Evaluating FakeCo, Inc.'s software maturity level using the Software Engineering Institute's Capability Maturity Model

MMIS 640

James T. Dollens, Ph.D.

Bryan P. Blue

Nova Southeastern University

December 5, 2004

Table Of Contents

Abstract

FakeCo, Inc. is based on a real software company, but the name has been changed for legal and ethical reasons. The software management team of FakeCo wishes to determine the Software Engineering Institutes (SEI) Capability Maturity Model (CMM) level. The CMM is broken down into 5 levels of expertise. A summary of the CMM that is addressed by using general indicators is used to determine FakeCo's current level. This is done instead of a full software audit as this would be beyond the scope of this paper and be too costly for a company of its size.

Introduction

As stated in Kit (1995), "…SEI CMM describes goals and activities that are of vital importance to anyone striving to improve their everyday testing process. It is also noted that there are few software organizations that are above Level 1 on the maturity scale". This may be misleading as Kit's information is going on 10 years old. The latest reports from the Software Engineering Institute (1994) shows that from appraisals collected from April 2002 through June of 2004, the reported peak now falls into level 2, or Managed maturity level.

FakeCo, Inc. is a real estate software company that is primarily concerned with the creation and maintenance of real estate publishing software. FakeCo is one of the oldest real estate magazine publishers in the nation and produce one of the nations largest bi-weekly homes magazines. Over the past 17 years they have developed their own advertising system and have been able to successfully roll the software out to other third parties. Although still successful, these rollouts have suffered periodic customer service nightmares due to lack of testing.

In order to improve the overall software development cycle, an understanding of the SEI's Capability Maturity Model is needed. From this understanding, a classification can be given to FakeCo, Inc. From there, appropriate steps can then be taken to attempt to improve the software process and testing. The goal of the paper is to evaluate the current status of the software company and then use this information to suggest improvements within the organization. A complete evaluation using true SEI appraisal methods is beyond the scope of this paper. Instead, the author will use the ideas of indicators as presented by Baumert and McWhinney (1992). These indicators and their associated definitions should be enough to classify FakeCo, Inc., although in an unsanctioned fashion.

Background

The Software Engineering Institute (SEI) was established in 1984 by Congress and is supported by the Department of Defense. It is a federally funded organization that is associated with Carnegie Mellon University. The SEI supports the creation of quality software by trained developers (Kit, 1995).

As shown in Kit (1995), the amount of time spent on management issues while testing can be related to the inverse of the maturity level. This can be explained by the fact that if the development team does not have a clear direction while testing, much of it's time will be spent trying to figure what to do next, rather than spending time on the actual testing procedures. Therefore, the lower the CMM level, the more of the software testing resources will be spent on management issues and not the main focus, development and testing.

The Capability Maturity Model (CMM) is a classification system that can rank software development organizations as to the their ability to create software using modern software engineering methods (Kit, 1995). The CMM is broken down into five different maturity levels.

Level 1 is the Initial level. In this level there is no real stable environment for development or maintenance of software. Performance can be determined at most on an individual basis. Few, if any, software processes have been put in place (Baumert, McWhinney, 1992).

Level 2 is the Repeatable level. Here, stable processes for planning and tracking software development have been put in place. Software managers are able to track costs, schedules and functionality. Project standards are identified and a quality assurance group makes sure the development team adheres to them. Problems in meeting commitments and deadlines are dealt with as they are found (Baumert, McWhinney, 1992).

Level 3 is the Defined level. At this level, there is an organization wide standard for developing and maintaining software. A software engineering process group, or SEPG, controls

this process definition and is responsible for determining where improvements can be made. Also, organizational wide training is supplied so that all staff and managers are ensured to have the skills they need to perform at their respective jobs. To improve software quality, a peer review system is also, setup (Baumert, McWhinney, 1992).

Level 4 is the Managed level. Quality goals within software development process are measured and maintained across the organization. A central repository of statistics is kept current for continuing analysis of key development processes (Baumert, McWhinney, 1992).

Level 5 is the Optimizing level. This level deals with constant process improvement. Weak areas can be determined and improved. Cost-benefit analysis of new technologies can be performed, as statistical data on process effectiveness is available (Baumert, McWhinney, 1992).

Consultants that are trained in the SEI's software process assessments (SPAs) perform the evaluation of a development organization. The SPA is centered on a set of questions about pre-defined subjects. As initial questions are asked, there are other related follow-up questions that can be asked to clarify previous answers. These questions have various weights that are then used to "score" the development organization and place them into a CMM level. The questions also help point the evaluation team towards key areas that improvement should be made by the development organization (Kit, 1995)

The benefit of being able to measure software development activities is key to the assessment process. This ability can also improve an organization by letting it better quantify tradeoff decisions, give a deeper insight into product development, give better control, planning and monitoring of the projects as well as many other capabilities (Baumert, McWhinney, 1992). Daumert and McWhinney are also quick to point out that due to half hearted commitments by upper management these benefits sometimes can never be achieved. The commitment has to be

more than a just a written policy, it has to be embraced by the organization. The measurement program has to be given the necessary resources to operate effectively.

Also, the measurement responsibility should be given to a dedicated group. The group is then responsible for determining the attributes to measure as well as maintaining the collection of measured values (Baumert, McWhinney, 1992).

The Capability Maturity Model is designed to help software developer team move towards a higher standard of excellence. It helps the team evaluate its' current position by comparing it against a known model. This information can then be used to determine a plan for improvement.

Indicators

Each level is based on a set of categories of indicators. Measurements can be made of these indicators and these results used to place an organization into a maturity level. It should be noted that the measurements by themselves is not enough to ensure that an organization is in a particular maturity level. The organization is responsible for other activities such as training, policies and implementation that are part of the respective level (Baumert, McWhinney, 1992).

Each maturity level has it's own set of indicators, but they should not be seen as discrete items. If the goal is continuous process improvement, all of the indicators should be reviewed initially and not on a level-by-level, as needed basis. To get a better picture of what is happening, indicators should also be reviewed together. Many times, groups of indicators are related, and by reviewing them all, a better picture of what really is going on can be seen. As a simple example, Baumert and McWhinney (1992) give the relation of cost, progress and training indictors. Viewing any one of the three may not give a complete picture.

Baumert and McWhinney (1992) classify the indicators into 13 different categories. They can be summarized as shown in Table 1.

Level 1 - Initial

As mentioned before, maturity levels are partially classified by the use of these indicators. Since Level 1, Initial Level, can be classified as “ordered chaos”, it does not possess any indicators. In fact, this can be used to partially identify a Level 1 development organization, the lack of use or availability of any indictor.

Level 2 - Repeatability

Level 2, Repeatable Level, can be thought of being evaluated on the progress, effort, cost, quality, stability and computer resource utilization indicators. In this level, the rudiments of project management have been put in place. Costs associated with the project plan as well as its schedule can be calculated. Other items such as quality assurance, configuration management and requirements management have also been put in place (Baumert, McWhinney, 1992).

The Progress indicator is measured based on project schedule activity completion. A measurement of the actual completion vs. the scheduled completion time can be used to evaluate the organizations commitment to the overall project plan. If there are large variations in the values, a problem is probably lurking within the organization.

This information can be collected at almost every level within the organization. From the details of individual tasks to the larger upper management summaries, each view of schedule adherence can be tracked. This information can then be used to determine the level of software project planning, tracking and oversight of even subcontract management. This indicator is also available by virtually all software management members (Baumert, McWhinney, 1992).

The Effort indicator allows the software manager to be able to determine the amount of staffing that is required. It allows the software manager to track the planned vs. actual use of personnel. It also allows for determining the appropriate allocation of workers based on the skills

they posses vs. the skills that are needed to perform the work. Again, this information can be collected at almost every level within the organization (Baumert, McWhinney, 1992).

The Cost indicator tracks the trends of forecasted costs vs. actual costs within the organization. This is used in conjunction with Effort and Progress to determine overall project costs based on employee skill usage and schedule constraints. This indicator is also available, at varying levels of detail, to all managers (Baumert, McWhinney, 1992).

The Quality indicator can be viewed as three separate indicators. These are quality assurance audits, life-cycle review results and trouble reports. Each of these three sources should be collected and then assembled into an indicator of quality for the organization (Baumert, McWhinney, 1992).

Quality assurance audits should be performed by an independent organization that specializes in this function. Its goal is to give the software managers an objective view of how well the organization is sticking with the procedures, standards and requirements of the organization.

Review results are also part of the quality indicator. These are formal reviews that make all participants in a project aware the current status and major issues. They are typically performed at the end of each major phase of a project and are sometimes linked with customer satisfaction or permission to continue with the rest of the project (Baumert, McWhinney, 1992).

Trouble reports can give a direct indication of the quality of a software product and development process. They can be recorded by quantity as well as by time period. For purposes of evaluation of a maturity level, the trouble reports are limited to the reports that are generated during the development life cycle and not after project completion. This gives feedback to managers on the effectiveness of testing and product reliability (Baumert, McWhinney, 1992).

Stability, like quality, can be broken down into two different indicators that together yield its' value. Stability can be broken down into requirements and size stability. Requirements stability

focuses on the number of changes that are made to the requirements of a project. As most software engineers already know, requirement instability can result in an increase in costs and project's length. Size stability is based on actual vs. predicted code size. It also can contribute to poor code quality. The code size is also a factor in the planning phases as a software manager needs to be able to determine the projects size in order to estimate costs, schedules and effort that will be required (Baumert, McWhinney, 1992).

Finally, Computer Resource Utilization is an indicator that shows how the software will impact system resources. This impacts all phases of development including design, implementation and testing. It ensures the organization has a handle on resource allocation and can predict when additional resources may need to be appropriated (Baumert, McWhinney, 1992).

Level 3 - Defined

Level 3 indicators can be summarized by progress, effort, cost, quality, stability, computer resource utilization and training. In level 3 the focus goes from project management, as in level 2, and moves towards organizational issues. It is important to note that the indicators of level 2 are already in place and will continue to be used. In level 3 though, measurements can be made that rate the organizations overall software development. The historical data that was collected while in level 2 can now be used to spot trends and determine when processes "go out of bounds."

Progress indicators are basically the same as in level 2, but with a history of values to access, an additional notion of "bounds" can be added. This additional information can help the managers determine when a project is going off track, making adjustments as needed to maintain the project plan (Baumert, McWhinney, 1992).

Effort, again, is essentially the same as in level 2. In level 3, more detail is added. At this level, higher level processes have been defined from level 2, now that management has level 2 experience, it can drill down to lower level activities as needed (Baumert, McWhinney, 1992).

Cost in level 3 can also pull from historical data collected from level 2 experiences. Previous projects that may have had the same scope as a current project can be evaluated on the basis of costs of the historical data to the present costs. As long as the costs are within the values of the previous projects, there is no real need for alarm. If the current project costs go outside the range that is determined from past experiences, it will require analysis into why with possible corrective actions (Baumert, McWhinney, 1992).

Quality indicators are essentially the same as in level 2. Quality assurance audit results, review results, trouble reports and peer review results can all benefit from being able to expand analysis to information about the quantity and quality of the processes themselves (Baumert, McWhinney, 1992).

Stability indicators are continued to be used as in level 2, but the history of these indicators can now be scrutinized. At this level, information on how many contract waivers have happened can be evaluated. Analysis into why these waivers happened can be done too. Patterns of frequent requirements changes can be found and used to improve the requirements phase of the development cycle. Size stability measures are essentially the same as in level 2, but with the amount of historical data collected, accuracy of forecasted size should increase (Baumert, McWhinney, 1992).

Computer resource utilization is essentially the same as in level 2. A new indicator at this level, Training, is one that will help managers account for effectiveness of training programs that are being offered. It can also be used to help identify the skills that the software development team possesses (Baumert, McWhinney, 1992).

Level 4 – Managed

Level 4 indicators are much the same as in the previous levels. One main difference is in the calculation of the range of acceptable control limits. In level 3, managers may use ad-hoc methods to establish an upper and lower bound based on historical data. In level 4, statistical methods are used to calculate the bounds based on the historical values (Baumert, McWhinney, 1992).

Progress and Cost is the same as in previous levels, but in level 4 the statistical bounds are calculated. Effort is the same as well, but new metrics may be introduced such as measuring experience levels based on architecture, application or tools, or even evaluating productivity vs. years of experience (Baumert, McWhinney, 1992).

Quality indicators are similar to previous levels as well, but at this level, noncompliance issues from process auditing should be approaching zero. Everyone should be well versed in the organizations software process. If noncompliance issues are not close to zero, this is a good level to determine why and correct the situation. Another difference in the quality indicators at this level is that indicators such as peer review results can now be analyzed statistically (Baumert, McWhinney, 1992).

Stability indicators are similar to previous levels. Requirements stability is essentially the same, but size stability benefits from the use of statistical analysis as in the other indicators.

Computer resource utilization and training are the same as the repeatable and defined levels respectively.

Level 5 – Optimizing

At level 5, the same indicators of progress, effort, cost, quality, stability, computer resource utilization and training are in place, but it can be noted there are no true indicators of

this maturity level (Baumert, McWhinney, 1992). These can be viewed as the expected indicators for this level.

Stability, computer resource utilization and training remain the same as in previous levels, but continue to be monitored. Progress and Effort are measured by evaluating the process improvement activities, defect prevention and innovation that occur to improve their respective values (Baumert, McWhinney, 1992). This can be evaluated by basing it on the statistical evidence that is collected from the process that was put in place during the previous level.

Cost is under a tighter control as at this level, all costs are known to the software manager. Using this information, whenever a change in technology is needed, a true cost/benefit analysis can be performed (Baumert, McWhinney, 1992).

Quality indicators are essential the same as in level 4 and prior. Software quality audit results, review results, trouble reports and peer review results are all maintained at this level. Defect prevention now becomes a key indicator. Zero defect should be the new goal whenever reasonable. If zero defects are not attainable, reducing common defects should be the goal. At this level, close monitoring of this indicator is performed (Baumert, McWhinney, 1992).

Analysis

In the 2004 study by the Software Engineering Institute, the majority (35.1%) of reported maturity levels can be classified as being in the level 2, or managed level. This is followed by the second largest group at 28.5% in the level 3 or defined level. It is not the intention of this paper to do a thorough analysis of FakeCo, Inc., but to take the ideas given by Baumert and McWhinney (1992) and develop an estimation of where FakeCo would be classified. The initial speculation would be that FakeCo has not progressed to level 2 and is still working on steps to get to that point.

By default, level 1 can be obtained by any organization. The organization has no stable development environment. Performance cannot be evaluated on an organization level. At most it may be possible to evaluate an individuals performance. These statements can be made about FakeCo without too much trouble. FakeCo holds infrequent meetings and members of the development team report their own status on projects. There is no central repository or project schedule. With no real comparison that can be made other than on memories of past experiences, true performance of the organization cannot be evaluated.

At level 2, the basics of software management should be in place. These can be evaluated through the use of questions and ideas that have been mentioned in this paper.

Appropriate questions that are shown in this paper come directly from Baumert and McWhinney (1992). These questions can be used to preliminarily evaluate the indicators for an organization that is in level 2, as given by Baumert and McWhinney. Questions that appear in their document and that do not have a tie to FakeCo's current development processes are not mentioned. Through the quick question and answer session given, it can easily be answered, "Has FakeCo achieved level 2 CMM status?"

Basic Progress Indicator Questions:

Q: Are the actual results and performance of the software project tracked against the software development plan?

A: FakeCo does not have a software development plan therefore tracking is not possible.

Q: Are the schedules of activities for the project consistent with each other and with the software requirements and the software development plan?

A: Again, FakeCo does not have a software development plan therefore scheduling consistency cannot be evaluated.

Basic Effort Indicator Questions:

Q: Has the software planning process resulted in a staffing profile that is consistent with the planned schedule and budget?

A: FakeCo does not have a software plan therefore staffing profiles and budgets cannot be calculated.

Q: Will the types of effort being applied to the contract have an impact on the quality of the final product?

A: Again, without a plan, FakeCo cannot really evaluate this question.

Basic Cost Indicator Questions:

Q: Is the planned budget revised according to established procedures when the actual results and performance data indicate a re-planning is necessary?

A: FakeCo does not have planned budgets for a given software project. If they do exist, they are not given to software management. Upper management controls all aspects of costs and budgets. Software management has no control or insight into this aspect.

Q: Has the software planning process resulted in a software development plan in which the planned budget is consistent with the effort and schedule required to develop the software and the software requirements?

A: FakeCo, without a software plan and access to budget information cannot hope to answer this question.

Basic Quality Indicator Questions:

Q: Are standards and procedures applied on the software project?

A: FakeCo is only now starting to introduce company wide software code standards.

Q: Are project personnel applying the standards and procedures correctly?

A: There is no formal audit procedure to ensure that all new code is written to these standards.

Q: Are the noncompliance issues being addressed in an appropriate manner?

A: Without an audit procedure in place, FakeCo cannot address noncompliance issues.

Q: Is the review process being followed?

A: FakeCoe does not perform formal review processes. It does perform basic, ad hoc reviews at major project completions.

Q: Is the testing activity complete?

A: Testing, in a formal sense, does not exist at FakeCo. There are not even peer reviews of code in place.

Q: Does the quality of the product indicate that the product is ready for release to the customer?

A: Since testing and quality control do not exist at FakeCo, it is up to each programmer to determine if their own effort is enough to release. This does not work well.

Q: Are project personnel addressing the trouble reports in a timely manner?

A: No statistics are being kept on trouble reports. Problems arise and are fixed as needed. FakeCo works more in a “fire stomping” mode.

Basic Stability Indicator Questions:

Q: Is the number of changes to the requirements manageable?

A: The number of changes to requirements is not tracked. FakeCo cannot answer this question.

Q: Is the number of changes to requirements decreasing with time?

A: Without tracking the number of changes, there is no way FakeCo can answer this question.

Q: Is the number of to-be-determineds (TBDs) decreasing with time, that is, are the TBDs being resolved in a timely manner?

A: TBDs only exist in notes and task lists that are maintained by individuals. There is no way to determine if the quantity is decreasing.

Q: Is the estimated productivity sufficient to allow the completion of added code on schedule or is more staff required?

A: FakeCo does not contain productivity estimates or ways to track productivity. Again, ad-hoc methods are used to determine resource allocation. Since there is no true schedule other than a final deadline, resources are often flip-flopped from project to project as needed.

Basic Computer Resource Utilization Indictor Questions:

Q: Are the actual values of the computer resources within the allowable limits?

A: Computer resources for software development are not formally tracked at FakeCo. Basic computer usage is tracked in the form of CPU usage graphs, but memory and I/O management is not formally defined. Decisions are made based upon maximums that are determined from snapshots of the system when usage is extreme and performance begins to suffer. Adjustments are then made.

Q: Should more computer resources be acquired?

A: Acquisition is very simple at FakeCo, most of the time upper management does not stop computer resource acquisition. The problem is that no formal method for forecasting future acquisitions is in place. Only after alternatives in optimizing software and hardware settings is tried is it finally determined that new hardware is needed. There is no history of performance other than CPU graphs and Query/Sec graphs to base the decision on. This data only extends back 1 year and does not give a clear picture of what is going on in the long term.

The answers to these questions as Level 2 indicators should clearly point out that FakeCo has not achieved Level 2 of the CMM. Questions were omitted due to their redundant answers and it should be easy to see the general software processes that FakeCo uses (or does not use) from just the samples given above. Since FakeCo has not reached Level 2, it must still be in Level 1.

It should be noted that this rudimentary analysis would not be applicable to any organization that had progressed to level 2 or beyond. This analysis is enough to show that

FakeCo is not up to level 2 standards in many ways. This is enough to rule out higher levels within the CMM as well. FakeCo needs to improve its basic software development cycle in order to achieve Level 2, the repeatable level.

Results

It should be apparent at this point that FakeCo requires many improvements in order to succeed in the future. The company cannot realistically make the next jump from being a software provider from hundreds to thousands of users if it does not start now to clean up and formalize its software processes.

One initial recommendation is the use of project scheduling. Nobody at FakeCo has a formal education in the use and maintaining of project schedules. FakeCo could benefit from training in this area. This would then turn into an invaluable tool that can be used for determining realistic project schedules. This is a starting place for progress, effort and cost indicators. Through the use of Gantt and PERT charts, project milestones and regular evaluations a current view of the software development cycle can be created. FakeCo would not be limited to only evaluating one-team members performance. The performance of the team as a whole could be seen.

Another recommendation would be the installation and consistent use of trouble reports. FakeCo has used various software products to try to accomplish this over the years, but no financial resources have ever been placed into this concept. Various freeware products have been used, each with their own level of success. Part of the failure can be attributed to lack of an upper level management decision that this would make this mandatory. When the tracking system was in place it was only partially supported within the software development team. Organizational support for a system such as this would go a long way in keeping trouble ticket tracking a reality.

A third recommendation for FakeCo would be the introduction of more computer resource monitoring. Tracking CPU and network throughput in a graphical form that does not lend itself to analysis is not a good practice. A better solution would be a tracking database system that could record values that can be viewed and analyzed by user defined time frames. This would also allow for tracking of long term patterns and keep a history for more than the last twelve months. This information could then be used to predict future events such as the time remaining before a system becomes over utilized and needs replaced or upgraded.

Summary and Conclusion

The Software Engineering Institute's Capability Maturity Model can be a tool that can help improve the software development cycle. Although it may seem fairly complex and prohibitive for a small company to use, lessons taught in its' methods can be used to improve an organization without a full-scale appraisal. This paper shows one example where a concept of indicators can substitute for an in-depth evaluation of a software company. This may be a viable alternative to any company that may be interested in seeing where it may stand within this model. Although not conclusive, this approach could be used to get a general feel for software development maturity.

Baumert and McWhinney created an overall picture of a complex model using indicators and questions that explain the core properties of each level of the CMM. This is a great starting point for anyone who wants to explore how the software appraisal works, without getting too technical. It can give a general feel for how an organization will score, but can in no way replace true software audits by a qualified consultant group.

Indicators such progress, effort, cost, quality, stability, computer resources and training that are mentioned for software engineering also apply to many other engineering disciplines. This gives credibility to their use as true benchmark measures.

The idea of indicators was enough to analyze FakeCo and determine what they are not. As it was mentioned before, the same analysis would not hold true in determining that a company had achieved a maturity level. For this paper, it clearly points out areas that FakeCo needs to investigate to improve their software development cycle. It is also enough to show that FakeCo has not made it to Level 2, the repeatable level.

It is also apparent that there are some basic ideas that can be learned from the concepts of the indicators. The indicators point towards systems that are in place within an organization. The presence of the system or method indicates a positive result for the indicator. This information can be used to point FakeCo in the right direction. In those occurrences where FakeCo did not have the supporting system in place, it can work towards improvements to the organization that will put these systems in place.

FakeCo would benefit from the addition of a more structured software development schedule. Much of what constitutes a Level 2 rating starts with this concept. Trouble report recording and tracking would be good system to put in place. Quality and other metrics can be partially determined by this system. Also, computer resource tracking would be another area to pursue. FakeCo has part of this system in place and it would not take too much to extend the current system into something that would meet the Level 2 requirements.

Tables

Table 1 Categories of Indicators

Category	Description
Progress	How the project is performing in respect to schedule commitments.
Effort	Visibility into the contribution that staffing has on project costs, schedule adherence, and product quality.
Cost	Tracking actual costs against estimated costs and predicts future project costs.
Quality	
Software Quality Assurance Audit Results	Estimation of product quality and compliance of staff to project processes.
Review Results	Status of action items from life-cycle reviews.
Trouble Reports	Insight into the quality of the product and processes and the effectiveness of testing.
Peer Review Results	Insight into the quality of the intermediate and final products and into the peer review and development processes.
Defect Prevention	Insight into the cause of defects and the effect of defect prevention activities on defect insertion rates.
Stability	
Requirements Stability	Visibility into the magnitude and impact of requirements changes.
Size Stability	Insight into the completeness of the requirements and capability of the staff to complete the project within current budget and schedule.
Process Stability	Insight into the effectiveness and quality of the defined process.
Computer Resource Utilization	How well the project is performing in respect to its computer resource

Category	Description
Training	Visibility into the effectiveness of the organization's training program in meeting skill requirements.

Note. From "Software measures and the capability maturity model," by J. Baumernt and M. McWhinney, 1992, p. 36. Copyright 1992 by the SEI. Adapted with permission.

References

Baumert, J., McWhinney, M. (1992, September). *Software measures and the capability maturity model*. Carnegie Mellon University. Retrieved November 11, 2004, from http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.025.html

Kit, E., & Finzi, S. (Eds.). (1995). *Software testing in the real world: Improving the process.* Great Britain: Addison-Wesly.

Software Engineering Institute. (2004, August). *Process Maturity Profile*. Carnegie Mellon University. Retrieved November 23, 2004, from http://www.sei.cmu.edu/sema/pdf/CMMI/2004aug.pdf