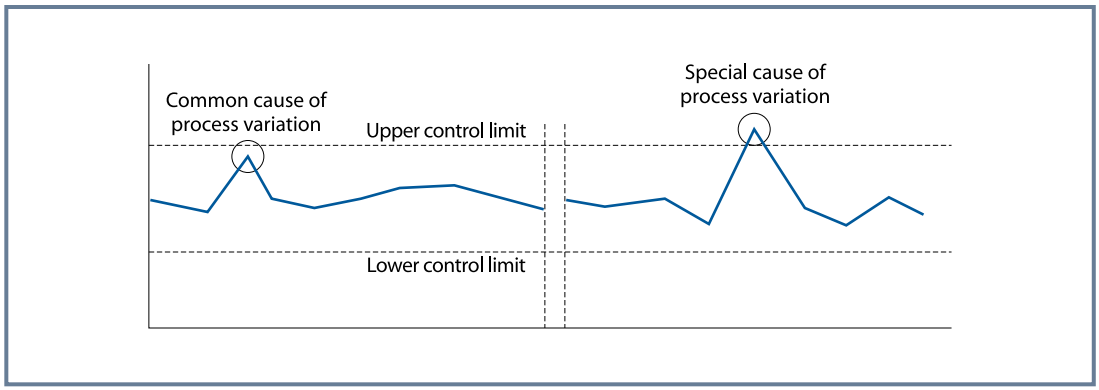

Figure 2. Common and special causes of process variation, represented graphically.

learning programs. All employees are encouraged to conduct workshops, both to share their knowledge and to improve their own presentation skills.

CSF 4: Strong cooperation within Motorola and with external organizations. Developing external contacts, participating in other engineering groups' activities, and maintaining links with parent organizations in the US help employees gain a greater awareness of the business context within the organization. This is seen as vitally important at Motorola. In addition to cooperation between the various functional groups at CIG, employees also have a great deal of contact with external organizations. These include the parent organization in the US, subcontractor companies, other Motorola sites around the world, educational institutions, professional societies, and many others. Cooperation occurs at both individual and organizational levels.

CSF 5: Peer review of all software products. Motorola has found formal inspections to be the most efficient and cost-effective means of removing defects from software products. At CIG, all software products and artifacts undergo peer review. The organization has adopted the formal inspection methodology developed by Michael Fagan and has evolved and tailored this process to suit the precise contingencies of Motorola's development process. The peer review process, always considered strong at Motorola, has formed a foundation for subsequent achievement of higher levels, as it has catalyzed the collection of a huge amount of data for level 4 analysis.

Critical success factors: CMM level 4

CMM level 4, the managed level, focuses on establishing quantitative goals for both the software process and software products. At Motorola, individual managers are responsible for developing metrics (for example, requirements estimation accuracy) to indicate their group's performance levels. The company also devotes resources to the full-time collection and analysis of data. Much effort is invested in measuring the performance of key process indicators and ensuring they remain within acceptable levels.

Thus the software process is under quantitative control, and *special causes* that represent deviations outside the control limits are identified and isolated (Figure 2). A special cause is a source of variation that is intermittent, unpredictable, or unstable; it is signaled by a point beyond the control limit.[10] *Common causes*, also shown in Figure 2 and discussed later, are signaled by points that approach but do not exceed the control limit.

For example, in C code inspections, scatter plots may reveal that the optimum inspection rate is 150 lines of code per hour. If inspection occurs at a rate (faster or slower) beyond the limits of acceptable process variation, inspection effectiveness falls dramatically, to an unacceptable level. Subsequent investigation could reveal that the team started late and was then in a hurry to finish or, alternatively, got bogged down in actually solving a problem rather than concentrating on fault-finding.

CSF 1: Data-driven culture for management of products, projects, and subcontractors. A strong data-driven culture exists at CIG. Metrics are collected during all phases of the software life cycle; hence a very rich and organizationally diverse set of raw data is available for analysis and refinement. These metrics are formally presented and reviewed monthly at internal quality and operations reviews, but the information is also made available to the wider engineering community through notice boards, the Internet, and individual group feedback sessions. The assessment team also noted the potential for increased leverage of Internet technology to further enhance Motorola's metrics collection activities.

Critical success factors: CMM level 5

At CMM level 5, the KPAs focus on continuous and measurable process improvement to optimize the software process. At Motorola, special empha-

The Prompt Notation

Prompt, which derives its name from Process Mapping and Tailoring, is a notation composed of four basic symbols—rectangles, triangles, circles, and dots—and a simple rule set. Prompt maps describe the actions of human beings in a field of work. They are not prescriptive—they do not attempt to describe how tasks are performed or who performs them, but instead facilitate and time-order such descriptions.

Prompt maps consist of the following:

♦ *Entry/exit criteria*: rectangles labeled Entry or Exit Criteria, containing the entry or exit criteria text. These provide the control signals that either start or end a process. Every Prompt map begins with a single Entry Criteria node describing the criteria to be satisfied before the first action can be started, and terminates with a single Exit Criteria node describing the criteria to be satisfied for the process to be fully completed.

♦ *Input/output criteria*: a rectangle with an arc leading from or to it and containing the input/output criteria text. Input criteria are artifacts required by an action point but satisfied by some other independent process; output criteria are artifacts used by some other independent process.

♦ *Action points*: a triangle containing an exclamation mark. These represent processes or other Prompt maps (submaps). Activities are collected into actions that are highly cohesive and loosely coupled. For example, the activities High-Level Design, Low-Level Design, and Code are sequential—coding cannot start if low-level design has not started, and low-level design cannot completely finish until high-level design has finished (Figure A).

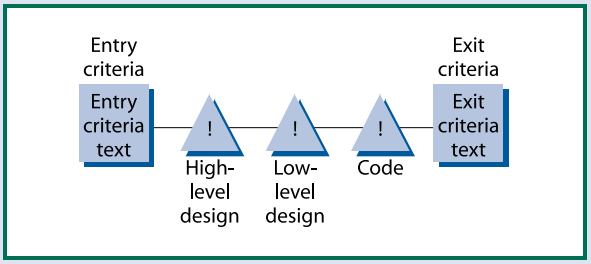

Figure A. Sequential ordering of action points indicates their dependent relationships.

♦ *Option Points*: a circle containing a question mark. These represent junctions where a human decision must be made. The decision is strictly Boolean; when a decision is made, only one of the two exit paths is followed. The selected action following an option point cannot conclude until the action immediately preceding the option point is complete.

♦ *Parallel Processing Points*: a dot/period on the arc between two nodes. These represent the instigation of independent, concurrent processing. A maximum of three exit paths may follow from a single parallel processing point. Activities following a parallel processing point cannot conclude until the action immediately preceding the parallel processing point is complete.

♦ *Document/Map References*: a dashed line perpendicular to the map. These point to a library reference, Internet URL, online documentation, or another Prompt map. Where a documentation reference emanates from an action point, the documentation is specific to that activity. Where the reference emanates from the arc between two nodes, the documentation is nonspecific but applies to the general process from this point on. Other Prompt maps (submaps) can only be referenced from an action point and obey the same rules as any other Prompt map.

Prompt maps are living artifacts and not intended to be definitive; they are strictly linear, flowing from left to right. They describe what has to be done, and the referenced procedures provide specific instructions on how to perform the work, including specific conditions, roles and responsibilities, iterations, recursions, and metric data to be collected or produced.

Prompt maps are typically Web-based, and the example in Figure B shows a process initiated by a Document Change Request. The first activity is to prepare the document change notification (DCN) using a DCN template, retrievable directly by

Continued on the next page

sis is placed on measuring and analyzing variations in software process performance to assess the common causes of problems and to effect prevention measures. The concern here is to go beyond the elimination of special causes, the focus at level 4, and to identify and eliminate the *common causes* of problems and defects to achieve a more uniform pattern of performance (represented by a flatter profile in Figure 2).

A common cause is a source of variation that is always present—it is part of the random variation inherent in the process itself and is indicated by the measure approaching the acceptable control limit (see Figure 2). Its origin can usually be traced to an element of the system that only management can correct.[10] To continue the code inspection rate example above, suppose that code inspection rates for a particular module commonly approach the acceptable upper limit. This could indicate that the module is overly complex or defect-prone and is thus a candidate for reengineering. The decision to proceed with such reengineering work would require management intervention.

Identifying and addressing common causes requires a two-pronged approach: first, continuous incremental improvement to the existing software process; and second, proactive efforts to introduce radical and innovative process changes through the deployment of new technologies.

clicking on the Prompt map component. A Web-based DCN process document that describes the DCN process is also available, retrievable directly from the Prompt map. The Retrieval and Update Process Document is a general document that describes the overall process from this point onward. After starting the DCN process, the user will begin the document retrieval process but will not be able to complete it until the Prepare DCN process is complete.

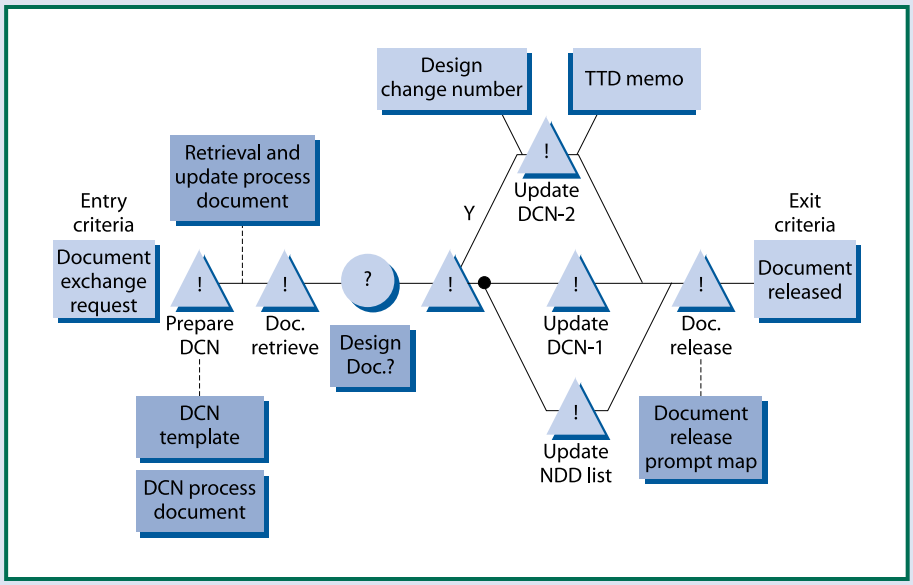

Figure B. A Prompt map for the document request process.

After beginning the retrieval process, the user needs to decide whether this is a design document, and can use tailoring criteria available online to make this decision. If it is a design document, the user can commence updating DCN-2 but will be unable to complete this process until the document retrieval process is complete and the design change number is available. At some point during this activity a Technical Tracing Department (TTD) memo will be made available to other processes. If the document wasn't a design document, the user needs to update DCN-1 and, at the same time, update the Non-Design Document (NDD) list. Neither of these activities can be completed until the Doc Retrieve process is completed. Regardless of the update path chosen, the user will at some point be able to commence the document release process, described by its own Prompt map. The entry and exit criteria for this submap are coherent with this process path. The document request process will eventually complete when the exit criteria are met.

In summary, the objective of the Prompt map is to specify a logical perspective for the activity domain by identifying actions, decisions, parallelisms, input/output artifacts, and overall entry/exit criteria. The terms used in this example are purposefully vague because humans and human processes use fuzzy logic, and attempts to eliminate the fuzz typically result in something inordinately complicated.

Motorola uses the Prompt notation widely in both Europe and the US. The Motorola Cellular Support Centers, which support Motorola's cellular customer base worldwide, aligned their processes using Prompt and thereby reduced many thousands of pages of documentation to a set of 21 maps that are Web-based and totally interactive. Now, as the worldwide Customer Network Resolution Centers, they serve Motorola's cellular customers with greatly enhanced service management facilities.

CSF 1: Technology change management program deployed across the whole organization. The CMM assessment team reported that while they were fully satisfied with the technology change management program developed at CIG, the evidence of its deployment had been drawn from only a single project type. While this evidence was good, Motorola wished to achieve deployment across all projects. This encapsulates the differences between KPAs and CSFs in that the KPA threshold had already been achieved in this case; however, Motorola viewed the CSF as a stronger basis for software process optimization.

We should note that the technology base at Cork is fairly stable; significant technological shifts are comparatively rare. Also, the decision to make a radical change such as using a new technology is more likely to be made at the higher group or sector levels and then passed down to the individual organizations for implementation.

CSF 2: Defect prevention program coordinated and deployed across the whole organization. The CMM assessment team noted that while many facets of the defect prevention program were in place at CIG, the group needed a more coordinated approach to organizing and deploying the program. This conclusion stemmed from the observation that while vast amounts of metric data were collected and analyzed, these activities were essentially reactive in nature. The defect prevention program needed to be more proactive, focusing on the prevention of faults in sub-

sequent releases of software based on data collected and statistically analyzed for the level 4 KPAs.

CSF 3: The organizational process improvement culture is proactive and involved in strategic process goal-setting. Again, the CMM assessment team found that while CIG had a strong process improvement culture, it was essentially reactive. It recommended that the organization identify more strategic opportunities for process improvement, set goals to realize those opportunities, plan ahead to reach those goals, and monitor progress towards their achievement. The net effect on the software process would be reduced statistical variation over a given period of time.

While CIG was addressing some of the level 5 KPAs, it was not achieving all the identified level 5 CSFs to the required level. It therefore shifted its focus to establishing these level 5 CSFs, as this would contribute significantly both to achieving level 5 and to ensuring optimization of its software process in the future.

We see the future for high-maturity organizations in terms of CSFs rather than KPAs. KPA achievement is somewhat simplistic and sterile in that it largely represents the achievement of a specific binary threshold, and there is no real incentive for an organization to hugely exceed any KPA. Also, because lower-level KPAs may be easier to achieve than some of the more abstract higher-level ones, software process improvement will progress at a diminishing rate as it proceeds to higher levels, perhaps to the discouragement of some organizations. However, as more organizations achieve CMM level 5, the future of the CMM will need to be considered.

Motorola found it more useful to concentrate on those critical success factors that reflect the richness of their process improvement program. These represent the tailoring of the process improvement program to the contingencies of the organization's development process. CSFs help push out the boundaries of process improvement and thereby form a richer base for continuing and future software process improvement. Some view creativity and software process maturity as mutually exclusive. Motorola rejects that view, and sees the coupling of their software process improvement program with CSFs as providing a basic infrastructure upon which creativity can be meaningfully nurtured. ❖

References

1. M. Paulk et al., "Capability Maturity Model for Software, version 1.1," *IEEE Software*, July 1993, pp. 18-27.
2. D. Zubrow, *The Software Community Process Maturity Profile*, Software Eng. Inst., Pittsburgh, Pa., 1997.
3. J.F. Rockart, "Chief Executives Define Their Own Data Needs," *Harvard Business Rev.*, Vol. 57, No. 2, 1979, pp. 81-93.
4. T. Butler and B. Fitzgerald, "A Review and Application of the CSF Concept for Research on the IS Development Process," *Information Systems—The Next Generation*, L. Brooks and C. Kimble, eds., McGraw-Hill, London, pp. 231-247.
5. H. Krcmar and L.C. Lucas, "Success Factors for Strategic Information Systems," *Information and Management*, Vol. 21, 1991, pp. 137-145.
6. J. Nandakumar, "Design for Success?: Critical Success Factors in Executive Information Systems Development," *European J. Information Systems*, Vol. 5, 1996, pp. 62-72.
7. D.D. Phan, D.R. Vogel, and J.F. Nunamaker, Jr., "Empirical Studies in Software Development Projects: Field Survey and OS/400 Study," *Information and Management*, Vol. 28, 1995, pp. 271-280.
8. C.V. Bullen and J.R. Rockart, "A Primer on Critical Success Factors," *The Rise of Managerial Computing*, J.F. Rockart and C.V. Bullen, eds., Sloan School of Management, Massachusetts Inst. of Technology, Cambridge, Mass., 1986, pp. 383-423.
9. F. Lau, "A Review on the Use of Action Research in Information System Studies," *Information Systems and Qualitative Research*, A. Lee, J. Liebenau, and J. DeGross, eds., Chapman & Hall, London, 1997, pp. 31-68.
10. M. Brassard, *The Memory Jogger. A Pocket Guide for Continuous Improvement*, Goal/QPC, Salem, N.H., 1994.

About the Authors

Brian Fitzgerald is senior researcher at the Executive Systems Research Centre at University College Cork, Ireland. He is actively involved in applied research projects in the areas of systems development approaches, foundations of the IS field, and executive information systems. He has more than 15 years of experience in the IS field, having worked in industry prior to taking up an academic position. He has published in *The Information Systems Journal*, *Information and Management*, *INFOR, the Journal of Information Technology*, and the *International Journal of Information Management*, among others. He is an associate editor for *The Information Systems Journal*.

Fitzgerald holds a PhD from the University of London.

Tom O'Kane is a senior staff engineer with Motorola, Cork, Ireland. He has over six years of experience in software process improvement and over 10 years of experience of software development in Motorola and with other US multinationals. At Motorola, his prime responsibility is managing the Cork software process improvement program utilizing the SEI CMM as the improvement framework. He works closely with other Motorola facilities both in Europe and the US, where he has acted as an internal process consultant. He regularly gives guest lectures on software process improvement and has written several internal publications on various aspects of it. He chaired the third Motorola European Software Engineering Symposium and was the first person to be presented with the General Managers Quality Award for Excellence for his work on process mapping.

Readers may contact Fitzgerald at the Executive Systems Research Centre, Room 321, O'Rahilly Building, University College, Cork, Ireland; e-mail bf@ucc.ie.