Software Process Improvement Efforts in the Economic Directorate

Howard Hogan, Deborah Tasky, David Hiller and Ron Farrar
U.S. Census Bureau, Washington, D.C. 20233-6200

Abstract
Managing software development is thus increasingly a major factor of survey success. Meanwhile, many methods and models for managing software development are being explored in the Information Technology community. This paper examines the plan to improve the way statistical software is developed to support Economic surveys at the U.S. Census Bureau. This effort includes both the Capability Maturity Model and the Team and Personal Software Processes. The paper discusses the general approach, including the management steps necessary to make the transition toward software process improvement. This paper looks at challenges such as the limited management and technical resources available, and areas of progress. It examines the existing culture and the need to bridge the gap between business expertise and software process expertise required to manage organizational and technological change.

Capability Maturity Model; Team Software Process, Software Process Improvement; Economic Programs; Current Surveys, Personal Software Process

1. Introduction

For the past few years, the Economic Directorate has been working to improve our software process by following the principles of Watts Humphrey's Capability Maturity Model or CMM. CMM is an approach to developing mature organizational processes. The emphasis is on management: what management has to accomplish to allow software engineers to work more efficiently. CMM says that management must first institute good project planning, good project tracking, processes to manage requirements, version control, and software quality assurance. CMM says almost nothing about how to do these things, just that the organization must do them.

Meanwhile, software engineers can work to improve their individual work methods. Watts Humphrey developed one approach which he calls Personal Software Process or PSP. It entails a disciplined process each software engineer uses to plan, write and review his/her own work. The engineer tracks how his/her time is spent: how much developing programs and how much attending meetings. Each engineer tracks how many and what kind of errors he/she usually makes and then proofs his/her own work to find and remove those errors. It aims to make the work of programming into real software engineering. The focus, again, is on personal improvement.

Software engineers usually work in teams. It is hard to be disciplined when those around you are not. To fill the gap between the organizational process improvement (CMM) and the personal process improvement (PSP), Watts Humphrey developed the Team Software Process or TSP. TSP defines specific functions that each team must accomplish and makes sure that each function is assigned. On a small team, one person can carry out many functions. It emphasizes careful design and planning by the team. The approach encourages a phased development cycle, with frequent builds, reviews and reassessment of the design, plan and schedule.

PSP and TSP are fairly new. We are developing a program of training, testing and evaluation working closely with Watts Humphrey and others at the Software Engineering Institute at Carnegie Mellon University. We are convinced that this approach is worth a try. If it works, it will revolutionize our work. Even if it doesn't, we think the Census Bureau will understand better how we work and how we can improve.

In the early years of Census Bureau programming, most software programming was done by statisticians. Standards for programming and documentation were rare, resulting in few "best practices." On the other hand, communication was great; the person who developed the software was the same person who analyzed the data produced by the software. The analysts, considered the end user of the software, knew quite clearly what they wanted the software to produce. As programs became more complex, a separation grew between programmers and statisticians. The programmers became more disciplined in software engineering by demanding written specifications and more formal management structures. However, this created its own set of problems, including decreased communications.

The Census Bureau has an established culture in terms of how software development and survey processing is accomplished. In this culture, statisticians document the software requirements as software specifications. Software specifications are not strictly user requirements. They are closer to, but not quite, what the software industry might consider the result of structured analysis and design. They tend to be written in the language of equations, algorithms, and logic tables.

Since statisticians write these documents, tools common in the software industry, such as Entity Relationship Diagrams or Data Flow Diagrams, are seldom used. The software specifications are quite technical and few people

are able to ascertain the true requirements. The specifications do not serve as an effective way of communicating the requirements of the system with others in the organization. This presents a problem when a statistician or programmer leaves the project, and someone else takes over. Communicating the software requirements to a wide community of system users is also a challenge.

Another aspect of the culture is that the statisticians are accustomed to being able to make frequent "improvements" to the system. Myriad informal arrangements have been built up over the years to work around the formal specification-code-test process. In addition, statisticians often use the files that are output from the programs; they need to know the database structure so they can write program scripts and conduct further data analysis. They have a need to know how files are structured, organized and related, and how the data are processed. On the other hand, programmers are also highly knowledgeable of survey processes. They may have worked on processing the same survey for years. A good aspect of this culture is the frequent and successful communication between the analysts and the developers. The downside of this culture of direct interaction is that establishing formal management control mechanisms, such as configuration change control, can be a major challenge, especially when developing large, generalized software systems. Few of the mid-level managers developed written project schedules or assessed project risks. Requirements were poorly defined and there was little coordination of resource allocation.

Five divisions form the infrastructure required to manage the economic programs.[1] There are three subject matter divisions: Service Sector Statistics Division, Manufacturing and Construction Division and Company Statistics Division. These divisions are further divided into functional areas largely comprised of subject matter experts, referred to as analysts, who plan the work such as developing the survey questionnaires and developing the tabulation requirements. They are heavily involved in reviewing, analyzing and editing the data. These divisions drive most of the requirements.

There is a central coordination division: Economic Planning and Coordination Division. This division is responsible for scheduling, allocating resources, documenting requirements and other planning functions. Finally, there is the Economic Statistical Methods and Programming Division that includes both survey methodologists and software programmers.

The economic programs collect data on retail trade, wholesale trade, transportation, communications, services, manufacturing, mining, construction and finance. It is roughly divided into two major program areas, one that is responsible for the Economic Census conducted every five years and one that is responsible for the many on-going surveys, referred to in this paper as the Current Surveys. The Current Surveys include several surveys that are conducted on an annual basis. In addition, the Current Surveys include 11 important "economic indicators." With the exception of the two quarterly surveys, these surveys are conducted monthly.

Those who guide economic policy closely follow the data produced by these indicator surveys. The data have an important affect on the financial markets. Given their sensitive nature, the indicator survey projects have strict scheduling requirements. Indeed, the data release is timed down to the second, with the financial markets reacting within seconds afterwards. In addition, to the Current Surveys and the Economic Censuses, there are a few "periodic" surveys, which altogether define the work environment of the Economic Programs.

Historically, the functional areas within these divisions conducted much of their work within their respective area. Each area had their own programmers, statisticians, analysts, etc. Many had their own computer-processing systems that supported these surveys.

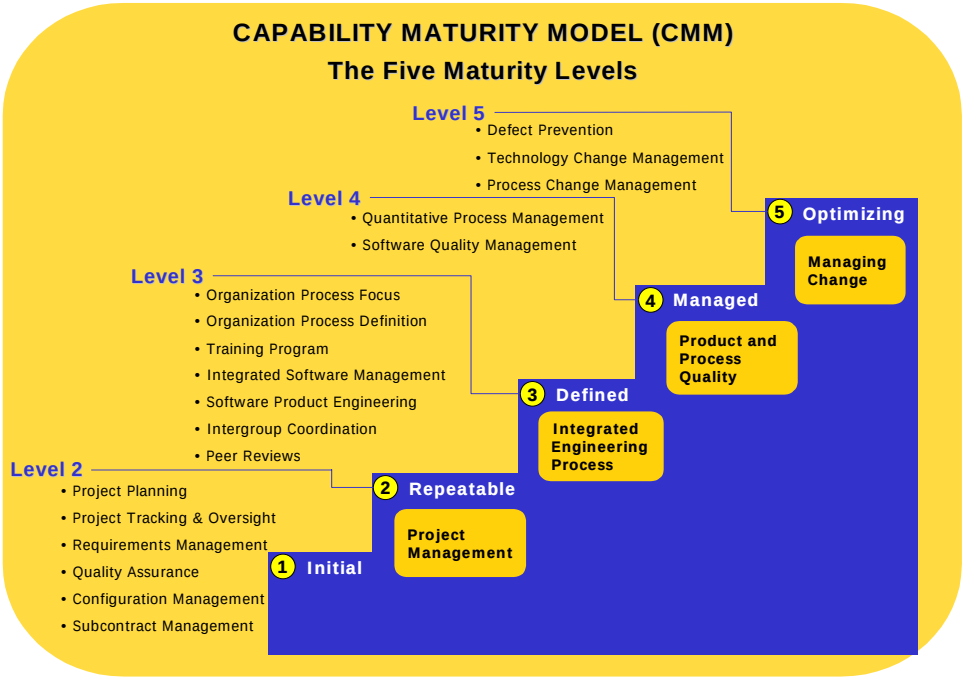
In 1997, a common processing system for all of the Current Surveys was developed. This system became known as the Standard Economic Processing System, or StEPS. The StEPS system was envisioned as an all-inclusive processing system. The development and migration to the StEPS system essentially created a multifaceted working environment. Initially, there was the design and development of the StEPS system. Then there was implementation of new surveys to the StEPS system, which included new functionality for those surveys. In addition, the StEPS system required on-going maintenance and troubleshooting. Lastly, there was the work involved in migrating the various current surveys from the legacy systems to the new generalized StEPS system. The migration work also included new functionality, as well as the process of transferring data and conducting testing in parallel with legacy systems. Since all systems have not yet migrated to StEPS, a certain amount of maintenance is necessary on the old legacy systems.

Since projects were being accomplished successfully meeting critical milestones, there was little doubt the Census Bureau had skilled analysts, programmers and managers. However, over time we relied more and more on software to achieve goals. Through technological advances to meet more complex demands, we realized a

need for a new, more structured framework from which to manage the survey work. The time was right to embrace change and begin an effort to improve the way business was accomplished, specifically within the Economic Directorate of the Census Bureau.

2. CMM Overview

The Capability Maturity Model (CMM), developed by the Software Engineering Institute, is a framework that describes the key elements of a software process (Carnegie Mellon University, 1995, Humphrey 1989). It covers practices for planning, engineering, and managing software development and maintenance activities, and employs an evolutionary improvement path from an *ad hoc* process to a mature, disciplined one. It is the goal of the Census Bureau to evolve toward a culture that integrates software engineering and management excellence with survey research and computing. As depicted in Figure 1, CMM has five levels from Level 1: Initial to Level 5: Optimizing. The Census Bureau's focus is on the key processes of Level 2: The Repeatable Process.



By the time an organization reaches Level 2, its software process is repeatable and under basic management control. Project managers are able to make reasonable estimates and project plans, and to track and control project performance via these plans fairly consistently. [Kenneth M. Dymond, "A guide to the CMM", page 2-1]

CMM-Level 2 includes six key processes:

- Software project planning
- Software project tracking and oversight
- Software configuration management
- Requirements management
- Software quality assurance
- Software subcontract management

The CMM is a proven framework, which can significantly improve the software development process and management of software development and maintenance efforts. To achieve a certain level of the CMM the organization must meet the goals of each of the key processes at and below that level. The CMM is not a particular lifecycle model for software development, such as the "water fall" or "V," nor does it rely upon a particular set of software tools. The CMM describes the practices of an effective software process, but the organization determines how the practices for a successful project will be executed.

2.1 Laying the Groundwork: CMM Pilot Program

The Census Bureau wanted to investigate the possibility of implementing the CMM on a corporate level. It conducted a pilot study that included three different projects from across the organization to see if the CMM framework would work in our environment and culture.

The overall goals of the projects were to improve their software development processes by implementing CMM practices such that they would achieve most of the CMM Level 2 compliance activities.

The Economic Directorate sponsored one project to participate in the pilot study. The CMM Pilot Program, had a positive effect on each of the three pilot projects. It demonstrated that the CMM model (at least at Level 2) could work at the Census Bureau. The approach gained much publicity and acceptance as evidenced by the feedback received from the pilot participants who indicated a desire to continue using the CMM to improve their software development efforts. Although none of the three pilots achieved Level 2, all three pilot projects expanded their original scope and incorporated software process improvement efforts of their own accord. Judging by the increasing implementation of software process improvement activities across the Census Bureau, there is a growing perception that software process improvement is a worthwhile investment.

The results of the pilot program included many lessons that were learned. One lesson is that implementing the CMM is not a cookbook exercise. Software process improvement activities must be targeted toward basic business practices to be effective. The biggest improvements were realized when the focus was on the pilot projects' needs - not just trying to implement each activity as prescribed by the model. The implementation of CMM involves investigating how projects currently do work and evaluating specific activities to improve the problem areas. The CMM can be used most effectively as a resource for suggested improvements.

Another lesson is that grassroots support is key, but equally as important is the need for strong upper management support and backing of the CMM principles. Implementing CMM involves upfront investment and time to be accepted by all. Training is also another important aspect of implementing the model. All of the project participants obtained at least an introductory course of the CMM. However, even with this training, project participants didn't fully understand the benefits of the model until they tried to implement portions of the model.

Other lessons include the realization that implementing the CMM is not an easy task; it requires more resources than ordinarily available to project managers. The pilot program had additional resources to support the effort. Specifically, each of the three projects had a part-time project coordinator, documentation specialist, quality assurance manager, and CMM expert consultant over and above what is normally assigned to Census Bureau projects. We learned we can't implement an overall process improvement effort without considering adding more resources, providing opportunities for training, or considering a smaller scaled, incremental approach that can be folded into normal project work.

With the completion of the pilot program, the Economic Directorate was committed to expanding the Census-wide improvement efforts to some of their major economic programs to help build a new management framework designed to target many of the issues that have existed for years.

2.2 The Challenges

As one might imagine, working in a large, complex environment can be challenging. In an effort to define a strategy for improvement and an effective management framework, we focused our initial efforts on a manageable population of our work, the Current Surveys. The management of the Current Surveys has many problems and challenges of its own. Our first step towards software process improvement involved taking a good look at the problems we faced.

Previous to the introduction of the generalized system, the traditional "chain of command" was successful in managing the production work and legacy systems. Many groups now existed to manage this work. However, clear authority and responsibility was not established.

Changes to the StEPS system came from many sources and in many forms. They came in the form of trouble tickets, e-mails, and word of mouth. They came in the form of user needs requested through the migration process, as well as in the form of whole newly funded surveys. In an early effort to move to a more structured software environment, a change control board was established. However, this board was universally seen as a failure. The principal reason given was that there was no workable system to vet the changes, so the change control board spent its time on trivial changes, better handled elsewhere, and not on important issues of design, schedule and resources. In addition, there was no follow-up or enforcement of the board's decisions. We made the mistake of not clearly defining roles and responsibilities as well as, not holding those assigned to the roles accountable through consistent enforcement. One can appreciate the challenges of enforcing consistent behavior through a cooperative arrangement with other line managers who in the past had only to worry about their own functional areas. This was the beginning of managing the work across multi-layered, organizational boundaries.

Survey migration schedules began to slip, so that surveys

were left on the legacy systems far longer than originally planned. This put the surveys at risk since updates of the legacy system were put-off in anticipation of the movement to the StEPS system. Schedules were lacking in terms of this new environment. Indeed, separate schedules existed for each of the separate projects or surveys; however, far too many did not include the key dependencies that required commitments of resources from others.

A final, but large problem, was that there was no final authority: No one to say no. There was no one to set priorities. There was no person or group to set the overall design and functionality. Those users first in line, with respect to the migration schedule, had their perceived requirements met. Those later, were simply told to wait.

Work in the new environment involved the consideration of new ways of developing requirements, new ways of interacting with others, new ways of managing resources and new ways of considering system changes. The change in technology, which seemed fairly straightforward in concept, proved to require a corresponding change in how we accomplished the work.

2.3 A New Framework

At the beginning of 2002, the Economic Directorate began to systematically tackle the problems associated with the software related work within the Current Surveys. The organization needed to adopt new behaviors that supported the new work environment. We were essentially developing the requirements of a new set of behaviors that we expected others to adopt. So, in our planning efforts, we had many things to decide.

- Where were our trouble spots?
- What new behaviors or practices were needed?
- What was needed such as policies, templates, guidelines, to assist others in making process improvements?
- Who will develop the policies, template, and guidelines?
- When will they be developed?
- How will they look?

The first step was to organize a management group to manage and oversee the project level work of all the Current Surveys. We set up the Current Surveys Processing Management Group or CSPMG. This group was an interdivisional group of managers from the programming division and the planning division. This strategy built on the existing management structure of those responsible for the StEPS processing system. Everyone on the group had clear roles and responsibilities, and established authority for software planning and development. If the management group did nothing more than improve communications between the two divisions, it would prove valuable.

The next step was to gather resources. Here the group benefitted greatly from an earlier Census Bureau initiative, the Software Engineering Process Improvement Group that was formed to develop a standard organizational software process to support software process improvement efforts across the Census Bureau. We were able to use this staff for guidance, coaching, and facilitating. This staff had been working on CMM for quite some time. We strongly recommend including a process consultant in any software improvement effort.

The effort also benefitted from another ongoing program at the Census Bureau, which included extensive project management training and support. As mentioned above, CMM Level 2 is focused on instituting basic project management principles. The group was able to build on this larger project management effort. After a while, we included an additional project coordinator who had gone through the project management training, and who had successfully coordinated other software related projects.

Still, the effort continued to try to borrow as many resources as possible. Those who are actually managing and carrying out projects are very busy people. Learning and remembering to follow a new process is hard. Helping to develop processes, while still getting the work done is too much to be expected. Managers must see "process improvement" as something that will make their work easier, not something that impedes the project. So we added a second process expert to work directly with one of the Current Survey projects to help them apply the practices of good project management.

The management group agreed that improvements in the area of Software Project Planning and Tracking, Software Requirements Development and Management, and Software Configuration Management were necessary first steps.

The guiding principle in the development and implementation of the processes was that they must be folded into our current environment and work with existing resources. Obviously, a process "watered down" to the point of being ineffective would serve no purpose. On the other hand, a process that required too many resources would be rejected and viewed as interfering with "the real work." We wanted our initial efforts to be successful, but not burdensome.

The basic plan included developing core software processes, piloting the processes on a small number of projects to refine, providing training on the new

processes, and implementing the processes on all current survey projects. In addition, the group needed to establish minimum standards to set clear expectations. The standards and processes must be effective in achieving project goals, consistent with available resources, supported with training, and must be flexible to meet individual project's needs.

2.4 Progress Report

The management group spent the first year in developing an overall management plan and a process improvement plan. In addition to the planning, the CSPMG put in place some key management structures to address scope management, time management, and issue management.

The CSPMG developed a Project Charter Template to assist projects in following the project planning and tracking process. The intent of developing project charters is to document the project scope and to facilitate the agreement of the project scope. The process of developing project charters has brought issues and misunderstandings to the table for discussion and resolution. The earlier in the project issues can be resolved, the better. It is easy for each of the user divisions to see each project from their own perspective and priorities, thus setting up expectations that cannot possibly be met. By requiring new projects to develop a project charter and obtain appropriate signatures on the charter, projects participants and sponsors are communicating and resolving potential misunderstandings in ways never before accomplished.

It was developed based on the recommendations of the CMM, as well as our own project experience and training. The process includes the development of a charter, the identification of a defined software life-cycle strategy, the documentation of the high-level system components, the development of written schedules, and sign-offs of key documents.

Since requirements flow in from a score of staffs, we developed a Software Requirements Development and Management process. Requirements range from the trivial (e.g. indent a line, italicize a word) to the major (e.g. get the Quarterly Services Survey in the field by first quarter '04). The goal is to define and agree upon the requirements (baseline) and then control changes by assessing their impact and risk and, again, agreeing to make the change. This process should, of course, be done at the lowest practical level, which is easy to write down and hard to do. We set up a sub-committee to define a workable process to address many of the issues related to poor requirements development and management. The Requirements Development and Management Process includes documenting high level requirements in a Users Requirements Document and documenting detailed software requirements in a Software Requirements Document. This is a substantial change in how we have historically developed software; however, this is one area which all agree needs improvement.

The CSPMG drafted a Software Configuration Management process and an appropriate plan to deal with the software configuration, including change control of the StEPS processing system.

A final, but quite major, piece of the work is developing an overall milestone schedule for the current surveys. This is a monumental effort because it requires identifying and defining the work of the current survey projects and summarizing that work in a meaningful way that can be managed. It requires establishing and agreeing upon clear priorities among the projects. We have instituted schedule review meetings for all chartered projects to give the management group insight into the status of projects so they can manage the resources and evaluate new project commitments.

3. Team Software Process and Personal Software Process

The Team Software Process (TSP), developed by the Software Engineering Institute at Carnegie Mellon University, is designed to allow self-directed teams to design and commit to a development plan and schedule of their own creation. The TSP team is made up of software engineers and technical writers, all of whom are trained in the Personal Software Process (PSP) so they can track their development time and coding defects. A defect is an error in the requirements, design or code that if found in testing would require the code to be changed. Defects are found and fixed throughout the process in order to create a quality product. The emphasis is upon finding the defects early in the process, rather than later in testing, with the goal of producing an error-free final product in less time than would be possible in a less structured approach. Another facet of the TSP approach is the collection of data. All team members record data such as the time spent on a given task and the number of defects created and removed. This data can then be used to evaluate and improve processes and, ultimately, to make future planning more accurate. Throughout the process, the Team is guided by a Team Leader and a Launch Coach. The Launch Coach facilitates the initial launch and answers questions that a team may have about the Team Software Process and Personal Software Process throughout the project.

3.1 The Team Software Process

The Team Software Process begins with a four or five day launch, comprised of nine separate “meetings.”

Management provides goals and requirements to the team in the first meeting of the launch. In meetings 2-8, the team creates a detailed plan for management approval, based on those goals and requirements. Alternate plans are created if needed. The plan is broken down into individual tasks, which are planned in detailed parts of 10 hours or less for the first three months of the plan and more broadly after that. It is important to keep the tasks small so that each team member is completing one or more tasks on most weeks, thus accruing earned value on a weekly basis. This allows for accurate tracking of the status of a plan, which in turn allows for adjustments to be made early if the plan is not meeting the schedule. In the final meeting of the launch, management is informed if the original broad schedule can not be met, and if so, must approve an alternate plan or possibly change or cancel the project.

The team creates a new detailed plan every three or four months in a process known as a relaunch. This allows the team to reevaluate the work done (or still to be done) in the initial plan, and to plan in greater detail those tasks that were only broadly planned in the initial plan. The re-launch also allows for changing circumstances, such as additional requirements, to be included.

The Team Leader should focus on managing the project, not on creating deliverables. The Team Leader does not have to take the programmer version of PSP. The team is self-directed. All members participate and make team decisions. Each member of the team has a role in order to share responsibility and reduce the management burden on the Team Leader. The roles are customer interface manager, planning manager, process manager, quality manager, support manager, design manager, implementation manager and test manager.

The Team Leader does not assign work or make schedules - the whole team does. Each team member is responsible for creating a detailed plan of their own work by creating an estimate of the size and effort for each task. If the workload is unbalanced the team negotiates reassigning tasks.

Each Team Member will have to record the time spent on each task and record each defect found. Time is the actual time spent on a task, not wall time. Normally a Team Member will only be able to spend about 15 hours or less of actual time per week on a project. It is generally advised to never have a team member spend less than half time on a project. Each Team Member will have to report in a weekly meeting their status, the state of their role and the state of the risks they were asked to track.

3.2 Personal Software Process

The Personal Software Process requires a software engineer to record data, review their own work and inspect the work of others. A software engineer creates their own process for doing their own work. For example, they will have design review checklists and code review checklists based on their own defects recorded. This personal process is constantly being improved based on the size, time and defect data being recorded by the software engineer.

PSP improves quality by checking for defects at many stages in the software life cycle. Requirements, High Level Design, Detail Design and Coding all have personal reviews and team inspections. The result then goes through Unit Testing, Integration Testing, Systems Testing and an Acceptance Test by the users.

PSP has two ideas to make it easier to find defects in designs. Each level of abstraction in a project, for example the system, subsystems, programs and modules, should have its own specifications. This allows the User Requirements to be traced down to Functional Requirements for each individual module. This then allows for requirements changes to have their impact traced throughout the system. Each level of requirements also serves as the test cases for that level of testing. In addition each level is divided into black box or 'What' specifications and white box or 'How' specifications. This allows for a separation between testable requirements and a design that is specific to the physical environment. The 'What' and 'How' specs can also be verified against each other.

3.3 Why use TSP and PSP?

TSP and PSP can provide accurate schedules. The schedules that these teams have developed so far seem more accurate and have more meaning to the team members than do the schedules we have previously developed in tools such as Microsoft Project. For one thing, the "plans" developed by the TSP team are much more personal to each developer. Each developer has his/her own individual plan and thus feels a degree of ownership for at least a portion of the overall plan. Secondly, the detailed tasks in the plan are broken down into less than 10 hour segments, thus increasing the precision of each estimate. Finally, the TSP tool makes it very easy to determine project status by recording actual numbers against plan numbers.

TSP allows an organization to manage their process while leaving the details of the work to the teams. The TSP plan is developed collectively by the team members, rather than by members of management. The team members can most accurately determine the amount of effort needed to accomplish the tasks assigned to them,

and their estimates become more and more accurate over time as they gather data on their work. The team decides on the best distribution of work and decides upon redistribution when necessary. These decisions have historically been management purview. Allowing team members to make these decisions increases their feelings of ownership and their commitment to the team.

PSP helps to create modules that are virtually defect free after unit testing, which greatly reduces the time needed in subsequent tests. PSP stresses a series of inspections, performed by the developers and other members of the team, that ensure defects are discovered and removed as early as possible in the development process. Software engineers perform inspections of high-level designs, detailed designs, and actual code, so there are three thorough examinations of the programming logic before the code is ever finalized. This results in the removal of the majority of defects prior to unit testing, which has been historically where most defects are uncovered.

The Team Software Process and Personal Software Process help individuals employ the principals of the Capability Maturity Model in their own work.

The TSP tool is a great motivator for team members. The TSP team uses a "tool," developed by SEI in Microsoft Excel, to develop and manage their plan. This tool makes it easy to quickly determine overall status or to modify the plan, if needed. It also allows team members to examine potential impacts of unexpected events (risks) by quickly developing an alternate plan based upon the original plan. All of this proves highly motivating to the team members. Each week they receive feedback from the tool on their progress in terms of earned value. They can quickly assess the impact of unexpected developments, thus reducing the uncertainty associated with that unexpected event.

3.4 Getting Started with TSP/PSP

TSP and PSP require a major commitment, changing the way people work. The first step is for management to make the commitment and let that commitment be clearly understood. The next step is to begin the required training, which is extensive and expensive.

We started by the senior managers of the Economic Directorate attending the 1½ day *Team Software Process Executive Strategy Seminar*. Each attendee read Watts Humphrey's *Winning With Software: An Executive Strategy* book prior to the seminar. We were exposed to the key elements of TSP and PSP and even had to do some number crunching ourselves with team measurement data. After this class we identified potential coaches and two pilot projects.

What followed were three basic training classes: One for our software engineers, one for non-software engineer team members and one for team leaders and team managers.

We learned that it is best to train the managers before conducting the other training. The three day course *Managing TSP Teams* should be attended by the team leaders and any line manager of a TSP team member. The first day is basically the executive strategy seminar material and then days two through three go into more details of how TSP teams work and how metrics are captured.

The software engineers take two classes, *PSP I for Engineers* and *PSP II for Engineers*. Each class lasts for a week and consists of lectures and a series of five programs to learn PSP skills. In the first week the engineers learn to estimate effort and defects, record data and begin their personal checklist of defects. There is a homework assignment between the two classes. The second class concentrates more on the processes of designing and reviewing personal work. The detailed recording of defects and one's time has not been met enthusiastically.

Non-software engineer team members took a two day class titled *Introduction to Personal Process*. We were first told that team leaders did not need to attend this class, but learned later that this would have been very beneficial. Instead of writing programs, class members worked through a balancing a checkbook exercise to learn skills of recording time and defects.

3.5 Pilots

The Quarterly Financial Report or QFR migration to StEPS was the first project to employ TSP and PSP within the Economic Directorate. It was also one of the first projects to take advantage of the procedures and document templates developed by the CSPMG. While the TSP has proven beneficial to the management of this project, we have come to understand that it must be modified to some extent to adapt to the unique culture of the Census Bureau. TSP is designed with a fully-dedicated staff of software engineers in mind. This situation almost never exists in the Census Bureau. In all cases, team members have other responsibilities beyond the QFR project. Thus, no single team member is devoted 100 percent of the time to the QFR project. Additionally, the TSP concept assumes that the team is made up entirely of software engineers. We at the Census Bureau typically divide the work of software development between engineers and technical writers. Still, we have been successful in molding the TSP concept to this situation and have produced a highly motivated and successful

team.

QFR was our first pilot project. Management was extremely impressed with the team presentation at the end of the launch. The team believed they could do everything asked of them in the time requested. Since this time, the team has conducted two relaunches. While a larger team had been meeting for nine months prior to the first launch, the TSP team was reduced to just the software engineers and the processing analysts, i.e. not the subject matter analysts. In our culture, we don't recommend teams leaving out the subject matter analysts. The benefits of the launch really bring a sense of team and commitment and should be shared with our subject matter analysts as well.

The BSR06 RU pilot project did consist of subject matter analysts as well. An interesting point in this pilot project is that at the first meeting of the launch, management added a few additional tasks never mentioned before. At meeting eight when the team presented their recommendations, they told management, if they did everything asked for, they would miss the target date by several months. They proposed a scaled back plan that had them only one week past the target date. This reduced scope project was accepted by management. This is much better than waiting a year and then telling management they won't make the date!

The BSR06 RU pilot project didn't have as well defined processes as the QFR project and thus their launch planning didn't match how they actually did business. Because of this, much of their project estimates and metrics were meaningless, and this was addressed at their relaunch.

Both pilot projects had positive reactions to the launch. It can make for long days, but the team felt they accomplished much more in these four to five days than they would have over several months in regards to the team plan. The team members learned to work together and they felt committed to the plan they presented to management. Management had the opportunity to speak up if they didn't accept what the team proposed.

We recommend that all team members be trained prior to the launch, although not all were and they were still successful.

As teams have progressed, we have learned that not all team members are comfortable entering their time and defect data into the tool, so we will be addressing better training. We also feel that onsite coaches will now aid in this issue. We are also finding that the planning of TSP helps address early on areas of concern and helps with accountability. Namely, instead of waiting until the end of a project and hope software engineers can catch up with late requirements, specifications, etc., we can see early on if earlier commitments are not being met and address them as soon as possible.

We are now in the process of some additional pilots involving staff from our census area.

4. Conclusions

CMM is a framework to develop improved processes to manage software development. CMM Level 2 reflects generally well-accepted practices of project management, coupled with software quality assurance. The Census Bureau has formally committed to CMM as evidenced by a Software Development and Maintenance Policy signed by Census IT managers. We are also committed to project management through the availability of a Project Management Certification Program that affords managers the opportunity to enrich staff through structured project management training. The Economic Directorate has begun implementing CMM-Level 2. We have generally found the model a useful framework that helps to organize our process improvement efforts. The processes that CMM suggests starting with do indeed correspond to the processes that we believe need improvement in our area. Although we currently have no way to quantify our successes, we believe that CMM presents a useful framework.

The TSP and PSP approaches look promising and have been reasonably well accepted by the teams involved. However, our efforts have just started and a full evaluation is some time off.

However, even within that framework, there is much work to do. Initially, core staff needs to be trained on CMM and basic project management principles. These same people, plus others as well, need to be trained on TSP and PSP. Secondly, a steering group or management committee needs to be established to oversee the development and implementation of sound software practices. We recommend building upon existing management structures. The software process improvement initiative needs to be managed as a project. To be successful, this process must involve and be supported by senior staff. It must also include the involvement of the people who are expected to implement the processes or else they won't support the initiative. The notion of meta-planning is a necessary component of the re-engineering process.

As noted above, CMM does not mandate a particular process of software development, nor does it rely upon a particular set of software tools. It does mandate that these

issues be explicitly addressed. Many organizations, including the Economic Directorate, had never seriously addressed these issues previously. Forcing this discussion is an important aspect of a software process improvement effort. However, the added time for this discussion must be considered.

Project management and process improvement can be made to seem as an unobjectionable and straightforward effort. In fact, nothing involving large groups of humans is simple or tidy. The current system, no matter how messy, is working for some people. Some people will have long standing working relations that allow them, effectively, to elevate their perceived needs to the head of the queue. An orderly process to determine priorities and control changes will mean that some people will no longer have their 'needs' met, they will be placed lower in the queue or simply be told 'no.' Do not expect these people to be supportive of the initiative. These are not insurmountable obstacles, but one should not be shocked or discouraged when they occur.

The second obstacle will be lack of resources, especially time. The Census Bureau has made a major commitment to CMM. The Economic Directorate has invested in TSP and PSP as well. Still, the task for the individual software engineers or project managers can seem daunting. They will still need to 'get the work done,' i.e., produce the survey. They will need to adopt new processes, which means taking the time to learn both the mechanics and the substance of the new process. This will be an added burden, with promised payoff in the future. For those doing the meta-planning, they will have to develop the new processes, templates, training, etc. This investment in the future is, in our view, worthwhile. The alternative is the current chaotic system. However, it is an investment that takes resources.

Moving to CMM, plus perhaps PSP and TSP, requires more than planning. It requires a change in culture and a change in the business processes. Changing culture is the hardest part of any re-engineering effort. A necessary condition is for people to see and believe that the new framework of developing and managing software-related projects is here to stay, and that the work they produce will be better as a result of it.

All paradigms for improving quality emphasize the needs for a strong management commitment. Fortunately, we are getting such strong management support, if only because the current situation is acknowledged to be unacceptable. The processes we are developing are increasingly being accepted and used. Some project leaders have adopted the entire package as the best way to ensure a successful project. Others have adopted only parts. With quality improvement, one can never say, "We have achieved success," only that things are improving. Thanks to hard work and senior management support, we are improving our process, although never as fast as we would like or dream. To summarize the moral of our story then, continuous improvement can be hard and hard to see. Don't get discouraged.

References

Adhern, Dennis M., Clouse, Aaron, Turner, Richard. CMM I Distilled. 2001. *A Practical Introduction to Integrated Process Improvement.* New York: Addison-Wesley.

Burwick, Diane M. 2001. *How to Implement the CMM.* BPS Publication, Ft. Knox, KY.

Caputo, K. 1998. *CMM Implementation Guide: Choreographing Software Process Improvement.* Reading, MA: Addison-Wesley.

Carnegie Mellon University, Software Engineering Institute 1995. *Capability Maturity Model: Guidelines for Improving the Software Process,* Addison-Wesley Logman, Inc.

Carnegie Mellon University, Software Engineering Institute, (No Date) *Capability Maturity Model for Software,* http://www.sei.cmu.edu/cmm.

Dymond, Kenneth M. 1995. *A Guide to the CMM: Understanding the Capability Maturity Model for Software.* Maryland: Process Transition International, Inc.

Humphrey, Watts S. 1989. *Managing the Software Process,* Addison-Wesley

---*Winning With Software: An Executive Strategy, A Discipline for Software Engineering*

---*Introduction to the Personal Software Process,*

---*Introduction to the Team Software Process*

Paulk, M., Weber, C., Curtis, B., and Chrissis, M. 1994. *The Capability Maturity Model: Guidelines for Improving the Software Process.* New York: Addison-Wesley.

About the Authors

Howard Hogan is Chief of the Economic Statistical Methods and Programming Division (ESMPD), Bureau of the Census, Washington, DC 20233. Deborah Tasky is Assistant Division Chief (ADC) in ESMPD. David Hiller is Technical Advisor in ESMPD. Ron Farrar is in the Economic Planning and Coordination Division. This paper reports the results of research and analysis undertaken by Census Bureau staff. It has undergone a Census Bureau review more limited in scope than that given to official Census Bureau publications. This report is released to inform interested parties of ongoing research and to encourage discussion of work in progress.

Footnote

1. A division has roughly 150 to 250 people, led by a Division Chief. Divisions are divided into 3 to 5 Assistant Division Chief (ADC) areas. There is a total of 900 people who work in these divisions. Two additional divisions have their own systems and are outside the scope of this paper.