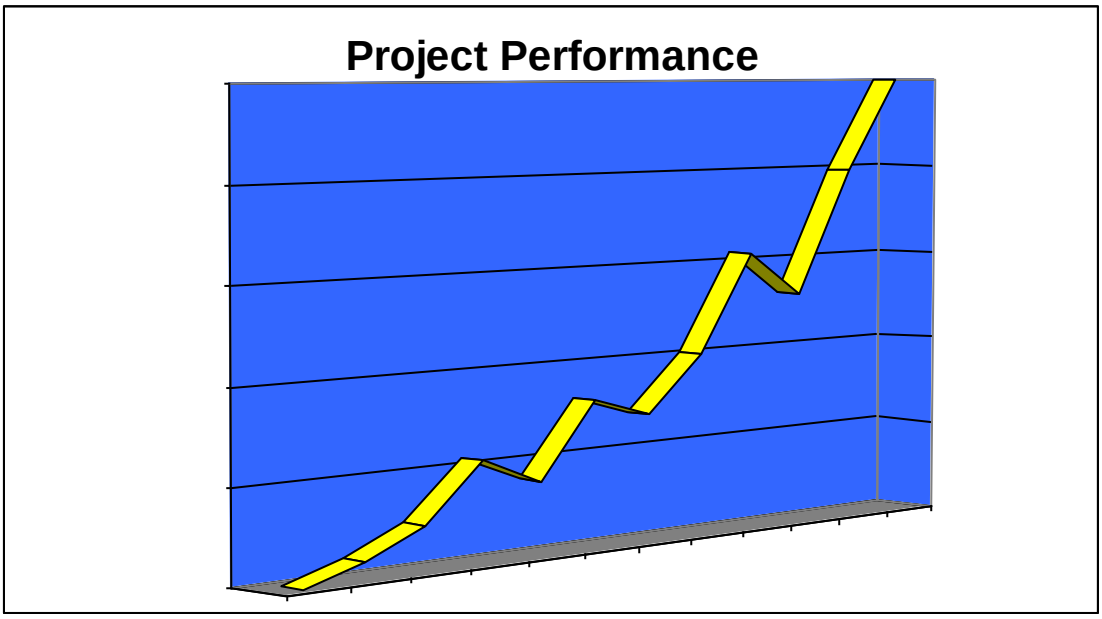
Chapter 14: Project and Process Learning and Maturity

Project Management Process Maturity Model: The Corporate Pursuit for Perfection

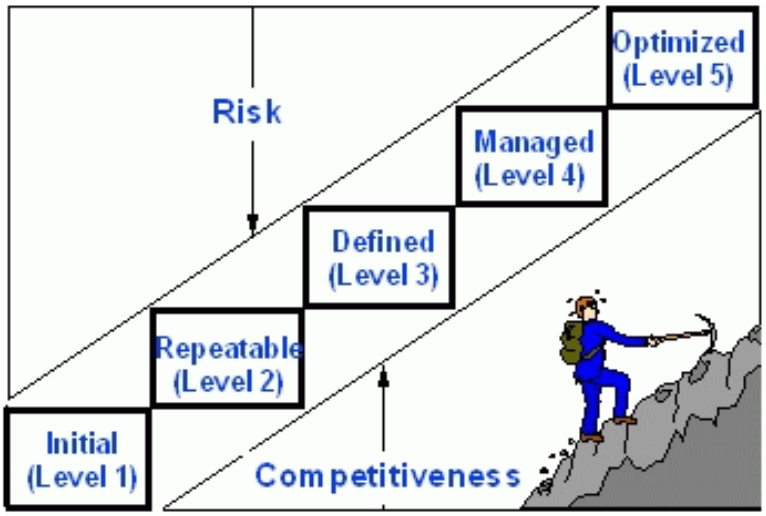


In today's world, organizations are searching for any advantage that will set them apart from their competition. Project management has recently been defined as the crux

of a successful business. Many organizations are taking project management to the next level by re-engineering their commitment to the management of projects as a discipline, not just as a collection of procedures in which there exists no uniformity between projects. It has become a mission-critical process and core competency for the "Big Five," as well as almost every successful business in America. The journey to become the best begins with an honest assessment of the organizations current state of project management and then focusing all attention on the concept of process improvement.

Lord Kelvin said over a century ago "When you can measure what you are speaking about, and express it in numbers, you [may] know something about it; but when you cannot express it, when you cannot measure it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, buy you have scarcely in your thoughts advanced to the state of Science." [cs]When comparing this philosophy to the practice of project management, Micro Frame Technologies, Inc. and Project Management Technologies, Inc. developed the Project Management Process Maturity Model to help determine the overall quality of an organization's project management techniques. This model provides a framework for helping organizations improve their project management functions and it also includes a set of phased maturity descriptions, improvement criteria, operating metrics, and questions that can be used to assess an organizations current level of project maturity. The objective of the Model is to improve the ability of organizations to *consistently* achieve cost, schedule, technical, and customer satisfaction goals [eazy].

There are five phases of the Process Maturity Model. Each phase defines characteristics concerning an organization's project management techniques. Certain steps need to be achieved in order for the firm to reach the next level. As an organization climbs the rungs of the model, project risk decreases while the organization's competitiveness among rival firms' increases. The maturity model begins with the *Initial* phase, then progresses to *Repeatable, Defined, Managed* and finally *Optimized*. Each phase is defined as follows:



The first phase of the Process Maturity Model is the *Initial* Phase. In this phase software process are ad hoc and chaotic. No planning is done and projects are just thrown together. Processes are not defined and project success depends on the individual effort of the team members, not that of the team. Determining the scope, duration, and cost of the project is nearly impossible due to the unstable environment in which the project is being performed and the fact that software processes are constantly being changed or modified as the project progresses [Burns]. Any sort of change in this phase is good. In order to reach the next level of the Process Maturity Model,

organizations must undergo rigorous project management techniques where the organization introduces and monitors basic management processes. Project estimation, scheduling, and tracking skills need to be addressed as well as change control and error tracking. Upon completion of these steps, an organization has reached a milestone in which most organizations have yet to achieve.

An organization has reached the *Repeatable* phase when basic project management policies are procedures are established. Cost, schedule and functionality are tracked by module and task. Every new project is started with a comparison of the firm's past projects. The successes or failures of those past projects are analyzed and a process discipline is put into place to repeat the success of those past projects. Any problems that are encountered are identified, documented, and archived. Project standards are defined and the project teams work with their customers and subcontracts to establish stable, managed working environments[sqi]. The disadvantages to being in the repeatable phase are numerous. First, there is a lack of well-defined testing. The lack of error data for earlier project activities and the lack of training are another handicap. The major drawback to the repeatable phase is that there is nobody tasked to identified, develop and make available documented processes with is the best practice.

The third phase of the Project Management Process Maturity Model is referred to as *Defined*. In order to obtain this phase, organizations must establish a Software Engineering Process Group (SEPG). This group is responsible for introducing company-wide standards regarding project management. They must perform inspections, ensuring those standards are being followed. More formal testing processes such as coverage analysis and traceability also need to be implemented. Once done, all software processes

are to be documented. Those processes then need to be standardized and integrated organization-wide. In order to gain employee support, training programs are implemented so that employees become familiar with the new procedures and do not reject them. During each project, a peer review process is used to enhance product quality. Process capability needs to be stable and based on a common understanding of processes, roles, and responsibilities in a defined process [cse]. Reaching that "common understanding" of an organization's project management processes is very difficult to acquire, but once obtained, the firm is defined as mature and ready to proceed to phase four.

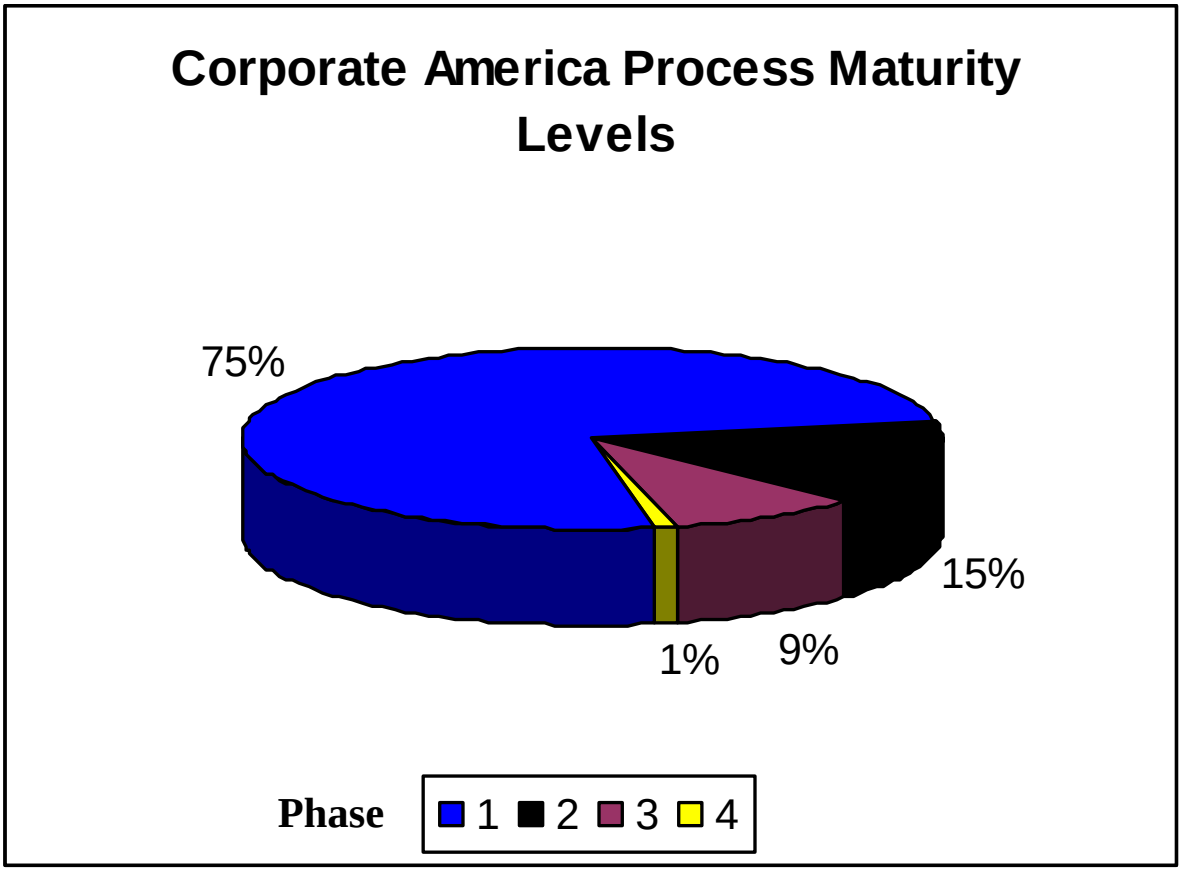
The *Managed* phase of the Project Management Process Maturity Model is reached through measuring and planning. The SEPG must establish aggressive, but attainable quality goals. These goals need to be expressed quantitatively and in the form of a plan. Product quality and productivity are collected and measured, then actual progress is compared to planed progress to help determine project success. Measurements are taken throughout the life of the project. Using these measurements, a productivity and quality database is defined. Product cycle time, number of defects, completeness, complexity, reliability, etc. are measured and archived in the database. This allows projects to achieve control by narrowing the variation in performance to within acceptable boundaries defined in the project plan. In the fourth phase reusability is a focal point. Project results are always being analyzed until an optimum method is reached, then that method is reused again and again to produce identical results [Burns].

The maturity model ends with the *Optimized* phase. This phase is knows as "The Phantom Phase [cs]." It is called this because it only exists on paper. No recorded

organization has yet accomplished this level of project maturity. In order to reach this level, an organization must focus on continuous improvement. This improvement is accomplished by testing innovative ideas and technologies then enabling quantitative feedback in order to enhance them. Improvement is very difficult in this phase. It can only occur from incremental advancements in existing processes [uregina]. Since those processes are so well honed from progressing through the other four phases, improvements in existing technologies is really the only way to catapult a company into this fifth phase.

Though the Process Maturity Model has benefited organizations worldwide, it does have some weaknesses. One frailty that the model possesses is that it does not address expertise in particular application domains. It also does not advocate specific software technologies, or suggest how to select, hire, motivate, and retain competent people [sqi]. Ultimately, a project's success does depend on the quality of those actually performing the project, but given the right tools, continuos project success can become a reality.

The Project Management Process Maturity Model is relatively new to the corporate world. Most corporations reside in the first two levels. The most successful organizations have entered the third phase, while only 1% has reached the fourth. While reaching the fifth level does not yet seem feasible, organizations are starting to think about project management as a discipline, and if they continue to do so, corporate *Optimization* is not that far away.


Corporate America Process Maturity Levels
75%
15%
9%
1%
Phase
1
2
3
4

BIBLIOGRAPHY

Burns, James. Texas Tech University: Lecture Notes, "Capable.ppt"

Http://www.cs.jmu.edu/users/foxcj/cs555/Unit10/Process/

Http://www.cse.unl.edu/~scotth/courses/461-fall98/week5-6/index.htm

Http://www.sqi.gu.edu.au/CMM/TR24/tr24.html

Http://www.eazy.net/~pmt/

Http://uregina.ca/~benedicl/Lectures/Ense474/Lecture3/ENSE474Lecture3/

THE CAPABILITY MATURITY MODEL

This model, which was developed by the Software Engineering Institute at Carnegie Mellon University, suggests five stages of improvement through which software development and maintenance organizations should go through.

IMMATURE SOFTWARE ORGANIZATIONS
- Processes are ad hoc, and occasionally chaotic
- Processes improvised by practitioners
- Testing and reviews usually curtailed under stress.
- Quality is unpredictable
- Costs and schedules are usually exceeded
- Reactionary management is usually firefighting
- Success rides on individual talent and heroic effort
- Technology benefits are lost in the noise

MATURE SOFTWARE ORGANIZATIONS
- Processes are defined and documented.
- Management plans, monitors, and communicates
- Roles and responsibilities are clear
- Product and process are measured
- Quality, costs, and schedules are predictable
- Management committed to continuous improvement
- Technology used effectively within defined process
- Process--the set of activities, methods, practices, and transformations that integrate managers and software engineers in using technology to develop and maintain software

Software Process Definition
Project Planning

Project Management
Software Engineering Procedures
Software standards
Software Quality Evaluation
Software Configuration management
The Five Levels of Software Process Maturity
    INITIAL
    REPEATABLE
    DEFINED
    MANAGED
    OPTIMIZING

INITIAL

Initially, the software process is ad hoc, even chaotic. The software processes are not defined. Success depends on individual effort. The environment is not stable. The benefits of software engineering practices are undermined. Planning is nonexistent or ineffective. Process capability is unpredictable because the software process is constantly changed or modified as the work progresses

REPEATABLE

Basic project management policies and procedures are established. Cost, schedule and functionality are tracked by module and task. A process discipline is put in place to repeat earlier successes. Managing new projects is based on experience with similar projects. Basic software management controls are installed. Estimations of cost and time to complete are based on history for similar projects. Problems are identified and documented. Software requirements are baselined. Project standards are defined. Project teams work with their customers and subcontractors to establish stable, managed working environments. Process is under the control of a project management system that is driven by performance on previous projects. A project performance database is defined and populated.

DEFINED

The software process is documented. The software process is standardized and integrated organization-wide. All projects use a documented and approved version of the organization\rquote s process of developing and maintaining software. A software engineering process group facilitates process definition and improvement efforts. Organization-wide training programs are implemented\par. The organization-wide standard software process can be refined to encompass the unique characteristics of the project. A peer review process is used to enhance product quality. Process capability is stable and based on a common understanding of processes, roles, and responsibilities in a defined process.

MANAGED

Quantitative quality goals are defined. Product quality and productivity are measured and collected. Both processes and products are quantitatively understood. Both processes and products are controlled using detailed measures. A productivity and

quality database is defined. Projects achieve control by narrowing the variation in performance to within acceptable boundaries. Process variation is controlled by use of a strategic business plan that details which product lines to pursue. Risks associated with moving up the learning curve of a new application domain are known and carefully managed. Process capability is measured and operating within measurable limits.

OPTIMIZING
Continuous process improvement is enabled by quantitative feedback. Continuous process improvement is assessed from testing innovative ideas and technologies. Weak process elements are identified and strengthened
Defect prevention is explicit. Statistical evidence is available on process effectiveness
Innovations that exploit the best software engineering practices are identified

Improvement occurs from

INCREMENTAL ADVANCEMENTS IN EXISTING PROCESSES
INNOVATIONS USING NEW TECHNOLOGIES AND METHODS

Software Maturity Framework

A software product is produced using some combination of tools and methods. This combination of tools and methods is called the software process. To address the problems encountered in developing software, we must treat the entire software task as a process that can be controlled, measured and improved.

The sequence of broad steps that an organization needs to follow in order to improve the software process are as follows;
1) See: Study the current development process to understand its status.
2) Forsee: Develop a vision of the process status that needs to be achieved.
3) Prioritize: Create a list of required actions for process improvement in order of priority.
4) Plan: Make a feasible plan to accomplish those actions.
5) Resource: Allocate the various resources to implement and carry through the plan.
6) Redo: Repeat the five steps above.

The Process Capability Maturity Model (CMM) was proposed by the Software Engineering Institute (SEI) at Carnegie-Mellon University. According to the CMM, the state of the software process can be categorized into one of five maturity levels. These levels are as follows;
1. Initial: The process is brought under statistical control.
2. Repeatable: The process has repeatable statistical control made possible by initiating rigorous project management of commitments, costs, schedules, and changes.
3. Defined: The process has consistent implementation and provides a basis for a better understanding of itself.

4. Managed: Comprehensive process measurements and analyses beyond those of cost and schedule performance are being performed.
5. Optimizing: A foundation for continuing improvement and optimization of the process has been established.

The optimizing process helps identify the areas of need and gives direction on how to best fulfil them, provides concise and quantitative information, and allows professionals to study work performance and to see how to improve it.

Principles of Software Process Change

A software team contains usually a mix of talents ranging from unusually talented to marginal. Such a team must be managed by a leader who leads with the conviction that long term improvements are possible and essential. The six basic principles of software process change are:
1. All major changes to the software process must originate at the top level.
2. Personnel at all levels must be involved.
3. A knowledge of the current process is required for effective change.
4. Change is continuous.
5. A conscious effort and periodic reinforcement are needed for software process changes to be retained.
6. Investment is required for software process improvement.

Once the decision to initiate process improvement has been made, these are the key topics to focus on. To improve the software process, someone needs to be the champion. Unplanned process improvement is wasteful. It is pointless to automate of a poorly defined process. Improvements should be made with caution with testing at every step. All personnel should be adequately trained.

Changes must be handled with care or they will generate resistance. The sequence of change involves an unfreezing step, the establishment of resources to serve as change agents, planning, implementing and communicating the change, and refreezing using education and training.

Software Process Assessment

Software process assessments help to identify the critical problems and the priorities. This process is conducted by a team of professionals who are well experienced. The process begins by identifying the areas needed to be improved priority wise. Senior management needs to be committed to the process by agreeing to participate personally.
Confidentiality is a must so that the assessors can talk to people to uncover the real problem issues. The assessment will be of a waste of

time unless the site manager (who is the senior manager of the total organization) personally participates by assigning qualified people and by periodical reviews of the progress made by these plans. The assessment team members should all be experienced software developers. Four to six professionals typically form an adequate team, The assessment team must have a set of ground rules where the site manager as mentioned above and the assessment team leader should sign a written agreement as proposed by the SEI. The team undergoes a training after which the work begins. To get accurate information copies of work products can be obtained from the respondents. Finally a written report is given to the site manager and staff. Proper follow through is necessary to implement the efforts towards improvement.

The Initial Process

The inability to deliver on schedule is often the rule and not an exception in the case of software organizations. Management frustration increases as new plans are successively missed.
Often programs need a lot more code than expected and as the programs get larger, newer technical and management issues crop up and automatically the costs increase.
The software scale affects the individual, the management system and the technical methods and tools that are used. Most managers and professionals start out writing small programs, that the larger scale system comes as a hard surprise. So though they may not have trouble writing and completing their own modules, they get increasingly frustrated at having to coordinate with so many others too.
Unless proper planning at all levels are done then the problems of scale will not be easily understood or anticipated when confronted with it. In very large software systems the most severe problems are not really obvious until it gets to the testing phase.
The solution out of this is to estimate, plan and manage the project. Both levels of managers and software professionals must act responsibly by first planning and then only committing themselves to a date when faced with a problem.

Managing Software Organizations

The Management system's role is to ensure successful completion of the projects. Now the foundation for proper software project management is the discipline of commitment. A commitment involves a planned completion date and a payment amount. The large software projects involve the cooperative efforts of many individuals. Commitment begins from the top level and their personal involvement will motivate the others in the hierarchy. Product plans focus on the activities and objectives of each project dealing with issues like function, cost,

schedule, and quality. Organizations have line and staff groups with conflicting goals because line management focuses on getting the product out of the door, while the staff is building the organization's competence. Quarterly reviews should be conducted by the senior manager as it provides a forum for resolving conflicts and monitoring progress. The topics should typically include an assessment of project performance against plan and the organization's performance against its goals. Organizational improvement is a matter of priorities as by being included in the quarterly reviews the item will get the priority attention required to produce results. Each project area establishes its own plans, which are reviewed prior to project initiation and then periodically updated and re-reviewed.
These disparate planning systems are coupled through senior management quarterly oversight reviews that provide the forum for resolving conflicts and balancing resources between the line and staff organizations.
In addition to the quarterly review process, management needs to periodically assess project progress. This is accomplished through a sequence of phase reviews held at key points in the project schedule.
In establishing a project management system, the first essential action is to obtain agreement from the senior management team that such a system is needed.

The Project Plan

The project plan provides a definition of each major task, an estimate of the time and resources required, and a framework for management review and control. It is developed at the beginning of a job and is successively refined as the work progresses.
With rare exceptions, initial resource estimates and schedules are unacceptable. This is not because the programmers are unresponsive, but because the users generally want more than they can afford. If the job doesn't fit the available schedule and resources, it must be either pared down or the time and resources increased.
The elements of a software plan are, goals and objectives, a sound conceptual design, Work Breakdown Structure (WBS), product estimates, resource estimates, and the project schedule. In addition to defining the work, this plan provides management the basis for periodically reviewing and tracking progress against the plan.

The measure used in program size estimation should be reasonably easy to use early in the project and readily measurable once the work is completed. Subsequent comparison of the early estimates to the actual measured product size then provides feedback to the estimators on how to make more accurate estimates. Once an estimate of the amount of code to be developed is obtained, this can be converted into the number of

programmer months and time required. From the total resource need, the project schedule can be developed by spreading these resources over the planned software engineering phases.
After the estimates and schedule are completed, the full development plan is assembled in a complete package and circulated to all involved groups for review and sign-off. At each phase review, the development plan is updated. The schedule shows task status and current projections compared to the plan. The most important single pre-requisite to good software cost estimating is the establishment of an estimation group.

Version and Change Control

One of the fundamental activities of software engineering is change management. Changes to the requirements can occur as a response when testing is done and sometimes the original requirements may be changed. These efforts require proper management due to the number of people involved and the volume of change and this is called software configuration management. The baseline is the official source for code and the repository for all completed work. When tests are run and problems are found and changes need to be made it is important to keep track of revisions. The change log could include all the information. The problem report is very important as it records every problem and the precise conditions causing it. To implement these controls and procedures, responsibility assignments are made for the configuration manager, module ownership, and the Change Control Board(CCB). The system library stores the development work products. This includes the source and object code for every baseline and change, the test cases, and the development tools. It has locks to prevent unauthorized changes and the capability to build the various system configurations, test drivers, and test scenarios or buckets required by development.

Software Quality Assurance

The role of Software Quality Assurance (SQA) is to assure management that the software development work is performed the way it is supposed to be. In small organizations it is possible for software managers to monitor the work so closely that SQA work is not needed. Its prime benefit to management is the assurance it provides them that their directions are actually implemented. To be effective, SQA needs to work closely with management but independently staffed with competent professionals. The SQA organization is not responsible for producing quality products or for making quality plans, these are development jobs. SQA is responsible for auditing the quality actions of the line organization and alerting management to any deviations. The SQA ought to report to a person in a high level of management and not to a software development manager in order to be of use.

If SQA fulfills its role, and if senior management refuses to allow line management to commit or to ship products until the SQA issues have been addressed, then SQA can help management improve product quality. Each development and maintenance project should have a software quality assurance plan that specifies its goals, the SQA tasks to be performed, the standards against which the development work is to be measured, and the procedures and organizational structure to be used.
The reasons why SQA teams fail is because of poor staffing, low negotiating skills, often operates without approved development standards, and not all software groups have real quality plans.

Software Standards

A standard is a rule or basis for comparison that is used to assess the size, content, value, of quality of something.
Standards help the SQA people in doing their work. It is wise to make an overall plan which combines the available standards, the priority needs, the status of the projects, the available staff skills, and the means of standard enforcement. Then one can go ahead and make a standards development program. This work can be done by individuals or small working groups of technical experts.
Standards must be kept current, but standards maintenance should not involve a great deal of work. Frequent changes to a standard probably means it covers a subject that is not ready for standardization. The standards and procedures should also be reviewed at least once every three or five years to ensure they are current and needed. Standards enforcement is the basic role of the SQA organization.

Software Inspections

The fundamental purpose of inspections is to improve the quality of programs by assisting the software engineers to recognize and correct their errors. Inspections are enormously effective, and all software organizations should use inspections, walkthroughs, or technical reviews in all major aspects of their work. Inspections involve requirements, design, implementation, testing, maintenance and documentation.
Inspections help detect errors early in the development phase and help to ensure that the appropriate parties work towards the right direction. Positive results because of inspections have increased their popularity in software organizations. Inspections are an important way to find errors and they score over testing as they detect the mistakes much earlier thus helping economically and saving time and effort too. Inspections should be a required part of every well run software process

Software Testing

Software testing is the execution of a program to find its faults.
A test is an experiment and should be approached as such. White box tests examine the basic design of the program and require that the tester have detailed knowledge of the program's internal structure. Black box tests examine the program to see if it meets the functional specifications. Effective test planning starts with an overall development plan that defines the functions, roles, and methods for all test phases. Every test should be treated with careful control and at the conclusion of a test a report with all the detail should be produced. The different types of bugs can be classified. The test results should be carefully analyzed to make decisions. Since programmers are inherently incapable of bug testing their own programs, special test groups can assume test responsibility. Unit test standards can help the programmers to do a reasonable job of testing following which the job can be transferred to a dedicated test group. As this new group gains experience in finding bugs they will soon become extra ordinarily effective at doing just that.

Software Configuration Management

Software Configuration Management (SCM) control over requirements and specifications is needed to ensure that the product being built and tested is what is wanted. SCM control must be maintained over the design throughout system life to ensure integrity and maintainability. An SCM plan can be developed and the specification is used as a basis for the development work and as a reference for developing the functional, system, and acceptance tests.
When changes are made, correspondingly data is updated to the code change. Finally, for large projects and for all projects during the maintenance phase, procedures are needed to handle the development of simultaneous versions of the same program. The tools used to design, implement, test, and maintain the software must also be maintained under configuration control.
The purpose of software configuration status accounting is to maintain a continuous record of the status of all baselined items. A software configuration audit is periodically performed to ensure that the SCM practices and procedures are rigorously followed

Defining the Software Process

Defining of the software process helps to give a framework to software organizations. The conflicting needs for customization and standardization can be met by establishing a process architecture with standard unit or "kernel" process steps and rules for describing and relating them. Customization is then achieved by their interconnection

into process models.
Models made of the software process are of 3 types- Universal(U) process level, the Worldly(W) process level, and the Atomic (A) process level. These are typically embodied in policies at the U level, procedures at the W level, and standards at the A level.
Every software organization should establish a process architecture and models suited to its own particular needs. The relationship between management, control and support activities have a critical impact on the behavior of the organization. Some guidelines on developing and using a process architecture are, establish objectives, define the basic process architecture, make sure it meets the needs of the projects, and then enforce it as an overall process framework. Also remember that each project, component and module is unique and its process should be uniquely determined. In developing process architecture one can first create a high level architecture and then slowly refine it a few steps at a time. Areas where professionals need guidance should be given priority.

The Software Engineering Process Group

The software process must have a framework which is well established and this process can be changed depending on the nature and scale of problems encountered
The Software Engineering Process Group(SEPG) can provide guidance as to the areas needing change though the decision to make the change will ultimately rest with the line management. The SEPG can be viewed as a change agent and it is their responsibility to provide resources, to track the progress and keeping the management informed
The SEPG must be staffed with competent software professionals and each organization could staff the group depending on finance and availability of men. This group could provide the technology support as it maintains the process database which contains data on the entire software engineering process. This group should not have to report to line development management and instead could report to the same executive reporting center as the SQA. This group in order to maintain focus on its tasks should limit its focus on tasks that can be handled effectively and quickly.

Data Gathering and Analysis

Software process data must be gathered with a clear objective or the right information may take long in being recorded. The principles of successful data gathering are, that the data must have a process model made in order to be able to get to the specific information. It is an expensive task and hence the goodwill of the management is essential.
The data gathering plan includes the user of the data, the need for the

data, method used to gather it and how it will be managed. Measurements can also be objective or subjective, absolute or relative, explicit or derived, dynamic or static, and predictive or explanatory. The main objective in software engineering is the use of dynamic, absolute, explicit measures to control the work that needs to get done. Considerations in data analyzing are to understand the shortcomings of the process and to effect change by providing the necessary resources. Data gathering is of infinite help in analyzing the software process.

Managing Software Quality

Evaluation of a software development project is to examine the quality of the product. This examination can help one understand the progress made and can establish a framework for improvement. A superior performance can be achieved by the management setting challenging goals, making a plan for accomplishing those goals and keeping track of it by reviews.
Quality of a product can be measured by, development, product, acceptance, usage and repair. During initial project planning, a quality plan is produced. This plan is documented, reviewed, and compared with actual experience. It is possible to keep track of quality performance by estimating the defect removal at each process stage and the effects of quality in each stage can also be found. The purpose of the whole exercise must be kept in mind, namely to motivate action and not to evaluate people. A company that has a quality action plan will always improve in its output whereas a company that always meets its quality plans has little room for improvement.

Defect Prevention

Software development and maintenance can attribute most of its costs to error finding and fixing. Once an error is found by inspections and testing, then rework needs to be done. Defect prevention is mainly instituted to give a focus for improving the process. Its fundamental objective is to make sure errors are not repeated and it is a skill that will take time to learn. Defects can be prevented when programmers evaluate their own errors and this can be done when management makes a strong commitment to quality. The process of defect prevention has a few steps, reporting of the defect, an action plan for removing the defect and then effective tracking of performance. Once this plan is put into action results will show up by the end of six months and a transformation of the organization can be observed.

Automating the Software Process

Automating the software process can improve the quality of work and the

strategy can be effectively put into action by being clear about the changes needed to be made, feasibility and an orderly plan to work towards it. Automation of the software environment must be convenient to use and should have a conceptual schema that will encompass the database, process data, and tool interfaces.
The basic steps required to establish a long- term automation plan are, establish an Automation Focal Point, understand current automation status, make an orderly assessment of the most promising available environments and tools, start work on a common data model for the software environment, and establish a common user interface.
After this the technology currently in use and the one that is being planned must be mapped out. A team of experts that will determine the impacts of the new technology and the support tools for it. Then a review of each project's savings schedule should be made with enough room for flexibility. Once this is completed then the savings schedule for the organization as a whole can be submitted to the management to get the approval needed. Above all, involve the financial people, and do the work in detail. Without line management's commitment to these savings, no approval is likely. It takes time to do it right but there is no shorter way.

Contracting for Software

The software process is essentially an agreement or contract between management and the development organization. One of the benefits of establishing a software process is to be able to assess the status of a project and thus progress can be known at various stages. Auditing of the project helps to see evidence of project performance. There may be different types of vendors and buyers in contracts but when both parties are technically competent then the level of cooperation is greatly increased .The basic system objectives are established at the outset, where specific functional requirements are known , they are stated. A documented plan is then produced foe establishing and validating the requirements and the operational concept against these objectives, and a joint requirements effort defines the system tasks with sufficient precision to permit design to start.
Quantitative process tracking is when check points are given for a job full completed. Quality can be monitored by making a plan where defect injection , removal, inspection coverage, and inspection coverage are included. Establishing SCM, SQA, SEPG(these fall in levels 1 and 2) groups and other steps help to make the software process more mature. But with all this it becomes more important for managers and contracting officers to give technical leadership by setting goals and striving to meet them .

Conclusion

All these approaches to making a mature software process are straightforward and have been proven to be effective. The problems addressed with regard to the software process have existed for a while. Undoubtedly as the software process evolves newer approaches will be found as people while confronting problems seek solutions. It is best to get started on the road to improvement by getting competent people to apply themselves to the making of a more mature software process. The need is now as the software industry has touched all the important areas of our daily life and it is important that fewer mistakes are made with proper management.

Software Maturity Framework

A software product is produced using some combination of tools and methods. This combination of tools and methods is called the software process. To address the problems encountered in developing software, we must treat the entire software task as a process that can be controlled, measured and improved.

The sequence of broad steps that an organization needs to follow in order to improve the software process are as follows;
1) See: Study the current development process to understand its status.
2) Forsee: Develop a vision of the process status that needs to be achieved.
3) Prioritize: Create a list of required actions for process improvement in order of priority.
4) Plan: Make a feasible plan to accomplish those actions.
5) Resource: Allocate the various resources to implement and carry through the plan.
6) Redo: Repeat the five steps above.

The Process Capability Maturity Model (CMM) was proposed by the Software Engineering Institute (SEI) at Carnegie-Mellon University. According to the CMM, the state of the software process can be categorized into one of five maturity levels. These levels are as follows;
1. Initial: The process is brought under statistical control.
2. Repeatable: The process has repeatable statistical control made possible by initiating rigorous project management of commitments, costs, schedules, and changes.
3. Defined: The process has consistent implementation and provides a basis for a better understanding of itself.
4. Managed: Comprehensive process measurements and analyses beyond those of cost and schedule performance are being performed.
5. Optimizing: A foundation for continuing improvement and optimization

of the process has been established.

The optimizing process helps identify the areas of need and gives direction on how to best fulfil them, provides concise and quantitative information, and allows professionals to study work performance and to see how to improve it.

Principles of Software Process Change

A software team contains usually a mix of talents ranging from unusually talented to marginal. Such a team must be managed by a leader who leads with the conviction that long term improvements are possible and essential. The six basic principles of software process change are:
1. All major changes to the software process must originate at the top level.
2. Personnel at all levels must be involved.
3. A knowledge of the current process is required for effective change.
4. Change is continuous.
5. A conscious effort and periodic reinforcement are needed for software process changes to be retained.
6. Investment is required for software process improvement.

Once the decision to initiate process improvement has been made, these are the key topics to focus on. To improve the software process, someone needs to be the champion. Unplanned process improvement is wasteful. It is pointless to automate of a poorly defined process. Improvements should be made with caution with testing at every step. All personnel should be adequately trained.

Changes must be handled with care or they will generate resistance. The sequence of change involves an unfreezing step, the establishment of resources to serve as change agents, planning, implementing and communicating the change, and refreezing using education and training.

Software Process Assessment

Software process assessments help to identify the critical problems and the priorities. This process is conducted by a team of professionals who are well experienced. The process begins by identifying the areas needed to be improved priority wise. Senior management needs to be committed to the process by agreeing to participate personally.
Confidentiality is a must so that the assessors can talk to people to uncover the real problem issues. The assessment will be of a waste of time unless the site manager (who is the senior manager of the total organization) personally participates by assigning qualified people and by periodical reviews of the progress made by these plans. The

assessment team members should all be experienced software developers. Four to six professionals typically form an adequate team, The assessment team must have a set of ground rules where the site manager as mentioned above and the assessment team leader should sign a written agreement as proposed by the SEI. The team undergoes a training after which the work begins. To get accurate information copies of work products can be obtained from the respondents. Finally a written report is given to the site manager and staff. Proper follow through is necessary to implement the efforts towards improvement.

The Initial Process

The inability to deliver on schedule is often the rule and not an exception in the case of software organizations. Management frustration increases as new plans are successively missed.
Often programs need a lot more code than expected and as the programs get larger, newer technical and management issues crop up and automatically the costs increase.
The software scale affects the individual, the management system and the technical methods and tools that are used. Most managers and professionals start out writing small programs, that the larger scale system comes as a hard surprise. So though they may not have trouble writing and completing their own modules, they get increasingly frustrated at having to coordinate with so many others too.
Unless proper planning at all levels are done then the problems of scale will not be easily understood or anticipated when confronted with it. In very large software systems the most severe problems are not really obvious until it gets to the testing phase.
The solution out of this is to estimate, plan and manage the project. Both levels of managers and software professionals must act responsibly by first planning and then only committing themselves to a date when faced with a problem.

Managing Software Organizations

The Management system's role is to ensure successful completion of the projects. Now the foundation for proper software project management is the discipline of commitment. A commitment involves a planned completion date and a payment amount. The large software projects involve the cooperative efforts of many individuals. Commitment begins from the top level and their personal involvement will motivate the others in the hierarchy. Product plans focus on the activities and objectives of each project dealing with issues like function, cost, schedule, and quality. Organizations have line and staff groups with conflicting goals because line management focuses on getting the product out of the door, while the staff is building the organization's

competence. Quarterly reviews should be conducted by the senior manager as it provides a forum for resolving conflicts and monitoring progress. The topics should typically include an assessment of project performance against plan and the organization's performance against its goals. Organizational improvement is a matter of priorities as by being included in the quarterly reviews the item will get the priority attention required to produce results. Each project area establishes its own plans, which are reviewed prior to project initiation and then periodically updated and re-reviewed.
These disparate planning systems are coupled through senior management quarterly oversight reviews that provide the forum for resolving conflicts and balancing resources between the line and staff organizations.
In addition to the quarterly review process, management needs to periodically assess project progress. This is accomplished through a sequence of phase reviews held at key points in the project schedule.
In establishing a project management system, the first essential action is to obtain agreement from the senior management team that such a system is needed.

The Project Plan

The project plan provides a definition of each major task, an estimate of the time and resources required, and a framework for management review and control. It is developed at the beginning of a job and is successively refined as the work progresses.
With rare exceptions, initial resource estimates and schedules are unacceptable. This is not because the programmers are unresponsive, but because the users generally want more than they can afford. If the job doesn't fit the available schedule and resources, it must be either pared down or the time and resources increased.
The elements of a software plan are, goals and objectives, a sound conceptual design, Work Breakdown Structure (WBS), product estimates, resource estimates, and the project schedule. In addition to defining the work, this plan provides management the basis for periodically reviewing and tracking progress against the plan.

The measure used in program size estimation should be reasonably easy to use early in the project and readily measurable once the work is completed. Subsequent comparison of the early estimates to the actual measured product size then provides feedback to the estimators on how to make more accurate estimates. Once an estimate of the amount of code to be developed is obtained, this can be converted into the number of programmer months and time required. From the total resource need, the project schedule can be developed by spreading these resources over the planned software engineering phases.

After the estimates and schedule are completed, the full development plan is assembled in a complete package and circulated to all involved groups for review and sign-off. At each phase review, the development plan is updated. The schedule shows task status and current projections compared to the plan. The most important single pre-requisite to good software cost estimating is the establishment of an estimation group.

Version and Change Control

One of the fundamental activities of software engineering is change management. Changes to the requirements can occur as a response when testing is done and sometimes the original requirements may be changed. These efforts require proper management due to the number of people involved and the volume of change and this is called software configuration management. The baseline is the official source for code and the repository for all completed work. When tests are run and problems are found and changes need to be made it is important to keep track of revisions. The change log could include all the information. The problem report is very important as it records every problem and the precise conditions causing it. To implement these controls and procedures, responsibility assignments are made for the configuration manager, module ownership, and the Change Control Board(CCB). The system library stores the development work products. This includes the source and object code for every baseline and change, the test cases, and the development tools. It has locks to prevent unauthorized changes and the capability to build the various system configurations, test drivers, and test scenarios or buckets required by development.

Software Quality Assurance

The role of Software Quality Assurance (SQA) is to assure management that the software development work is performed the way it is supposed to be. In small organizations it is possible for software managers to monitor the work so closely that SQA work is not needed. Its prime benefit to management is the assurance it provides them that their directions are actually implemented. To be effective, SQA needs to work closely with management but independently staffed with competent professionals. The SQA organization is not responsible for producing quality products or for making quality plans, these are development jobs. SQA is responsible for auditing the quality actions of the line organization and alerting management to any deviations. The SQA ought to report to a person in a high level of management and not to a software development manager in order to be of use.
If SQA fulfills its role, and if senior management refuses to allow line management to commit or to ship products until the SQA issues have been addressed, then SQA can help management improve product quality.

Each development and maintenance project should have a software quality assurance plan that specifies its goals, the SQA tasks to be performed, the standards against which the development work is to be measured, and the procedures and organizational structure to be used.
The reasons why SQA teams fail is because of poor staffing, low negotiating skills, often operates without approved development standards, and not all software groups have real quality plans.

Software Standards

A standard is a rule or basis for comparison that is used to assess the size, content, value, of quality of something.
Standards help the SQA people in doing their work. It is wise to make an overall plan which combines the available standards, the priority needs, the status of the projects, the available staff skills, and the means of standard enforcement. Then one can go ahead and make a standards development program. This work can be done by individuals or small working groups of technical experts.
Standards must be kept current, but standards maintenance should not involve a great deal of work. Frequent changes to a standard probably means it covers a subject that is not ready for standardization. The standards and procedures should also be reviewed at least once every three or five years to ensure they are current and needed. Standards enforcement is the basic role of the SQA organization.

Software Inspections

The fundamental purpose of inspections is to improve the quality of programs by assisting the software engineers to recognize and correct their errors. Inspections are enormously effective, and all software organizations should use inspections, walkthroughs, or technical reviews in all major aspects of their work. Inspections involve requirements, design, implementation, testing, maintenance and documentation.
Inspections help detect errors early in the development phase and help to ensure that the appropriate parties work towards the right direction. Positive results because of inspections have increased their popularity in software organizations. Inspections are an important way to find errors and they score over testing as they detect the mistakes much earlier thus helping economically and saving time and effort too. Inspections should be a required part of every well run software process

Software Testing

Software testing is the execution of a program to find its faults.
A test is an experiment and should be approached as such. White box

tests examine the basic design of the program and require that the tester have detailed knowledge of the program's internal structure. Black box tests examine the program to see if it meets the functional specifications. Effective test planning starts with an overall development plan that defines the functions, roles, and methods for all test phases. Every test should be treated with careful control and at the conclusion of a test a report with all the detail should be produced. The different types of bugs can be classified. The test results should be carefully analyzed to make decisions. Since programmers are inherently incapable of bug testing their own programs, special test groups can assume test responsibility. Unit test standards can help the programmers to do a reasonable job of testing following which the job can be transferred to a dedicated test group. As this new group gains experience in finding bugs they will soon become extra ordinarily effective at doing just that.

Software Configuration Management

Software Configuration Management (SCM) control over requirements and specifications is needed to ensure that the product being built and tested is what is wanted. SCM control must be maintained over the design throughout system life to ensure integrity and maintainability. An SCM plan can be developed and the specification is used as a basis for the development work and as a reference for developing the functional, system, and acceptance tests.

When changes are made, correspondingly data is updated to the code change. Finally, for large projects and for all projects during the maintenance phase, procedures are needed to handle the development of simultaneous versions of the same program. The tools used to design, implement, test, and maintain the software must also be maintained under configuration control.

The purpose of software configuration status accounting is to maintain a continuous record of the status of all baselined items. A software configuration audit is periodically performed to ensure that the SCM practices and procedures are rigorously followed

Defining the Software Process

Defining of the software process helps to give a framework to software organizations. The conflicting needs for customization and standardization can be met by establishing a process architecture with standard unit or "kernel" process steps and rules for describing and relating them. Customization is then achieved by their interconnection into process models.

Models made of the software process are of 3 types- Universal(U) process level, the Worldly(W) process level, and the Atomic (A) process level.

These are typically embodied in policies at the U level, procedures at the W level, and standards at the A level.
Every software organization should establish a process architecture and models suited to its own particular needs. The relationship between management, control and support activities have a critical impact on the behavior of the organization. Some guidelines on developing and using a process architecture are, establish objectives, define the basic process architecture, make sure it meets the needs of the projects, and then enforce it as an overall process framework. Also remember that each project, component and module is unique and its process should be uniquely determined. In developing process architecture one can first create a high level architecture and then slowly refine it a few steps at a time. Areas where professionals need guidance should be given priority.

The Software Engineering Process Group

The software process must have a framework which is well established and this process can be changed depending on the nature and scale of problems encountered
The Software Engineering Process Group(SEPG) can provide guidance as to the areas needing change though the decision to make the change will ultimately rest with the line management. The SEPG can be viewed as a change agent and it is their responsibility to provide resources, to track the progress and keeping the management informed
The SEPG must be staffed with competent software professionals and each organization could staff the group depending on finance and availability of men. This group could provide the technology support as it maintains the process database which contains data on the entire software engineering process. This group should not have to report to line development management and instead could report to the same executive reporting center as the SQA. This group in order to maintain focus on its tasks should limit its focus on tasks that can be handled effectively and quickly.

Data Gathering and Analysis

Software process data must be gathered with a clear objective or the right information may take long in being recorded. The principles of successful data gathering are, that the data must have a process model made in order to be able to get to the specific information. It is an expensive task and hence the goodwill of the management is essential. The data gathering plan includes the user of the data, the need for the data, method used to gather it and how it will be managed. Measurements can also be objective or subjective, absolute or relative, explicit or derived, dynamic or static, and predictive or explanatory. The main

objective in software engineering is the use of dynamic, absolute, explicit measures to control the work that needs to get done. Considerations in data analyzing are to understand the shortcomings of the process and to effect change by providing the necessary resources. Data gathering is of infinite help in analyzing the software process.

Managing Software Quality

Evaluation of a software development project is to examine the quality of the product. This examination can help one understand the progress made and can establish a framework for improvement. A superior performance can be achieved by the management setting challenging goals, making a plan for accomplishing those goals and keeping track of it by reviews.
Quality of a product can be measured by, development, product, acceptance, usage and repair. During initial project planning, a quality plan is produced. This plan is documented, reviewed, and compared with actual experience. It is possible to keep track of quality performance by estimating the defect removal at each process stage and the effects of quality in each stage can also be found. The purpose of the whole exercise must be kept in mind, namely to motivate action and not to evaluate people. A company that has a quality action plan will always improve in its output whereas a company that always meets its quality plans has little room for improvement.

Defect Prevention

Software development and maintenance can attribute most of its costs to error finding and fixing. Once an error is found by inspections and testing, then rework needs to be done. Defect prevention is mainly instituted to give a focus for improving the process. Its fundamental objective is to make sure errors are not repeated and it is a skill that will take time to learn. Defects can be prevented when programmers evaluate their own errors and this can be done when management makes a strong commitment to quality. The process of defect prevention has a few steps, reporting of the defect, an action plan for removing the defect and then effective tracking of performance. Once this plan is put into action results will show up by the end of six months and a transformation of the organization can be observed.

Automating the Software Process

Automating the software process can improve the quality of work and the strategy can be effectively put into action by being clear about the changes needed to be made, feasibility and an orderly plan to work towards it. Automation of the software environment must be convenient to

use and should have a conceptual schema that will encompass the database, process data, and tool interfaces.
The basic steps required to establish a long- term automation plan are, establish an Automation Focal Point, understand current automation status, make an orderly assessment of the most promising available environments and tools, start work on a common data model for the software environment, and establish a common user interface.
After this the technology currently in use and the one that is being planned must be mapped out. A team of experts that will determine the impacts of the new technology and the support tools for it. Then a review of each project's savings schedule should be made with enough room for flexibility. Once this is completed then the savings schedule for the organization as a whole can be submitted to the management to get the approval needed. Above all, involve the financial people, and do the work in detail. Without line management's commitment to these savings, no approval is likely. It takes time to do it right but there is no shorter way.

Contracting for Software

The software process is essentially an agreement or contract between management and the development organization. One of the benefits of establishing a software process is to be able to assess the status of a project and thus progress can be known at various stages. Auditing of the project helps to see evidence of project performance. There may be different types of vendors and buyers in contracts but when both parties are technically competent then the level of cooperation is greatly increased .The basic system objectives are established at the outset, where specific functional requirements are known , they are stated. A documented plan is then produced foe establishing and validating the requirements and the operational concept against these objectives, and a joint requirements effort defines the system tasks with sufficient precision to permit design to start.
Quantitative process tracking is when check points are given for a job full completed. Quality can be monitored by making a plan where defect injection , removal, inspection coverage, and inspection coverage are included. Establishing SCM, SQA, SEPG(these fall in levels 1 and 2) groups and other steps help to make the software process more mature. But with all this it becomes more important for managers and contracting officers to give technical leadership by setting goals and striving to meet them .

Conclusion

All these approaches to making a mature software process are straightforward and have been proven to be effective. The problems

addressed with regard to the software process have existed for a while. Undoubtedly as the software process evolves newer approaches will be found as people while confronting problems seek solutions. It is best to get started on the road to improvement by getting competent people to apply themselves to the making of a more mature software process. The need is now as the software industry has touched all the important areas of our daily life and it is important that fewer mistakes are made with proper management.

Towards a Project Management Curriculum

- Basics: lifecycle
- Scheduling
- Conflict management
- Change management
- Negotiations
- Program management