Defence Science Journal, Vol. 53, No. 4, October 2003, pp. 403-414

REVIEW PAPER

Software Process Management

Arundhati Bhattacharyya
Research Centre Imarat, Hyderabad–500 069

ABSTRACT

The paper gives an overview of the software process using the capability maturity model instituted by the Software Engineering Institute at the Carnegie Mellon University, Pittsburgh, PA and also briefly focuses on a sequence of maturity levels in the software process that can be observed in development organisations. The paper aims to express the need for process management and understand something of the conditions that determine where one is and where one can hope to be. An attempt is also made to provide a framework and some techniques for evaluating and improving the process of developing software.

Keywords: Capability maturity model, process maturity levels, key process areas, process assessment, software maturity framework, continuous process improvement

1. INTRODUCTION

The rapid advancement and wider spread of information technology today, is seeing the development of software applications ranging from trivial to the mission-critical. Reliability and predictability have become critical criteria for software deployment. The Defence projects today, are all software intensive and highly complex. Handling of such complex software in the present Defence scenario without a defined process is extremely difficult, if not, impossible. A well-defined software process management is essential to produce reliable and quality software within the stipulated project time schedules. The US DoD incurs heavy expenditure every year in the development, maintenance, and improvement of military software. In India, the Defence Departments are fast catching up with their US counterparts in this respect.

Given the extent to which software underpins all our everyday activities, software development has become a critical issue for modern organisations as well as for all of society. Software management is critical to an organisation's survival and success. Timely production of high quality software can significantly enhance an organisation's capability.

2. NEED FOR PROCESS MANAGEMENT

Today, large complex projects require the coordinated work of many teams. Methods applied for larger projects work equally well for the smaller groups and the individual professionals. Even the best professionals need a structured and disciplined environment in which to do cooperative work. Also, building software is not just a monolithic process, always the same. All this calls for the need of a more structured approach to software process management. It is also essential that at the individual level, one must be capable of assessing and grading one's current software process. One must be capable of knowing how to improve it and from where to make a start.

Re-revised 30 June 2003

In transforming the ad hoc, chaotic development environment into a smooth-running and, process-driven environment, a major culture change occurs, ensuring improved quality, reduced cycle time, and improved productivity. It also leads to an employee's satisfaction, which, in turn, produces successful programs. An inspired employee's performance is a powerful tool for an organisation. Motivated employees can work with unbound energy, enthusiasm, and create innovative solutions. Management should not overlook one of the most powerful tools they have–the highly satisfied and motivated employees. Some magic tool or method will not, by itself, make a significant improvement. Increasing software process maturity is an obvious and logical step in addressing the software crisis. Organisations that systematically implement improvements are more likely to become successful operators[1,2].

3. SOFTWARE PROCESS MANAGEMENT

The software process is a set of actions required to efficiently transform a user's need into an effective software solution. In simple words, process is how the work is done. The main reason for defining the software process is to improve the way the work is done. By thinking about the process in an orderly way, it is possible to anticipate problems and devise ways to either prevent or resolve these. Many software organisations find difficulty in defining and controlling this process, which is where these have the greatest potential for improvement.

A software development project is one in which a software product, to fulfil some needs of a customer, should be developed and delivered within a specified cost and time period.

3.1 Responsibilities & Objectives

Process management is a set of activities and infrastructures used to predict, evaluate, and control the performance of a process. The objectives of the software process management are to produce products according to a plan, while simultaneously improving the organisation's capability to produce better products.

At the individual level, the objective of the software process management is to ensure that the processes one operates or supervises are predictable, meet customer's needs and (where appropriate) are being continually improved. From the larger, organisational perspective, the objective of the process management is to ensure that the same holds good for every process within the organisation. The following four key responsibilities are central to the software process management[3] (Fig. 1):

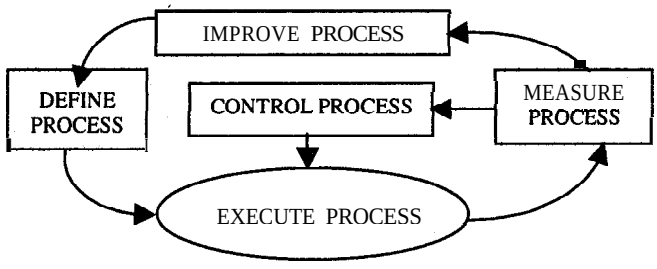


Figure 1. Four key responsibilities of process management

(a) Define the process, i.e., create disciplined and structured environment required for controlling and improving the process

(b) Measure the process; measurements being the basis for detecting deviations from the acceptable performance

(c) Control the process, ie, keeping the process within its normal (inherent) performance boundaries, ie, making it behave the way one wants it to

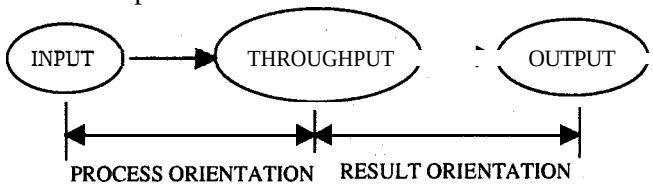
(d) Improve the process.

Execution, or rather adherence to the process is not a process management responsibility, but is the responsibility of project management.

The key concern of the software process management is how to identify topics that should be the focus for improvement and prioritise these. The basic principles of the process management are those of statistical control, which have been used effectively in many fields. When a process is under statistical control, repeating the work in roughly the same way will produce roughly the same results. To obtain consistently better results, it is thus necessary to improve the process and to have the process under statistical control for sustained progress. The basic principle behind statistical control is the measurement[4].

3.2 Software Development Life Cycle

The software development life cycle (Fig. 2) process consists of an input, a throughput, and an output[5]. By continuously monitoring the feedback from the software life cycle activities and plugging the deviation back into the process, it is possible to achieve high quality and highly reliable software products.



INPUTPARAMETERS	*PROCESS PARAMETERS*	*OUTPUT PARAMETERS*
• PEOPLE	• TRAINING	• PRODUCTIVITY
• MACHINES		• ZERO DEFECT
• STRUCTURE		• LIFE CYCLE
• MANAGEMENT		• REWORK

Figure 2. Software development life cycle process

3.3 Process Assessment

Organisations often launch software process improvement initiatives with a comprehensive process appraisal. The purpose of process assessment is to gain knowledge; it does not lead to any benefits by itself. The software process assessment helps software organisations to improve themselves by identifying their critical problems and establishing improvement priorities. An assessment is a part of the investment an organisation makes in the software process improvement. Some of the basic assessment objectives are: (i) to learn how the organisation works, (ii) to identify its major problems, and (iii) to enrol its opinion leaders in the change process.

4. FORMAL FRAMEWORK APPROACHES & CAPABILITY MATURITY MODEL

Formal framework approaches, such as the capability maturity model (CMM), Trillium, BOOTSTRAP, and SPICE promote improvement using the systematic processes and management practices for software engineering[6]. These approaches identify the best practices for the management of software engineering and provide methods for assessing an engineering organisation's maturity level, gauged by the staged adoption of specified processes. Such frameworks have basically three stages: (i) understand, (ii) control, and (iii) improve. Their final goal is a continuously improving the status of engineering organisation.

The software engineering institute's software-capability maturity model (SW-CMM) is one of the few branded methodologies that has had any effect on typical software organisations. It is a comprehensive and effectively realised model of the process maturity. The model attempts to quantify a software organisation's capability to consistently and predictably produce high quality software products. It provides a simple analytical tool for benchmarking an organisation's status and prioritising actions to improve that status. The capability maturity model presents a set of recommended practices in a number of key process areas that have shown to enhance software development and maintenance capabilities[1,7-10].

Though the capability maturity model has its own drawbacks, is rigid and has some harmful side effects along with its positive effect, it has become significant, chiefly because it defines a standardised software process. Increasing software process maturity' is an obvious and logical step in addressing software crisis. The best way to gauge the capability of an organisation is to observe how it behaves during a crisis, that is, when good practices are most important and that is when software professionals often have the least guidance.

The capability maturity model has had major impact on software organisations throughout the world. It is based on knowledge acquired through the software process assessments and extensive feedback from both the industry and the Govt. The capability maturity model was initially applied to the Govt and the military software development, but its use is now spreading to many industrial sectors as well. A comparison is unavoidable between the ISO 9001 model–a generic certification standard for software organisations and the capability maturity model, as both share a common concern for quality and deal with the process management. Though, both are driven by similar concerns and are intuitively correlated, but they differ in their underlying philosophies. However,

the capability maturity model is deeper and comprehensive than ISO 9001[11-14].

5. SOFTWARE MATURITY FRAMEWORK

In launching an improvement program, it is necessary to first consider the characteristics of a truly effective software process. It must be predictable in terms of cost estimates and scheduled commitments, and the resulting products should generally meet user's functional and quality expectations.

The software process maturity defines a framework for software process management that specifies some characteristics that the process must have to qualify as a process of some maturity. The primary objective of the maturity structure is to achieve a controlled and measured process as the foundation for continuous improvements. But this maturity structure is to be associated with an assessment methodology and a management system.

The management system assesses the position of an organisation and identifies its specific maturity status. This can help the organisation to concentrate on those items that will help it advance to the next level. Poor project planning generally leads to unrealistic schedules, inadequate resources, and frequent crises. Several factors like an ill-defined process, inconsistent implementation, and poor management can limit the effective use of software technology.

6. SOFTWARE PROCESS IMPROVEMENT

The first step in addressing software problems is to treat the entire software task as a process that can be defined, controlled, measured, and improved. For this purpose, a process is defined as a set of tasks that, when properly performed, produces the desired results. To improve their software capabilities, organisations must take the following steps:

(a) Understand the current status of their development process or processes

(b) Develop a vision of the desired process

(c) Establish a list of required process improvement actions in the order of priority

(d) Produce a plan to accomplish the required actions

(e) Assign/allocate the resources to execute the plan, and

(f) Start over at step (a).

For an organisation, however, the processes are more than a sequence of steps; these encapsulate the collective experience of that organisation capturing the past experiences in executing projects and enabling the organisation to leverage this experience in future projects, leading to process improvement. The CMM framework suggests a way for an organisation to gauge its progress. It roughly parallels the quality maturity structure and characterises the software process into one of the five maturity structure levels.

7. CAPABILITY MATURITY MODEL

The capability maturity model is a five-level model[15] designed so that capabilities at lower stages provide progressively stronger foundations for higher stages. The capability maturity model is based on principles of product quality given by Shewart[1], Deming, Juran and Crosby. The model describes an evolutionary path from ad hoc, chaotic process to mature, disciplined software processes.

The five maturity levels describe successive foundation for continuous process improvement. The levels define an ordinal scale for measuring process maturity and evaluating process capabilities. The levels also help an organisation to improve its efforts for developing the software in a better way. Understanding something of the conditions that determine where one is and where one can hope to be, is often the key to growth, for turning the comer from chaotic software to a more controlled and manageable process.

7.1 Process Maturity Levels

The capability maturity model's five levels of process maturity[1,16] are: (i) initial, (ii) repeatable, (iii) defined, (iv) managed, and (v) optimising.

At the initial level, software projects depend on the technical skills and often heroic efforts of specific individuals. These proceed in an ad hoc fashion from one issue to another. At the repeatable level, the focus is on establishing effective project management controls meant to enhance product quality and to improve the project's ability to set and meet reasonable time and budget commitments.

At the defined level, the improvement efforts concentrate on developing tailored software processes to be used throughout the organisation. At the managed level, the emphasis is on monitoring the software processes quantitatively and improving these to better meet product quality goals. Finally, at the optimising level, quantitative data are used consistently to improve the organisation's processes on an ongoing basis.

7.2 Key Process Areas

An organisation's current development process is evaluated against the key process areas of the capability maturity model, and the assessment provides the organisation with a list of actions that need to be taken to improve its development process[1,14,17,18]. The software engineering institute's level for a project or an organisation is defined as the level at which all associated key process areas are considered strength, i.e., all key process areas average scores must be equal, say, 7 on a 10-point scale or more. The key issues of each level that needs to be addressed to move to the next higher level, are given in Table 1.

Table 1. Key process areas of software capability maturity model

Level	Focus	Key process areas
5 Optimising	Continuous process improvement	• Defect prevention • Automation of software process • Technology change management • Process change management
4 Managed	Product and process quality	• Quantitative process management • Software quality management (SQM)
3 Defined	Engineering processes and organisational support	• Organisation process focus • Organisation process definition, establish software engineering process group (SEPG) • Training programme • Integrated software management, software configuration management • Software product engineering • Intergroup coordination • Peer reviews
2 Repeatable	Project management processes	• Requirements management • Commitment • Software project planning • Software project tracking and oversight • Software subcontract management • Software quality assurance (SQA) • Software configuration management (SCM)
1 Initial	–	• Competent people and heroics

Apart from the key process areas listed, training and change principles are the other important key areas that need to be addressed.

7.3 Training

Generally, large software teams contain a mix of talents. Work needs to be allocated based on the individual's capability. Though it is always desirable to have the best people, but it is always best to do with what is available. Besides, it is necessary to keep in mind the emotions, feelings, and the type of motivation required for the people involved. The three major components of creativity that need to be addressed are expertise, creative thinking, and motivation. It is, therefore, essential to continuously train people at various levels to bring out their full potentials.

7.4 Change Principles

A single factor, which stands out above all other key process areas, is the process change. The six basic principles of software process change are:

(a) Major changes must start at the top

(b) Ultimately, everyone must be involved

(c) Effective change requires a goal and knowledge of the current process

(d) Change is continuous

(e) Software process changes will not be retained without conscious effort and periodic reinforcement

(f) Software process improvement requires investment.

8. CASE STUDY

As a case study, the software development process of the light combat aircraft–digital flight control computer (LCA-DFCC) has been taken up. The LCA-DFCC software is quad redundant mission-critical software based on DOD MIL STD 2167A.

The development process started by generating the software requirements specification (SRS) based on the project requirements. The softwarae requirements specification was reviewed and analysed before the commencement of the design phase. Next, software development plan (SDP), software test plan (STP), and software quality program plan (SQPP) documents were generated. At this stage of software development, three core teams–teams for the design, testing, and quality were formed. Design reviews were conducted, followed by coding, testing, and integration. Quality was ensured at all stages of system design and integration. At each step of the development phase, RDD100 computer-aided software engineering(CASE) tools were used and quality checks were conducted. Appropriate metrics were used throughout the process. Standardised process was set for estimating, coding, and quality assurance. The entire phase of development followed the incremental model. All test results were documented and development folders were maintained for each software module by every designer of the module. These folders were then audited by the quality assurance team. Verification and validation were also done for every software module. Configured files like design files and test files were maintained for use by the software configuraion management (SCM) team. The fully developed software finally passed through an endurance test and is now functional.

9. A SUGGESTED APPROACH

On thes basis of the case study, an approach has been suggested for improving the process of developing software. Before any form of process management can begin, the status quo must be understood. Whatever be the existing process, no matter how ill-defined, it must be documented so that it can be used as a point of reference for process improvements. The main objective of process management should always be kept in mind at all times; to develop a software product to meet the customer's needs, maintaining the quality of the product, and meeting development time schedules. The starting point for selection of a new process improvement is to recognise the need with the overall goal of at least maintaining productivity, if not increasing it. Real-time adjustments to the process improvement may be needed to

keep the project working efficiently. To improve the process the five points to be followed are:

- Keep it simple
- Emphasise productivity
- Emphasise quality
- Emphasise cycle time
- Avoid process for its own sake.

9.1 Model the Existing Process

The first step towards process improvement is to define the process[19,20]. A process-definition document must .include information to be used by the management, SQA and SCM auditors (representing the customer) and the developers. The process definition must be reviewed and agreed upon by all the groups responsible for each phase. Generate elegant specification and automated analysis, understandable and usable definitions.

It is necessary to document the methodology that has been in vogue, define the software requirements specification of the project/organisation, standards used, the sequence of activities involved, the documents/reports available at various steps, and other such details. The software requirements specification serves as a central project document– a defining factor in relation to other elements of the project plan. It is also necessary to decide on some standards to be followed. The IEEE/EIA 12207 is one of the key integrating software engineering standards that provides a framework for developing and managing software[21]. Its main purpose is to establish a common framework for the life cycle of a software. It contains concepts and guidelines to foster better understanding and application of the standard and provides a simple model for integrating and coordinating the SEI CMMs for a comprehensive process improvement effort. Subsequently, at all stages of the development process/processes, use standards compliant to IEEE/EIA12207.

The adoption of standardised processes for estimating, coding, and quality assurance is an essential step towards continuous process improvement. Appropriate computer-aided software engineering tools like those based on unified modelling language need to be chosen to document the methodology. The use of computer-aided software engineering tools is essential for the entire development process.

Then, model the existing process. There are various forms of modelling, but if one is to make a start from the initial level, it is probably sufficient to record the life-cycle model in use and then make a PERT chart of typical activities. However, this would be very vague, since many interactions and common paths would simply not be remembered.

The software process model helps to capture three views: (i) functional–to define the tasks that make up the process, (ii) organisational–to identify the performers of those tasks, and (iii) behavioural– identify the development techniques needed to perform those tasks, check when the tasks are to be performed, and check out the relationships existing between all the tasks in the process. Besides the generation of the software requirements specification, it is also necessary to generate three more documents, ie, (i) software development plan, (ii) Software test plan, and (iii) software quality programme plan.

9.2 Assess the Existing Process

Once the status to the process has been accorded, it must be assessed. It is preferable to seek assistance from external independent consultants specialising in software assessments who can usually focus on weak portions of a software process based on the past experience. However, this activity turns out to be expensive both in terms of time and money. It is now that the software development plan and other plans are reviewed. The software development plan defines the software design. Ensure conceptual integrity, as this is the most important consideration in system design. The main objective is to identify the strong and weak points of the existing software process as a basis for the next step. The analysis process is necessary to ensure that the requirements

are complete and unambiguous as well as a way to determine what components are affected upon processing of a change. It also defines the characteristics of all possible solutions.

Process improvement is a continuous process. To be effective, software processes must be reviewed in terms of the entire development environment.

9.3 Identify Objectives

Any software process improvement programme must have a goal; the main choice being productivity and quality. Ultimately, productivity becomes a by-product of quality. Hence, quality standards need to be defined and suitable measurements/metrics should be adopted to implement the same.

A software measurement process is a systematic method of measuring, assessing, and adjusting the software development process using objective data. (Fig. 3). In defining issues for software measurement, consider risks, problems, and uncertainties.

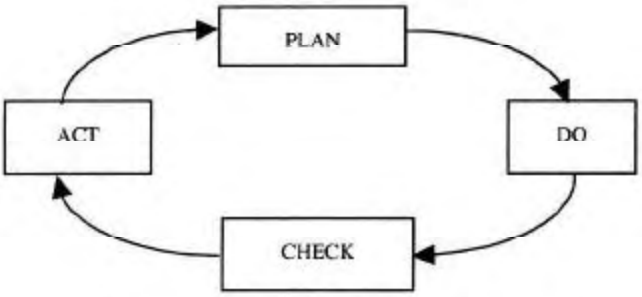


Figure 3. Software measurement process

The organisations that follow MIL-STD need to use four basic measures recommended by the DoD (Table 2), among the management tools for acquiring, developing, and maintaining software systems[4,22]. These measures are also recommended as the SEI core measures. These address important product and process characteristics central to planning, tracking, and process improvements.

9.4 Develop a Strategic Plan

Once the assessment has been made and the goals are agreed upon, the organisation can begin to develop a strategic software process improvement plan. The strategic alternatives are determined largely by choice between quality and productivity. There are three fundamental strategies: (i) build on present strength, (ii) focus on present weaknesses, (iii) try to maximise improvements.

Table 2. Department of Defence/Software Engineering Institute core measures

Unit of measure	Characteristics addressed
Physical source lines of code (PSLOC)	Size, reuse, rework
Logical source lines of code (LSLOC)	—
Staff hours	Effort, cost, resource allocations
Calendar dates for process milestones	Schedule, progress
Calendar dates for deliverables	—
Problems and defects	Quality, improvement trends, rework, readiness for delivery

Organisations driven by a need to improve quality must focus on the weakest parts of the existing process. Though the focus on weaknesses is a long-term commitment, but the eventual rewards are greater. To drive to this point at this stage, it is necessary to form three core teams (Fig. 4): (i) design team, (ii) testing team, and (iii) software quality program plan team.

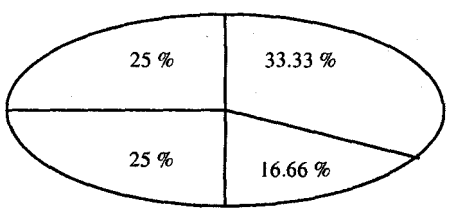
Establish testability and traceability metrics for requirement testing as a part of the total process. Traceability requirement is important for assessing and improving the process. Coding is done by the design team subsequent to the preliminary and critical design reviews. Code listings are to be passed over to the testing team modulewise, while the quality team ensures quality from the start of

Figure 4. Three core software teams

the system design to the final phase of software integration of all the modules.

The design-testing phase is an iterative loop. Test results are to be documented. All modules are finally integrated and tested. This is followed by the integrated verification and validation of the developed system software. Development folders must be maintained for each software package by every individual, which must be audited by the quality assurance team. Configured files like the design files and the test files are maintained for use by the SCM team. Schedule the software tasks[23,24] as in Fig 5.

No part of the schedule is so much affected by sequential constraints as component debugging and system test. Furthermore, the time required depends on the number and subtlety of the errors encountered. It is necessary to allocate more time for testing as delay always comes at the end of the schedule.



33.33 %: PLANNING FOR DETAILED SPECIFICATIONS
16.66 %: CODING
25.00 %: COMPONENT TESTS AND EARLY SYSTEM TESTS
25.00 %: SYSTEM TEST, ALL COMPONENTS IN HAND

Figure 5. Software tasks schedule

9.5 Define Success Criteria

Success criteria are the refinement of the overall goal of software process management. The top management plays a key role in process improvement. Without specific success criteria, an organisation can never know when a process improvement initiative has been successful. Success criteria depend on the strategy chosen, eg, choosing the requirements generation as process weakness. Success is defined say, by some quality metrics, that 99 per cent of the designers have used the design process methodology/technology identified in the next steps.

The Lynch and Cross's performance pyramid model[25] for business operating systems (Fig. 6) can be adopted by an organisation. An organisation works at various management levels. The highest management level of the organisation namely the corporate level, has to provide the vision, determine the overall framework, and also ensure that the spirit of the implementation is understood besides providing resources. Various entities are important at each of these levels, both from the external and the internal point of view. The pyramid identifies the prime concerns and objectives and metrics at each level.

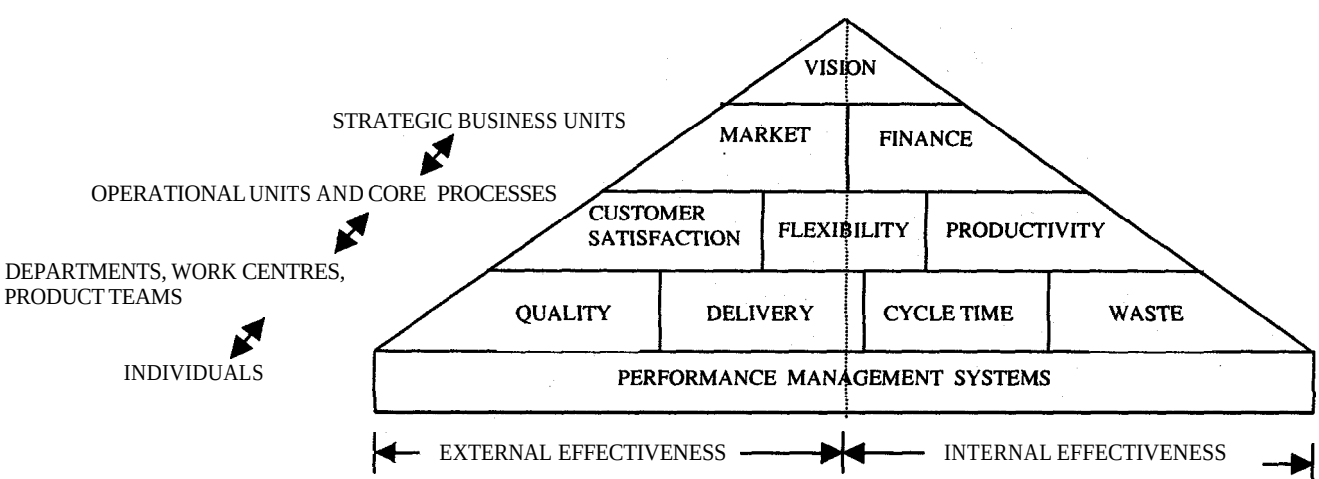


Figure 6. Performance pyramid

At the operational level, the core processes are finalised that set the directions for work centres. These decide what initiatives to be taken in the direction of achieving customer satisfaction, productivity, and bringing out flexibility in the organisation. At the individual level, it is how the individual's performance is managed in a systematic way so as to result in better quality of the product, on time delivery, waste minimisation, and cycle time improvement.

9.6 Investigate Technologies

The next step would be to identify appropriate tools and techniques. Examine the effect of application of the weak points across various sections of applications like system software, real-time software, embedded software, etc. Explore new technologies, which could eventually aid in process improvement.

It is also essential to attend technology-centric conferences and expositions, working with user groups and professional groups, and participating in software engineering courses at reputed institutes. This will provide an opportunity to meet and exchange views with the people from other organisations who are faced with similar problems.

9.7 Acquire & Introduce Technology

Technology acquisition has two parts: (i) methodology and (ii) tools. It is necessary to use tools at all levels of process development. Select appropriate software technologies to suit the need and appropriate tools to support these techniques. This decision however springs from the operating systems and the development languages to be used. Procure the right kind of computer-aided software engineering tools to suit the need. Performance simulators should also be used. The next step is to find a good source of technology training and use it.

9.8 Change Corporate Culture

The key is to focus on people realising the consequences of choices they make. Organisations should structure their policies by the kinds of consequences, such as positive or negative, immediate or future, certain or uncertain. This approach to corporate behaviour is extremely effective once new tools and technologies have been successfully introduced by defining the process. Besides, maintaining the integrity of all software work products is essential for any software development organisation.

9.9 Monitor Process Improvement

Once a process is started, it is necessary to ensure continuous process improvement. Establish Software Engineering Process Group (SEPG) for this purpose. The SEPG will monitor the process using suitable metrics and recommend changes for further progress.

9.10 Crux

(a) It is necessary to have flexible process techniques that have a balance of structure and creativity.

(b) Handle projects (2002–03) using task automation and repeatable techniques to achieve high productivity and tighter schedules.

(c) Emphasise on incremental and iterative development processes with the state-of-the art CASE tools.

(d) Process change will come about only if one uses the object-oriented approach, from the requirements generation to the coding level.

(e) Develop a historical projects database[26] (HPD) of all the released and closed projects, for data storage, metrics analysis and reporting, which are accessible to the entire organisation through the intranet. The project data is collected throughout the entire life cycle of a project. The historical projects database will contain project-related information, lessons learnt, work breakdown structure, metrics data, and project teams details.

(f) Maintain reusable software components library

(g) By continuously monitoring the feedback from the software life cycle activities and plugging the deviations back into the process,

it can be ensured that the quality of the software delivered to the customer is highly dependable and reliable.

However, given the costs these incur, process improvements must be undertaken with a view to return on investment. One could easily setup a high SEI-maturity organisation that would suffer from slowed delivery times and reduced productivity if the process was followed for process's sake. Thus, in addition to the traditional SEI CMM emphasis, one must tailor the processes and focus on cycle time.

Process improvements, as well as the rationale and expected benefits of the changes, should be communicated to all concerned projects and groups. The organisation should develop a deployment plan for the updated processes and monitor conformance to that deployment.

General survey shows that each level of capability maturity model improves quality by a factor of about 2. A project can hope to gain enough from a single well-chosen method improvement to repay the time and money invested in the change. Projects cannot realistically hope to accommodate more than one method improvement over their duration.

10. CONCLUSION

Capability maturity model is required to come up with updates taking into account the experiences from the last 10 years. Nevertheless, capability maturity model is definitely the standard that sets the pace. At present, it is the framework that allows benchmarking, and that contributes most to solving software engineering crises-just by defining standard terminology and condensing industry best practices. Nevertheless, SEI is working with the International Standards Organisation in its efforts to build international standards for software process assessment, improvement, and capability evaluations. This effort will incorporate concepts from many process improvement methods.

To make major process improvements, software organisations need an overall plan, some dedicated people, clear goals and management's commitment to these goals. As the software process evolves, new methods and better approaches will undoubtedly be found. To make progress, assign the job to some capable people and get started.

ACKNOWLEDGEMENTS

The author is grateful to the Director RCI, Hyderabad, for his immense support and help, the Director, Aeronautical Development Establishment (ADE), Bangalore and Shri K.A. Ramakrishna, Scientist G, Head, Software Engineering Division, ADE, Bangalore, for their support, suggestions, and guidance. The author also sincerely acknowledges Dr R.N. Biswas, Ex-Director, Computer and Information Centre, and advisor DRDL, Hyderabad, Dr A.L. Moorthy, Scientist-F, DRDL, Hyderabad, and the reviewers for their valuable suggestions, guidance, and support.

REFERENCES

1. Humphrey, Watts S. Managing the software process. Software Engineering Institute, Addison-Wesley Publishing Company, 1990.

2. Yamamura, George. Process improvement satisfies employees, *IEEE* Software, 1999, **16**(5), 83-85.

3. Florac, William A.; Park, Robert E. & Carleton, Anita D. Practical software measurement: Measuring for process management and improvement. Report No. CMU/SEI-97-HB-003, 1997.

4. Carleton, Anita D, et al. Software measurement for DoD systems: Recommendations for initial core measures. Report No. CMU/SEI-92-19, 1992.

5. Saraswathi, K. & Varanasi, Ashima. An approach to process improvements–NIIT experience.

6. Dutta, Soumitra; Lee, Michael & Wassenhove, Luk Van. Software engineering in Europe: A study of best practices. *IEEE* Software, 1999, **16(3),** 82-89.

7. Dion, Raymond. Elements of a process improvement program. *IEEE* Software, 1992, **9(4),** 83-85.

8. Jalote, Pankaj. Excerpts from CMM in practice. Better processes for better quality. Dataquest, 2000, **18**(3), 167-75.

9. Putnam, Lawrence H. Process improvement in organizations. Dataquest, 1999, **17**(20), 183-86.

10. Diaz, Michael & Sligo, Joseph. How software process improvement helped motorola. IEEE *Software*,1997, **14**(5), 75-81.

11. Paulk, Mark C. How ISO 9001 compares with the CMM. IEEE *Software*, 1995, **12**(1), 74-83.

12. Excerpts from ISO 9000, Vol. 2: The switchover. Dataquest, 1999, **17**(5), 194-98.

13. Paulk, Mark C. A comparison of ISO 9001 and the capability maturity model for software. Report No. CMUISEI-94-TR-12, July1994.

14. Parker, David. Quality assured in software. Dataquest, 1999, **17**(22), 173-75.

15. Abrachan, P.A. Quality–need of the hour. Dataquest, 1999, **17**(21), 174-79.

16. Pressman, Roger S. Software engineering–a practitioner's approach. McGraw Hill Edition, 2001.

17. Goldenson, Dennis R. & Herbsleb, James D. After the appraisal: A systematic survey of process improvement, its benefits and factors that influence Success. Report No. CMU/SEI-95-TR-009, 1995; ESC-TR-95-009, 1995.

18. Wiegners, Karl E. & Sturzenberger, Doris C. A modular software process mini assessment method. IEEE *Software,* 2000, **17**(1), 62-69.

19. Henry, Joel & Blasewitz, Bob. Process definition: Theory and reality. IEEE *Software*, 1992, **9**(6), 103-05.

20. Godbole, Nina. Software process improvement. Information Technology, 1999, **8**(3), 64-68.

21. Ferguson, Joan & Sheard, Sarah. Leveraging your CMM efforts for IEEE/EIA 12207. IEEE *Software*, 1998, **15**(5), 23-28.

22. Rozum, James A. Defining and understanding software measurement data. In Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA

23. DeMarco, Tom. The deadline–A novel about project management. Dorset Publishing House, 1997.

24. Brooks, F.P.(Jr). The mythical man-month. New York, Addison Wesley, 1995.

25. Sinha, Sushil. Improving organisation effectiveness through metrics-based processes. Texas Instruments (India) Limited, Bangalore.

26. Rao, Kiron K. & Shah, Bharat P. Continuous process improvement in software development. International Software Division, Blue Star Limited.

Contributor

Ms Arundhati Bhattacharyya obtained her postgraduation in Physics (Electronics) from the University of Hyderabad (Central University). She had also acquired a postgraduation engineering in Software Systems from the Birla Institute of Technology and Science (BITS), Pilani. She joined DRDO as Scientist B after completing one year Electronics Fellowship Course at the Institute of Armament Technology (IAT), Pune. She is currently working as Scientist E at the Research Centre Imarat (RCI), Hyderabad, in the field of avionics software. Her areas of interest include: Information technology and design of software systems.