Where is the Return on CMM?

Termpaper

ICS 225 Software Processes – Fall 1997

Joachim Feise

1. Introduction

Many companies and organizations use the Capability Maturity Model (CMM) as a vehicle for software process improvement. These organizations have spent large amounts of money as well as considerable effort in implementing the management practices prescribed by the CMM. Despite these investments, these organizations have often not the faintest idea if there is any return on their investments.

Other organizations are reluctant to implement a CMM-based software process improvement program because the upper management does not want to commit to invest considerable efforts into process improvement without evidence of positive effects to the corporate balance sheet.

As Howard Rubin observed, "Although much is written about the topic in qualitative terms, little quantitative information is available. In many ways, the engineering process is an informational 'black hole' - it draws in money and resources like a magnet but little data emerges."

In addition, companies of smaller size are concerned that they can not afford the up-front costs of implementing a software process improvement program.

In this termpaper I intend to report on studies in industry and government organizations that try to quantify the return on investment for software process improvement.

2. The Capability Maturity Model

In this section, I give a short overview of the Capability Maturity Model for Software (CMM), as well as the history of the CMM.

2.1. History of the CMM

In 1987, the US Department of Defense (DoD) established the Software Engineering Institute (SEI), an affiliate of the Carnegie-Mellon University in Pittsburgh, PA, to develop a means to evaluate the software development capabilities of the companies bidding for DoD contracts.
In response to the DoD's request, the SEI developed a questionnaire to measure the software process maturity of organizations. The questionnaire consisted of a five-level framework. In Subsequent work the SEI evolved this framework into the five levels of the CMM.
According to the IEEE, a process is "a sequence of steps per-formed for a given purpose." The SEI expands the notion of a process to define a software process as "a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products, e.g., project plans, design documents, code, test cases, and user manuals."
The five-level CMM model, which is shown in figure 1, identifies the extent to which a software organization has adopted and institutionalized the application of process engineering concepts, techniques and practices to monitor, control and improve the software process.

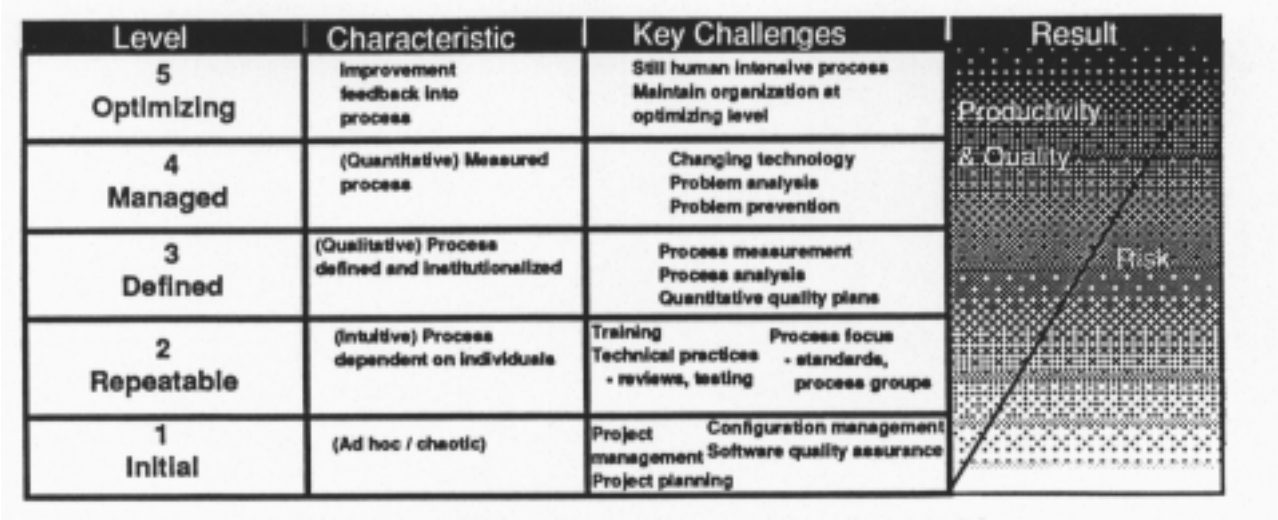


Figure 1: The SEI Capability Maturity Model

2.2. The CMM levels

The initial level, level 1, describes organizations that have an ad hoc, chaotic process. Organizations at this level usually don't follow formalized procedures,

cost estimates or project plans. Even if these formalized project control procedures exist, the management does not enforce the use. Successful projects in organizations at this level typically hinge on the efforts of a dedicated team.

Organizations at the repeatable level (level 2) use the fundamental project control mechanisms of project management, product assurance and change control. Organizations at this level can draw on their experience with similar work (hence the term repeatable), but there are high risks of breakdowns of the project control mechanisms when new, unknown challenges arise.

At the defined level, level 3, organizations are able to collect process data, and to use this data to examine the process and aid in the decision of how to improve the software process.

Level 4, the managed level, builds on the data collection that was established at level 3 of the CMM. At this level, organizations have established quality and productivity measurements for each key task. Organizations should also have a process database in place, together with the resources to manage and maintain it, and to analyze the data and assist in the use of the data.

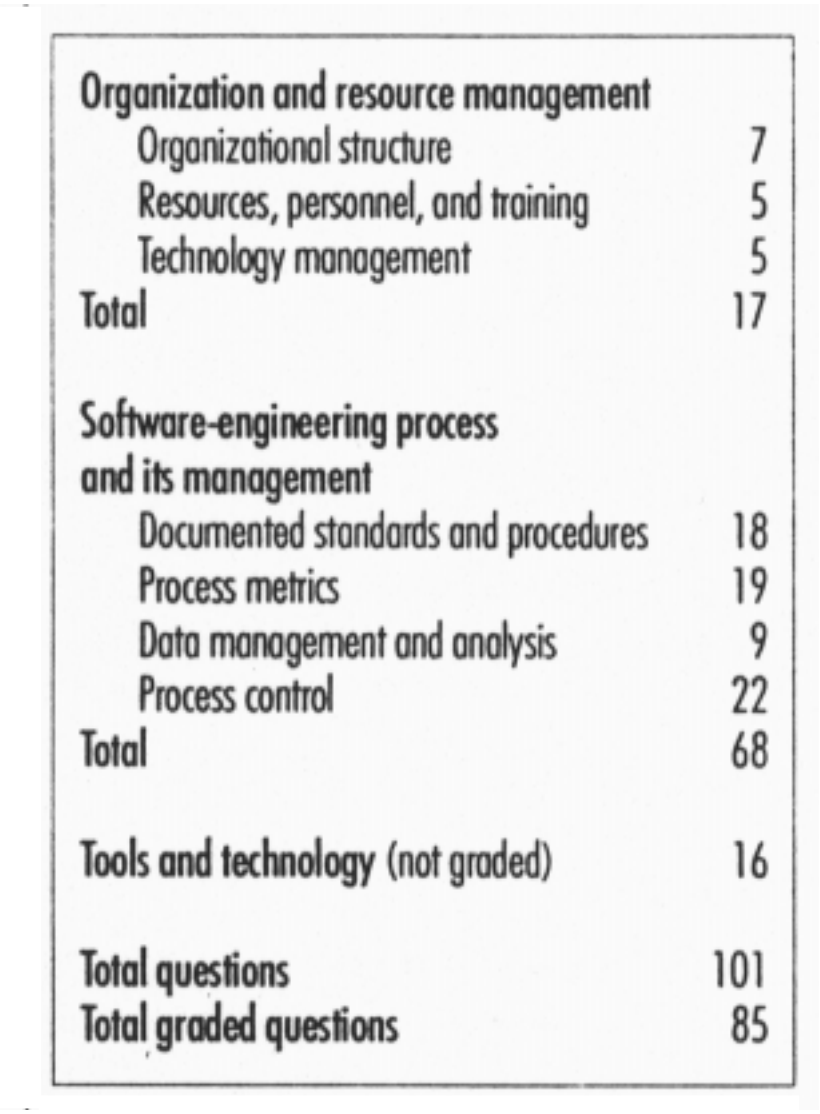
At the optimizing level (level 5), organizations are able to analyze the software processes and identify the weakest process elements and take actions to improve them. The data collected allows rigorous defect cause analysis and defect prevention.

According to Humphrey, et. al. (1991), "these maturity levels have been selected because they do the following:

- Reasonably represent the historical phases of evolutionary improvement of actual software organizations,
- Represent a measure of improvement that is reasonable to achieve from the prior level,
- Suggest interim improvement goals and progress measures,
- Make obvious a set of immediate improvement priorities, once an organization’s status in this framework is known."

2.3. The Software Process Maturity Questionnaire

The SEI process-maturity questionnaire consists of a set of yes/no questions on a variety of software engineering and process issues. The questionnaire is designed to facilitate reasonably objective and consistent assessments of software organizations.
The questions in the maturity questionnaire cover three main areas (see figure 2).



Organization and resource management	
Organizational structure	7
Resources, personnel, and training	5
Technology management	5
Total	17
Software-engineering process and its management	
Documented standards and procedures	18
Process metrics	19
Data management and analysis	9
Process control	22
Total	68
Tools and technology (not graded)	16
Total questions	101
Total graded questions	85

Figure 2: Maturity questionnaire subjects by topic (from Bollinger et. al., 1991)

- Organization and resource management.
  This area is concerned with functional responsibilities, personnel, and other resources and facilities.
- Software engineering process and its management.
  This area measures the scope, depth and completeness of the software engineering process and the way in which the process is measured, managed and improved.
- Tools and technology.
  In this area the tools and technologies used in the software engineering process are considered. The questions in this area help determine the effectiveness with which the organization employs basic tools and methodologies.

According to Bollinger et. al. (1991), the questions from the tools and technology area are not used in determining the organization's process maturity level. The questions form seven hurdles, each of which can be seen as a small true/false test for a specific set of questions (see figure 3).

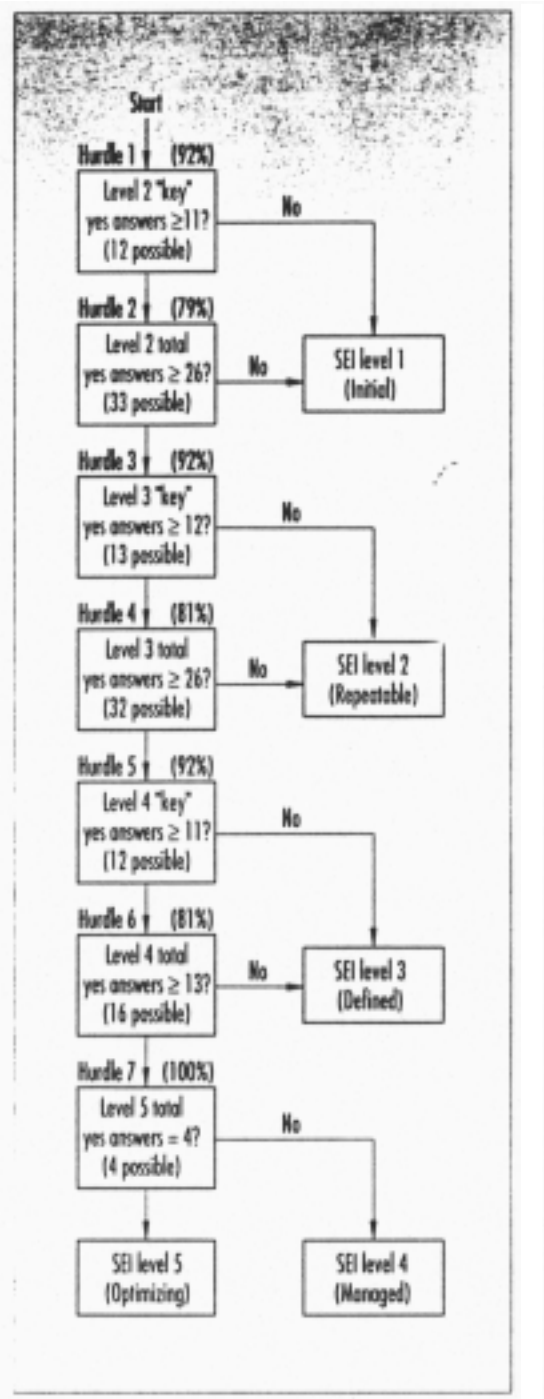


Figure 3: Grading of SEI Questionnaires (from Bollinger et. al., 1991)

As can be seen from figure 3, the CMM Level 1 is really not a level at all, but a failing grade since it requires no effort whatsoever to reach that level. Even answering all questions with no leads to a level 1 rating.

As pointed out by Humphrey & Curtis (1991), the questionnaire is not the sole means used in evaluating organizations. They note that when properly trained evaluators look for evidence of sound process practices, they have no difficulty identifying organizations with poor practices. They assert that well-run software processes leave a documented paper trail, like change-review procedures, approval documents, and minutes of control-board meetings that less competent organizations are unable to emulate.

3. Company Studies

In this section, I am summarizing several studies and company reports on the return on investment on process improvement that appeared in the literature since the CMM was introduced.

3.1. Process Improvement at Raytheon (Dion, 1993)

Raytheon's Equipment Division started with a process improvement program in the division's Software Systems Laboratory in 1988, after performing a self-assessment using the CMM. This self-assessment resulted in a rating at the initial CMM level, level 1.

The process improvement program focussed on the areas where the self-assessment identified deficiencies: policy and procedures, training, tools and methods, and process metrics.

Raytheon developed a process improvement paradigm that is based on a three-phase cycle of stabilization, control and change (see figure 4).

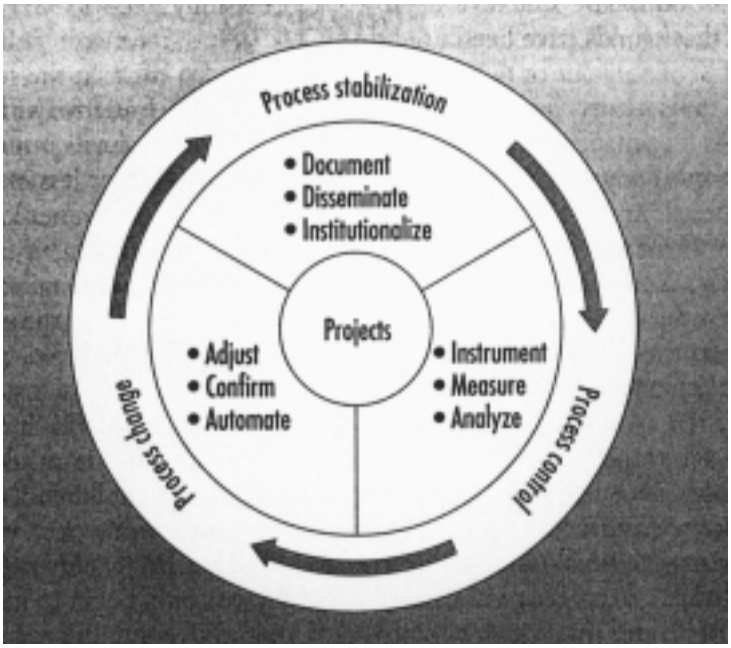


Figure 4: Raytheon's process improvement paradigm (from Dion, 1993)

The process stabilization phase is concerned with distilling the elements of the actual process used and institutionalizing it across all products, therefore achieving visibility and providing repeatability.

The process control phase shifts the emphasis to the gathering of significant data by instrumenting projects and to the analysis of the data to aid in understanding how to control the process.

The process change phase puts the emphasis on determining how to adjust the process, based on the results of the measurement and analysis of the previous phase, and how to diffuse the new methods among the engineers.

The improvement is continuous, so the next cycle begins with the newly improved process.
During the approximately five years the process improvement program was underway before the date of the report, about $1 million per year were invested into it. This investment moved the company to level 3 of the CMM.
However, getting management approval for continuing investments in this height required quantitative data to show a return on the investment that justifies such a large expenditure.
Raytheon's approach to quantify the benefits of process improvements categorizes the costs associated with a process as

- Performance: the costs associated with doing it right the first time.
- Appraisal: the costs associated with testing the product to determine if it is faulty.
- Rework (nonconformance): the costs associated with fixing defects in the code or design.
- Prevention: the costs incurred in attempting to prevent the fault from getting into the product.

The sum of appraisal, rework and prevention costs is called "the cost of quality." In three studies in 1990 and 1992 Raytheon analyzed the data of 15 projects, all "reasonably large real-time systems." The analysis indicates a savings of about $15.8 million in rework costs from the start of the process improvement initiative to the end of the third study in 1992 (see figure 5).

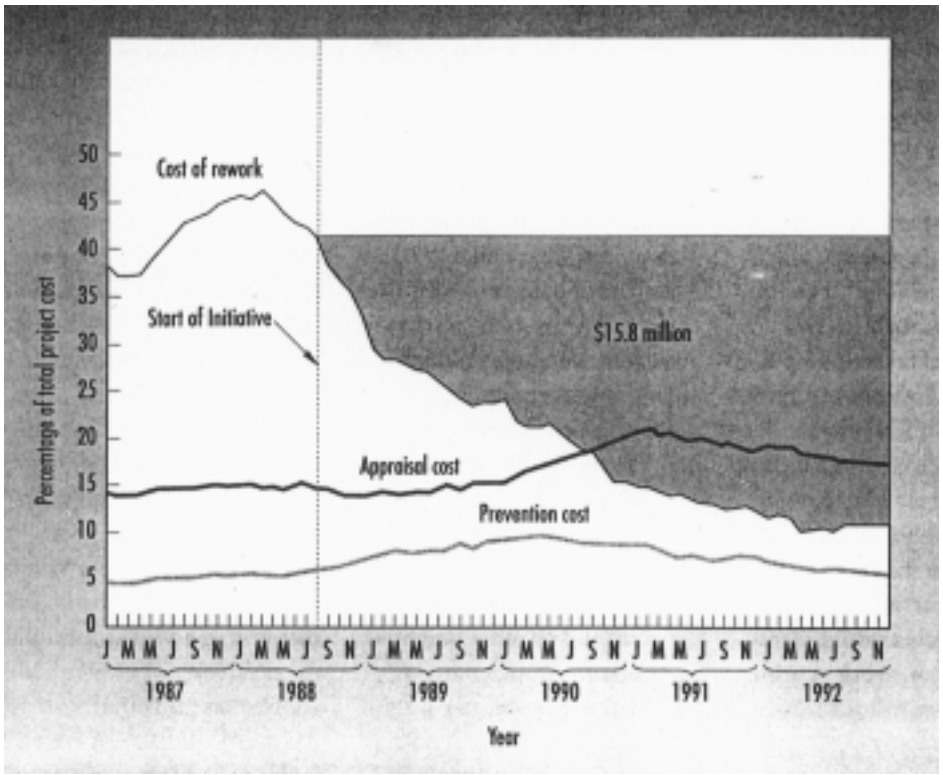


Figure 5: Raytheon Savings analysis

Raytheon attributes the raise of the approval costs mainly to a 30-percent decrease in total process costs, which gives the appraisal cost a proportional push as a percentage of the total costs. However, the figure does not elaborate on the total project costs.

As figure 5 shows, the rework costs shrank from 41 to 11 percent during the study period. In analyzing these savings Raytheon found that the cost of fixing defects found during the design and code phases rose slightly, mainly because formal inspection replaced informal reviews. The largest contributor to the rework savings was found to be the cost associated with fixing defects found during integration, which decreased to 20 percent of its original value. Raytheon attributes these savings to the design and code inspections, procedures, training and requirements stability.

The cost of retesting also decreased considerably, to about half its original value. This cost reduction is attributed to the fact that fewer problems were found during the first test, which in turn is a direct effect of removing design and code errors during inspections.

Raytheon also obtained productivity data on the projects in their analysis. The data was measured in terms of "equivalent delivered source instructions per man-month", which is used within Raytheon to capture the engineering effort involved in developing the product. It is basically a measurement of source-code size, which modified and reused lines of code being weighted according to the relative effort (as compared to new code) of modifying or reusing it, including the related documentation. Raytheon acknowledges that this is not scientifically accurate because of variations among projects, but it is a measure widely accepted by managers and developers. Figure 6 shows the productivity data.

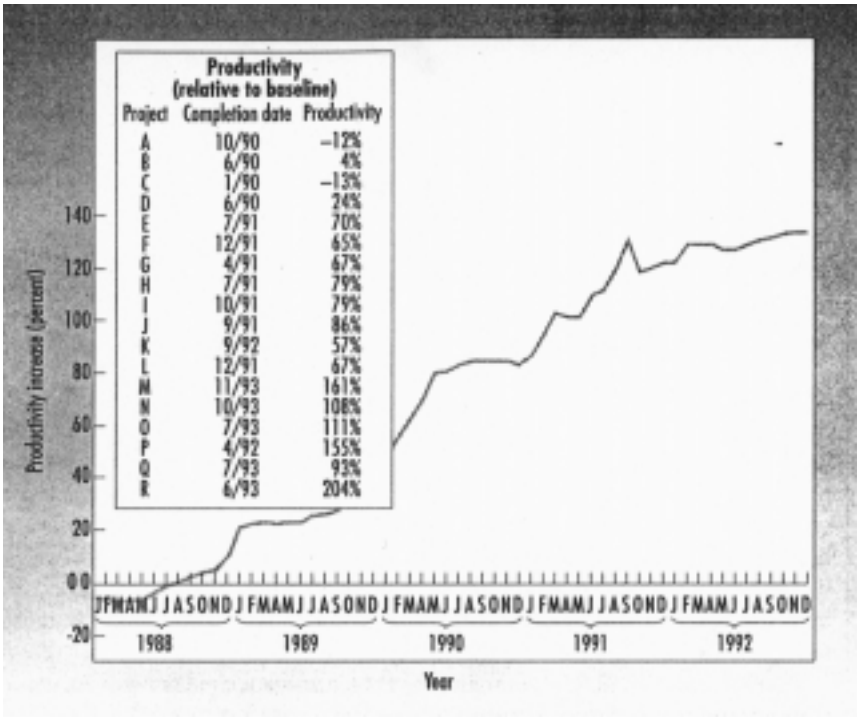


Figure 6: Productivity increase at Raytheon

3.2. Process improvement at Motorola (Diaz, Sligo, 1997)

In November 1995, Motorola's Government Electronics Division was assessed at the CMM level 4. The assessment rated the policies and procedures at level 5. Motorola uses quality, cycle time and productivity metrics to evaluate their development programs. Also, each project performs a quarterly self-assessment, in which any score for the Key Process Areas as defined in the CMM that falls below 7 is considered a weakness.

Quality. The quality metric as used my Motorola is defined as defects per million earned assembly-equivalent lines of code. A defect is defined as a problem that escapes detection in the phase it was introduced; earned assembly-equivalent lines of code is a measurement of the source code instructions, adjusted to the percentage of project completion.

The project results show that on each level of the CMM the quality improves by a factor of about 2 (see figure 7).

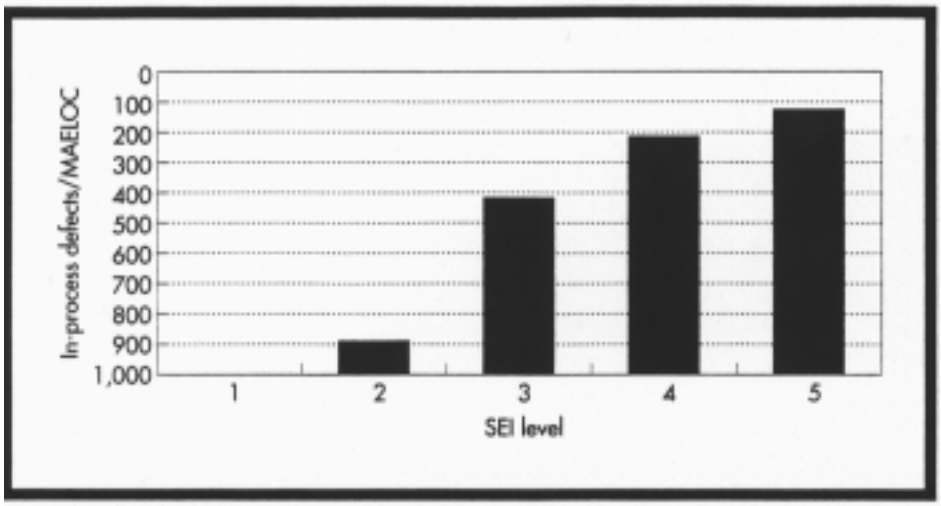


Figure 7: Quality by CMM level at Motorola GED

The improvement in quality from level 2 to level 3 is largely attributed to the introduction of peer reviews at level 3. Similarly, the quality improvement from level 3 to level 4 is due to the metric data collection at level 4. The data analysis at level 5 causes another jump in the quality of the product.

Cycle time. The cycle time metric is calculated by dividing the calendar time for the baseline project by the time required for the new project. So if the baseline project took six months to complete, and the new project took two months, the cycle time or X factor would be 3.

As in the Raytheon study, the selection of the appropriate baseline project could not be done with scientific rigor. At Motorola, each project tracks its cycle time by selecting a project with a similar target domain that was completed prior to 1992, and by tracking its progress against that baseline project. The cutoff date of 1992 was apparently chosen because Motorola moved from level 2 to level 3 of the CMM in that year. The paper does not explain the origin of the cycle time for the levels 1 and 2, which Motorola had reached before the cutoff date. Motorola also

adjusts the baseline data to eliminate the effects of a qualification test cycle that typically is part of government contracts as well as the effects of government funding changes on a project. This way, Motorola asserts that the data is more accurately comparable to data from non-governmental contracts.
Analysis of the cycle time data in figure 8 shows that overall, the cycle time increases with the transitioning to higher levels of the CMM, reflecting the underlying assumption that higher-maturity projects have a higher probability to be completed within schedule.
The sole exception is the move from level 2 to level 3 of the CMM, which, according to Motorola, may indicate a weak correlation between schedule performance and maturity level.

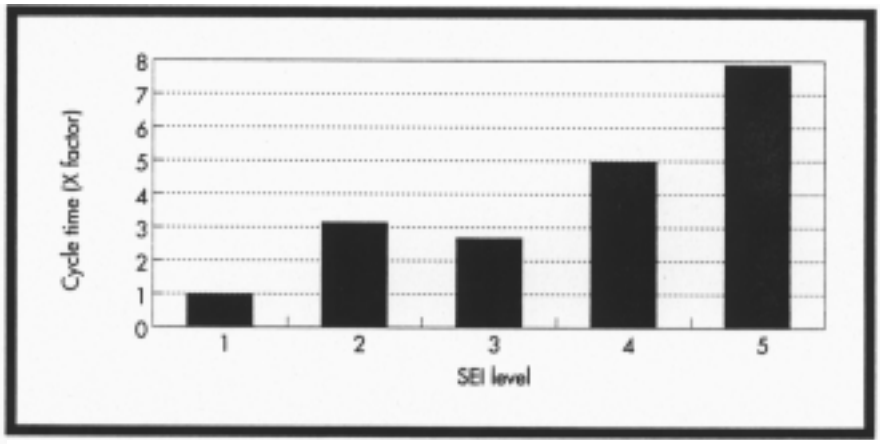


Figure 8: Cycle time by CMM level at Motorola GED

The paper notes that this data is of preliminary nature, since many of the projects used as basis for the analysis are still in development, so that the final schedule performance is not yet known.
Productivity. In this study, Motorola defines productivity as the amount of work produced divided by the time to produce that work. Motorola uses the same lines of code metric as in the quality measurements for the productivity measurements, i.e. they measure the assembly-equivalent lines of code and the number of hours needed to produce that code.
In the paper, Motorola did not reveal the actual number of lines of code per hour, for "proprietary reasons". The data in figure 9 is therefore normalized to level 2 of the CMM.
The productivity is affected by other factors besides process maturity, most importantly by technology changes and reuse. These factors act as multipliers in the productivity of high-maturity-level projects.
Notable from the data shown is a drop in productivity when moving from level 2 to level 3 on the maturity scale. Motorola assumes this to be a side effect of asking the project staff to do many new things all at once on level 3. The institution of level 3 processes greatly affects the way the individual process members perform their tasks, and an "absorption cycle" will be needed before the full benefits of the higher-maturity processes can be observed.

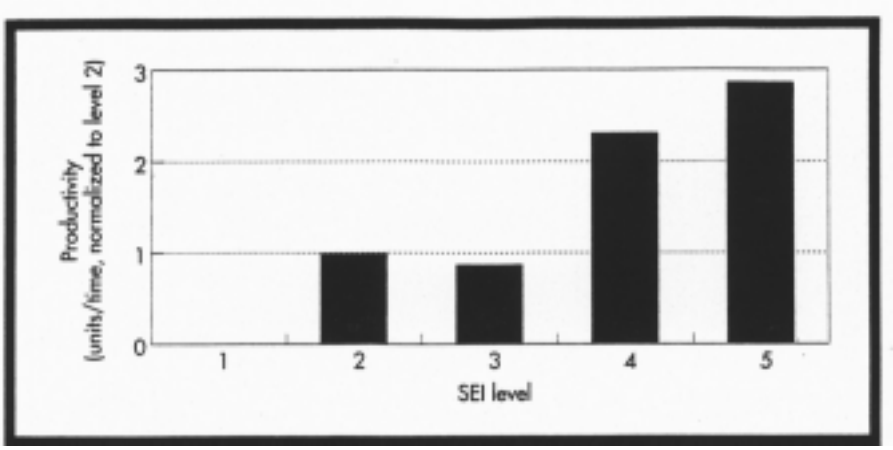


Figure 9: Relative productivity by CMM level at Motorola GED

Motorola calculated a theoretical return of investment of roughly $600K for an investment of about $100K, giving a return of 677 percent. However, given the government nature of the contracts under study, not all these savings could be realized as profits. Rather, Motorola emphasizes the non-monetary benefits of process improvement, especially the increase in the number of projects that finish within schedule, so that more engineering resources can be applied to the acquisition and development of new business.

3.3. Software Development Improvement at Lockheed (Safford, 1993)

In 1990, General Dynamic Fort Worth Division, now Lockheed Fort Worth Company, created their Software Process Review Team and tasked it with the development and documentation of a company Standard Software Development Process. This was triggered by a study of nine software industry software systems that showed an alarming number of failures (see figure 10).

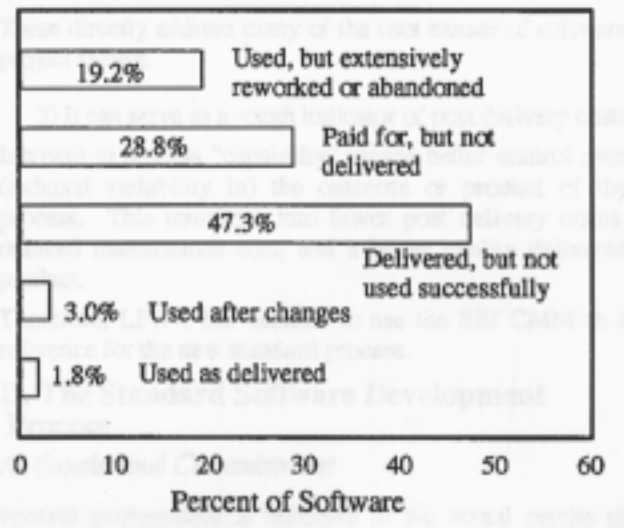


Figure 10: Study results of software systems

The study also found that over 50 percent of the contracts had cost overruns and over 60 percent of the contracts had schedule overruns.
The team used the Capability Maturity Model as a reference to implement their company's software improvement process to

- Improve the software process estimates
- Improve the ability to manage software project costs and schedules
- Achieve consistency in Quality of Product
- Reduce the cost of software maintenance

The team designed a process architecture consisting of a matrix of four major activities which covers six development life-cycle phases. The process was mandated for all new development. Management expected immediate improvements in the form of better estimates, better control, and a more capable work force.
Results. One of the primary concerns of the management was the cost of the process. In particular, the cost of training and administering of the process could cause budget overruns.
With cost and schedule data tracked against the planned budget these cost and schedule results were observed:

- Equipment cost is within plan
- Purchased software cost is within plan
- Man-hours are within plan
- Cost-to-complete expected within plan
- All deliverable schedules were met
- Internal schedules varied from plan

Another concern is the ability to benefit from the investment in training and process improvement in future projects. The emphasis on the reusability of the work products and the benefits of a more capable work force lead to these results:

- Development methodology and training: Engineers from the pilot project were able to "seed" other projects with knowledge of the software process. This also made better estimations and planning possible.
- Requirements: Creating a detailed Software Requirement Specification took twice as long as originally planned, but the time was recovered through better communication with the customer and better communication and coordination within the development team.
- Design: A framework of proven component designs with detailed specifications of interfaces and functionality allows to quickly determine the reusability of components.
- Code: Reused code must still be integrated and has to undergo a certain amount of testing, so the reuse of code, while important, is of limited use.
- Configuration management: The plans and procedures for configuration management were written for reuse by other projects from the onset. This can

lead to future savings of project overhead costs, and ensures that experience with the process gets propagated.

- Software Development plan: The Software development plan is used to record design decisions at the beginning of the project, and also changes that are made during the project. This is useful for planning purposes in future projects.

Unfortunately, the author of this study does not report quantitative data on the improvements that were obtained by the use of their software process.

3.4. Return on Investment throughout the industry (Brodman, Johnson, 1996)

The final study I analyzed reports on data on the return on investment from approximately 35 companies and government agencies.
This study found that most organizations that participated in the study did not have data on the return on investment from the implementation of software processes. However, the authors were able to collect raw data and analyze the data to produce a meaningful and more complete picture of the return on investment.
The authors noticed a lack of consensus of a definition of "Return on Investment" between government organizations and the industry. In general, government organizations focused on the dollar value of the return on investment, while the industry focuses on the personnel side. Since labor resources are a precious commodity in the software industry, process improvement initiatives are viewed in light of the effort in labor hours required to implement the initiatives. The return values expected by industry are by and large centered around meeting cost predictions, improving performance and staying within or under budget.
To find out about process improvement trends common to the organizations under study, the authors mapped the process maturity against the company backgrounds. Table 1 shows the observed correlations between the maturity levels and factors that influence the process maturity.

Degree of correlation	Factor correlating to process maturity
Insignificant correlation	Percent of company revenue from DoD
	Number of employees in company
	Duration of CMM-based process improvement program
Correlation to lower process maturity	Smaller software staff
	Shorter project duration
	Software contract average value below $1million
Correlation to higher process maturity	Directorate-level funding
	Number of software organizations within company
	Percent of software budget for process improvement

Table1: Factors influencing process maturity

The benefits of process improvements reported from the organizations were classified into nine categories of metrics. The data shown in table 2 indicates that the greatest growth in metric usage occurs as the organization matures from level 1 to level 2 of the CMM. This confirms the observation of Bollinger et. al (1991) that level 2 is the first level to require effort from the organization.

Metric category	Measurement	Level 1 percent usage	Level 2 percent usage	Level 3 percent usage	Level 4 percent usage
Process	Compliance	0	43	50	100
Stability	Resources	0	14	25	100
	Software requirements tracking	0	25	75	100
Quality	Problem reporting status	100	86	100	100
	Statistics on development defects	40	71	100	100
	Defect profile	20	71	75	100
	Product quality	0	14	50	100
	Percent of defects found in inspections	0	14	50	100
Cost	Cost data	25	100	100	100
Staffing	Personnel trends	100	100 (more consistent across lifecycle)	100 (more consistent across lifecycle)	100 (more consistent across lifecycle)
Productivity	Delivered lines of code	20	86	75	100
	New lines of code	40	86	100	100
	Percent of reused code	0	71	75	100
Effort	Estimate to complete	80	71	50	100
	Rework effort and time	0	29	50	-
Schedule	Variance	20	100	100	100
	Milestones	100	100	100	100
Progress	Module development	80	86	75	100
	Design	60	57	100	100
	Test	80	86	100	100
	Rework	-	14	-	-
	Earned value	0	57	75	100

Table 2: Measurement changes with growth in maturity

Further analysis of the data showed that there is an inconsistency with which these metrics are collected within the organizations participating in the study over the software lifecycle. The majority of the metrics is collected in four to five different patterns across the lifecycle.
Data on the benefits of CMM-based process improvement initiatives exists in the form of pockets of individual statistics. When evaluated across the industry, the data shows that process improvement is making a difference. The results are summarized in table 3.

Metric category	Measurement	Benefits realized by various software organizations*
Productivity	Increase in productivity	10-20%, 90-100%, 50%, 15-20%, 5%, 130%, 12%, 2.5-6.3%, 35%
Quality	Reduction in defects	10%, 80%, 50-70%, 50%
	Reduction in error rate	45%
	Product error rate	From 2.0 down to 0.11 per KLOC, from 0.72 down to 0.13 per thousand non-commented source statements
Cost	Project dollars saved to dollars invested	1.5:1, 2.0:1, 4:1, 6:1, 7.7:1, 10:1, 1.26:1, 5:1
	Project dollars saved	$2million to $3.4million
	Code problems during integration	20% of original value
	Decrease in cost of retesting	50%
	Cost savings of metrics program	50-300%, 40-290%
Schedule	Within estimate	5% of estimate
	On-time deliverables	From 51% up to 94% on time
	Project completion	From 50% down to 1% late
	Savings in schedule	10%, 20%
Effort	Reduction in rework	5 to 10%, from 40% down to 25% of effort, from 41% down to 11% of project cost
	Savings in test time	10 test hours per one analysis hour
* Results from different organizations are separated by commas		

4. Conclusions

It is now ten years after the Software Engineering Institute was established to develop a questionnaire to evaluate the software development capabilities of companies and governmental organizations.

I found it rather astonishing that only a few case studies have been published that indicate that CMM-based software process improvement pays off. In addition, most of these studies focus on DoD contracts, which can appear to the outside observer that these companies just "followed the money", because without implementing process improvement their business basis would have been diminished or completely vanished, since the DoD nowadays takes the process maturity into account when awarding new contracts.

Even the SEI itself acknowledges the lack of data, noting that the CMM does not explicitly require the results of each change to the process being measured (Herbsleb, Goldenson, 1996). This also hints at the possibility that a large number of companies that are doing business with the DoD invest into process improvement only because it is a requirement of the DoD, but they don't seem willing to go the extra mile and determine for themselves if process improvement helps their own bottom line.

All the studies I looked at stressed the importance of support from upper management to overcome resistance to change on the developer level and in middle management. Funding and support from the directorate level has a high correlation to the success of process improvement activities.

The available data shows that process improvement appears to pay off, although it may take a couple of years for the benefits to outweigh the investments. The biggest weakness in the data is that it is unclear whether the data is representative. It may be that only successful organizations publish these case studies, and unsuccessful experiences are not shared because of the fear of a negative impact on future business with the DoD.

I also noticed that commercial software development companies were mostly absent from the published studies. This again reinforces my suspicion that the upper management in most companies is still not convinced of the potential payoff that software process improvement has to offer, and dismiss it as a DoD fad that has no real place in the commercial world. Of course, the scarcity of the available data only serves as a case in point.

In my opinion, I think that the management in commercial software development companies is convinced that process improvement reaps benefits, but they are not willing to commit to the considerable upfront investment that all studies seem to imply.

5. Literature

Bollinger, T.B., McGowan, C. (1991): A Critical Look at Software Capability Evaluations, IEEE Software, July 1991

Brodman, J.G., Johnson, D.L (1994): What Small Businesses and Small Organizations Say About the CMM, Proceedings of ICSE-16, 1994

Brodman, J.G., Johnson, D.L. (1996): Return on Investment from Software Process Improvement as Measured by U.S. Industry, Crosstalk, April 1996

Caputo, K. (1996): Software CMM Maturity Levels: Can You See Beyond the Labels, Crosstalk, August 1996

Diaz, M., Sligo, J. (1997): How Software Process Improvement Helped Motorola, IEEE Software, Sept.-Oct. 1997

Dion, R. (1993): Process Improvement and the Corporate Balance Sheet, IEEE Software, July 1993

Hersleb, J.D., Goldenson, D.R. (1996): A Systematic Survey of CMM Experience and Results, Proceedings of ICSE-18, 1996

Humphrey, W.S., Curtis, B. (1991): Comments on 'A Critical Look', IEEE Software, July 1991

Humphrey, W.S., Kitson, D.H., Gale, J. (1991): A Comparison of U.S. and Japanese Software Process Maturity, Proceedings of ICSE-13, 1991

IEEE-STD-610, Institute of Electrical and Electronic Engineers

Kitson, D.H., Masters, S.M. (1993): An Analysis of SEI Software Process Assessment Results: 1987-1991, Proceedings of ICSE-15, 1993

Lawlis, P.K., Flowe, R.M., Thordahl, J.B. (1995): A Correlational Study of the CMM and Software Development Performance, Crosstalk, September 1995

Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V. (1993): Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA, 1993

Rubin, H.A. (1993): Software Process Maturity: Measuring its Impact on Productivity and Quality, IEEE International Software Metrics Symposium, 1993

Safford, E.L. (1993): Improving Software Development Through the Proper Implementation of a Standard Software Process: Case Study and Lessons Learned, IEEE 12th Digital Avionics Systems Conference, 1993