TSP SYMPOSIUM 2010

CHANGING THE WORLD OF SOFTWARE ENGINEERING

5th Annual Software Engineering Institute
Team Software Process Symposium

September 20-23, 2010 • Omni William Penn Hotel, Pittsburgh, Pennsylvania

Software Engineering Institute | Carnegie Mellon

www.sei.cmu.edu/tspsymposium

The ideas and findings in this publication should not be construed as an official U.S. Department of Defense (DoD) or Carnegie Mellon position. It is published in the interest of scientific and technical information exchange.

Trademarks and Service Marks

The following authors granted special permission to reproduce the following documents:

How to Teach Programming – Proposal for Introducing
PSP into a University Curriculum
©Barry Dwolatzky

Introducing PSP/TSP Massively in Mexico
©Héctor Joel González Santos

Will TSP Shape the Future of Software Development
in South Africa?
©Barry Dwolatzky, Lisa Lyhne, Tamasin Bossert, Alok Goswami

Illuminating the Intersection of TSP and CMMI
High Maturity Process Performance Models
©Robert Stoddard, Shigeru Sasao, Dave Webb, Jim VanBuren

New Team Software Process Paths: Systems
Engineering Team Uses TSP
©Daniel M. Wilson

Achieving Academic Success Using the Team Software Process
©Berin Babcock-McConnell, Saurabh Gupta, Jonathan Hartje,
Marsha Pomeroy-Huff, Shigeru Sasao, Sidharth Surana

AIM Case Study: Moving from TSP to CMMI ML3
©Oscar A. Mondragón, Edgar D. Fernandez

Team Software Process Symposium Proceedings

EDITOR'S NOTE

The experiences and ideas presented in these papers are those of the authors, and not necessarily of the SEI. Most individuals and teams customize the Team Software Process (TSP) and the Personal Software Process (PSP) to best fit their own needs. The processes were designed to be flexible and support customization. As always, use judgment before following advice from others. Take special caution when modifying basic principles of the PSP and the TSP. Examples include the use of personal data for anything besides personal improvement and providing team status.

TSP Symposium 2010 Program Committee

William Nichols, Software Engineering Institute
TSP Symposium 2010 Technical Chair

Lana Cagle, U.S. Navy

Tim Chick, Software Engineering Institute

David Saint Amand, U.S. Navy

Rafael Salazar, Tec de Monterrey

Rajan Seriampalayam, Oracle Corporation

Karen Smiley, ABB Inc.

Kathy Smith, HP Enterprise Services

David Webb, U.S. Air Force

How to Teach Programming – Proposal for Introducing PSP into a University Curriculum

Barry Dwolatzky, *JCSE at Wits University, South Africa*

1.1 INTRODUCTION

In this paper I propose a change in the way that university students are taught to write computer programs. The impetus for proposing this change arises from my own first-hand experience in learning PSP. In October 2009 I attended the "PSP Fundamentals" course presented in Johannesburg by the SEI as part of a TSP adoption programme in South Africa. I followed this up with the "PSP Advanced" course in December 2009 and PSP Instructor training in January 2010.

After having read and heard about PSP over the past few years, it took the experience of actually learning PSP myself to convince me that the way in which my colleagues and I have been teaching programming at South African Universities for several decades needs to undergo a radical re-think.

1.2 LESSONS LEARNED IN MY OWN PSP TRAINING

I wrote my first computer program in 1971 in Fortran IV. It ran on an IBM 360 mainframe, and had to be submitted as a batch job on a stack of punched cards. The results were collected a day later as a printout. In the 40 years since writing that first program I've written many programs, large and small, in a range of different programming languages, individually and in teams. I've also taught programming to generations of university students.

With 4 decades of programming experience under my belt it is not surprising that I thought of myself as a "good," even an "excellent" programmer. With a high level of self-confidence I started the first program of the PSP training in October 2009.

As part of the PSP training, I kept a careful record of the actual time taken to write the program. The time log was broken down into several component phases:

Phase	Actual time [Minutes]	Percent of total
Plan	3	1.8 %
Detailed Design	5	2.9 %
Code	97	53.1 %
Compile	27	14.8 %
Unit Test	49	26.8 %
Post Mortem	1	0.7 %
TOTAL	182	100.0 %

I was interested to note that **94%** of the 182 minutes it took to write this program was spent on three phases – code, compile and test. I became aware that these three phases were not distinct activities, but were carried out as one "super phase" in which the cycle: code→compile→test was repeated many times.

Before writing "Program 2" I was required to use the PSP tool to estimate the size (in lines-of-code [LOC]) of the final program. As an input to this estimation I developed an initial design of my program. Based on this design, the PSP tool required me to enter the number of "parts" (i.e. functions, classes, etc.) that would be needed, and an estimate of the complexity of each "part." Using this information the PSP tool estimated the size of the program, and the time required.

The estimates of size and time, based on my initial design, were as follows:

Estimate of size for "Program 2"	135 LOC
Estimate of development time for "Program 2"	240 minutes

As it turned out, both of these estimates were substantially inaccurate. My actual size and time data on completion of "Program 2" were as follows:

Actual size of "Program 2"	361 LOC
Actual development time for "Program 2"	530 minutes

As with the first program, in developing "Program 2," the vast majority of my time (91%) was spent repeating the cycle: code→compile→test. This cycle is, in fact, a combination of iterative design and debugging. I also observed that this cycle is extremely inefficient and unpredictable.

Writing Programs 1 and 2 provided me with a "baseline." I believe that my behaviour in writing these programs accurately represented the way I have worked for nearly 40 years. The PSP Training aims to change the behaviour I demonstrated in Programs 1 and 2. Did it achieve this change?

In terms of estimation, my data at the end of the training (i.e. for "Program 7") were as follows:

	Estimate	Actual	Error [%]
Size [LOC]	126	118	-6.4 %
Development Time [minutes]	176	242	37 %

More important than the accuracy (or lack thereof) of my predictions is the fact that the estimates for "Program 7" were not based on "engineering judgement," or on standard historical data stored in the PSP tool, but on *a careful analysis of* ***my own*** *historical data*, collected from Programs 1 to 6. This certainly represents both a change and a significant improvement in my behaviour!

The other key lesson became evident when I analysed **how** I developed "Program 7." When one breaks the 242 minutes spent in developing the program into distinct phases the data are as follows:

Phase	Actual time [Minutes]	Percent of total
Plan	10	4.1 %
Detailed Design	45	18.6 %
Design Review	21	8.7 %
Code	79	32.6 %
Code Review	21	8.7 %
Compile	22	9.1 %
Unit Test	32	13.2 %
Post Mortem	12	5.0 %
TOTAL	242	100.0 %

In writing "Program 7" I spent only **55%** of my time in the code, compile and test phases. These phases were now *distinct and separate*. All the rest of the time was spent planning, designing, and reviewing – in other words ***thinking*** *about what I was going to do, and* ***checking*** *what I had done.* Before the PSP training I had spent less than 10% of my time on these activities.

1.3 QUALITY: THE SECRET INGREDIENT

Forty years of writing programs taught me that software **always** has bugs. Some are found during unit testing, others during system testing and (worst of all) are those that appear "in the field" after the development has been completed. In my experience programmers learn to accept that, since programming is a complex and difficult task, there will always be some defects that cause programs to function incorrectly in certain circumstances. What did the PSP training teach me about this received wisdom?

In writing all seven programs, I recorded every defect that I found – no matter how minor. For each defect I specified where in the development process it was injected and where it was removed. The training taught me that by dealing in a careful and coordinated way with defects, and by finding and removing them as soon as possible, PSP has the potential to help me remove the major source of uncertainty in the development process. By examining my own data I've understood that locating defects during unit testing is a slow, unpredictable, frustrating and wasteful activity. Most importantly, I've also understood that this waste of time and effort is not necessary. Although I didn't, during the PSP training, achieve the level of performance in defect removal that I should be capable of, I certainly gained a clear understanding of how defects affect productivity and predictability.

The PSP training taught me that early and efficient defect removal is not only the key to good quality. It is also the best way to ensure that time estimates are far more accurate. Quality certainly is the secret ingredient in PSP!

1.4 TEACHING STUDENTS TO PROGRAM

In universities around the world students are taught to program. The ability to write a computer program in some general-purpose language is a skill required in science, engineering and other academic disciplines. Many students also go on to write software professionally. There is an ongoing debate at universities and, more generally in the Information and Communication Technology (ICT) sector, on how to teach programming.

The most common approach in teaching students to program is to focus on language syntax, algorithmic skills and abstraction mechanisms. In 2001 the two major professional bodies in the ICT sector, the Institute of Electrical and Electronic Engineers (IEEE) and the Association for Computing Machinery (ACM) set up a large international "Joint Task Force" that developed a "Guide to undergraduate degree programs in computing." This document includes suggestions on how to teach programming. In one of the components of this guide [ACM/IEEE 2001] the authors state that "concentrating on the mechanistic details of programming constructs often leaves students to figure out the essential character of programming through an *ad hoc* process of trial and error. Such courses thus risk leaving students who are at the very beginning of their academic careers to flounder on their own with respect to the complex activity of programming."

By the end of the PSP training I felt inspired to bring the lessons I had learnt into my own teaching of Electrical Engineering students at Wits University and to influence some of my colleagues at other South African Universities to do the same. We have certainly been guilty of "leaving students … to flounder on their own" when it comes to the aspects of programming that I learnt on the PSP training. A simple solution would be to incorporate the "standard" PSP training as a stand-alone module in the existing curriculum. I wanted, however, to explore ways of radically re-thinking how I teach students to program.

In contemplating a radical re-think in how we teach programming I remembered how I had learnt 40 years ago. As I mentioned earlier, my first experience in writing programs involved punch-cards and batch processing. One of the good things I learnt about how to program was that I paid dearly for defects that escaped into the compile and test phase. Each defect in compile or test cost me a day or more in turn-around time. This taught me to review my design and my code before I submitted my deck of cards. This is the same lesson that Watts Humphrey incorporated into PSP.

If I learnt to program in this way why have I not retained this discipline? The answer is simple: I became lazy and was seduced by the apparent ease of working within an interactive environment.

The question then is how to encourage, or even force, students to follow the principles of PSP and then stick to this way of working? I believe that the answer lies in the only currency that students respect, namely grades awarded in assessment.

My proposal is that learning to program should proceed in four stages:

- Stage 1- Learn the syntax of a language. This is covered in a typical introductory programming course. The focus will be on syntax, algorithmic skills and abstraction mechanisms. It is essential that the student is able to write a simple program in a general purpose language before PSP concepts are introduced. While the focus is on learning the language, something must be done at this early stage to resist the onset of bad process habits. This should take the form of a code review step before the compiler is run. This could be supported by providing the student with a "standard checklist" that she/he can modify to reflect some individual typical defects. To encourage the use of code reviews students should receive bonus marks in assessment of laboratories and tests if the first compile results in less than a specified number of compile errors.
- Stage 2 – Advanced programming, using a process and collecting data: This should cover advanced programming concepts together with the introduction of a PSP-like process and collection of time and defect data. While the focus is still on developing good algorithmic and syntax skills, process and data should become important. Again marks awarded in assessment should reinforce the adherence to process and accurate collection of data. It is important that in assessing work, the content of the data should not be examined. All that is required is that data must be collected. Defect information must be used by students to develop their own design review and code review checklists.
- Stage 3 – Formal PSP training: In Stages 1 and 2 the student should have become a competent enough programmer to gain maximum benefit from PSP training. In this stage the full 7 or 8 program PSP course should be taught. The focus is on estimation based on personal data, design and design review, and developing and improving one's own personal process.
- Stage 4 – a team-based capstone project using TSP: The capstone project should introduce TSP concepts and should reinforce PSP principles.

1.5 CONCLUSION

This paper presents a high-level proposal for introducing PSP-like principles into a sequence of programming courses within a university curriculum. It still remains for this proposal to be incorporated within an actual curriculum and then piloted. I believe very strongly that there is something fundamentally wrong in the way students are taught to program, and that the solution lies in making significant changes in existing courses and curricula.

REFERENCE

ACM/IEEE 2001

"Computing Curricula 2001: Computer Science," Final Report, Dec. 15, 2001, ACM/IEEE, p. 27.

BIOGRAPHY

Barry Dwolatzky

Barry is Professor of Software Engineering at the University of the Witwatersrand, or Wits, in Johannesburg, South Africa. He is also Director and CEO of the Joburg Centre for Software Engineering (JCSE) at Wits.

Barry graduated as an Electrical Engineer from Wits in 1975. He stayed on to complete his PhD in 1979. Between 1980 and 1989 he lived in the United Kingdom working as a post-doctoral research associate at the University of Manchester Institute of Science and Technology (UMIST) and Imperial College, London. He joined the company GEC-Marconi as a researcher in 1985 and left in 1989 to return to South Africa. Since returning to Wits he has been on the staff of the School of Electrical and Information Engineering.

Barry has published extensively in academic journals, has presented at conferences and has successfully supervised 5 PhD's and over 30 MSc research students. His major current interest is in promoting the growth and competitiveness of the South African and African software development sector. He believes that this can be achieved by promoting the adoption of sound software engineering principles and practices.

He is chairman of the South African Chapter of the IEEE-CS. He is also a certified CMMI and PSP Instructor.

Introducing PSP/TSP Massively in Mexico

Héctor Joel González Santos, *Kernel Technologies Group*

How can Mexico's SME (Small and Medium Enterprises) accelerate their process maturity to achieve organizational processes and quality? How can Mexican engineers be prepared to compete on a global high competitive environment? The answer is PSP and TSP, We are convinced that TSP is the key to achieve process, product and quality maturity and first we shall start by training our engineers.

The purpose of this proceeding is to show Kernel's strategy and results when implementing this first phase of our TSP implementation in SME's by training 250 engineers in diverse companies (50 Mexican SMEs). These companies were selected from a group which had already implemented the Mexican standard for IT VSEs (MoProSoft), to help them increase the performance of their development teams. We will also present Kernel's initiative to achieve at least 100 PSP Certified Developers; further we will demonstrate our conclusions: benefits of the program, performance results from the engineers, and challenges when implementing PSP training massively.

Which are the next steps? How can the SEI help in the implementation of TSP Projects in SMEs? What shall we do as a SEI's Partner?

1.1 IMPORTANCE ON SOFTWARE DEVELOPMENT AND STANDARDS

Why is it important to verify and assure standards in the software development process? Good practices in software development and maintenance are the keystones on which the organization involved in this IT Industry focus the achievement of their business goals and strategies. It is important to certify organizations that are capable of meeting quality and schedule standards. Following these standards, clients are assured that organizations are capable and properly comply with its objectives in the development of software quality.

1.2 MOPROSOFT SUPPORT

The Mexican government IT assessors, considered that none of the international Process improvement models, such as CMMI and ISO/IEC 12207/15504, were not suitable for SME's. In Mexico 83% of the IT Industry is very small, from 2 to 10 resources. In this environment, the Mexican government decided to create MoProsoft, which is the Mexican model/standard for software development process improvement. MoProSoft is based on CMMI process areas levels 2 and 3 and inspired by the framework of ISO / IEC 15504 for its evaluation. MoProSoft has adopted and included different practices from PMI, CMMI and ISO 9001:2000 to provide a model that fits Small and Medium IT Industry.

At this time we have 263 Clients (SME's) who have implemented MoProsoft model. Kernel is positioned as a leader in its implementation, 65% of all certified companies under a MoProSoft level have been implemented by Kernel. Our clients are continuously improving and with the support of the Mexican Government, PSP and TSP is being introduced to accelerate process maturity and the competiveness of SME's in Mexico.

1.3 STRATEGY FOR IMPLEMENTING AND MAINTAINING TSP/PSP

This year we offered our clients PSP and TSP together with MoProsoft. Some companies, during their PSP training, implemented PSP practices in their organizations and this motivated them to accelerate their improvement process.

Kernel's software factory tested a pilot, implementing TSP to accelerate process maturity and achieve MoProsoft Level 2 with minimum effort. The project was successful since TSP integrated perfectly with two of MoProsoft Roles (DMS-Development and Maintenance Management and APE-Project Management).

Support from SEI was great, we managed a Second Amendment during this period which facilitated a discount on the seats for each students enrolled in these PSP Courses.

SEI supported us with the development of two internal testing centers and two proctors located at Mexico City and at Monterrey, they provided us with discounts on the purchase of two bundles of 100 exams.

1.4 109 MEXICAN ENGINEERS PSP CERTIFIED

50 SME's participated in this project; eight companies from Queretaro, thirty one from Mexico City, and eleven companies from Monterrey. Each of them sent around five engineers to PSP training. Overall 250 Engineers were trained from November 2009 through March 2010.

The course was structured depending on the location. In Mexico City we created 5 groups each of around 30 Students. The five groups took the same lecture each week in parallel. In took 2 months to teach PSP Week 1 and Week 2. In Monterrey and in Queretaro the courses were taught continuously day by day.

Our team consisted on one onsite Instructor accompanied by an Instructor assistant and three remote Instructors who graded the students' assignments.

Before the PSP Examinations, we reviewed the PSP Body of Knowledge; 16 hours were dedicated to review all concepts.

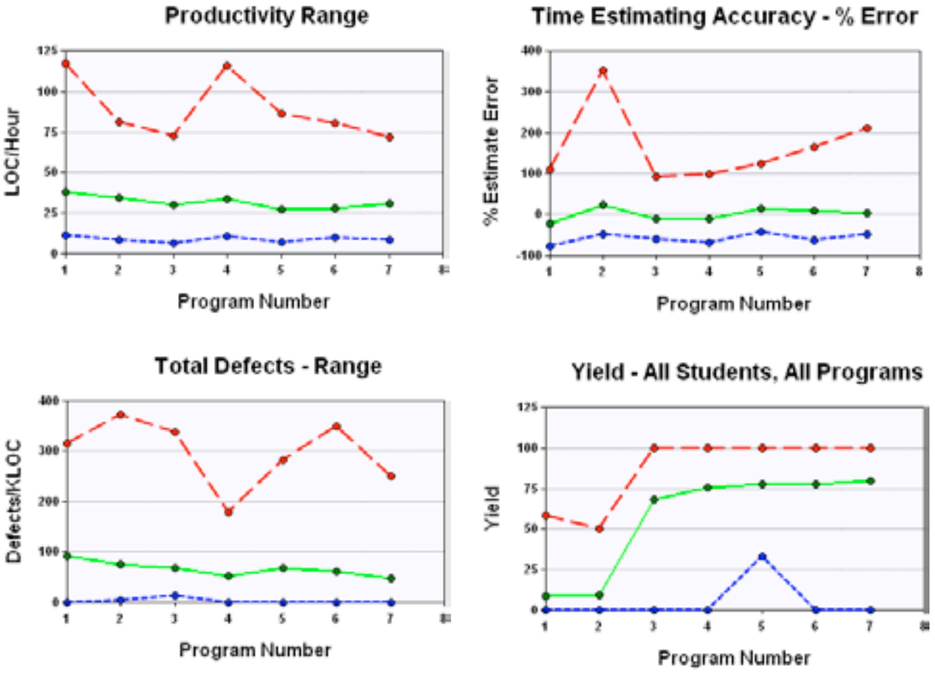


Figure 1: Performance of 54 Mexican Engineers who completed all PSP Course Assignments

Sixty-four percent of the Students who took the exam successfully achieved the certification. Figure 1 shows the performance of those engineers who completed all assignments throughout the course.

Currently from all PSP Certified Engineers in the world, Kernel has contributed with the 20%.

1.4.1 LESSONS LEARNED

During the course the following were some struggles we had to confront to meet our objectives in achieving 109 certifications.

- Balance between high priority job activities and PSP Assignments. VSEs engineers hardly submit assignments on time; during lab sessions, on-site instructors pre-approved students' assignments.
- 20% of the Students dropped from the course. Such probability was given when the course was scheduled in a long format course (one course day at a week).
- English language was a factor that in some cases was an obstacle for the certification. During the PSP BOK review sesión, we related PSP spanish terms to english languaje. An spanish version of the exam would help achieve more certifications.
- Limited availability of Testing Centers for students from different regions. Kernel developed two authorized SEI-Kryterion Testing Sites for students at Mexico City and Monterrey. An opportunity would be to implement web-proctored exams.

1.5 NEXT STEPS

Kernel Technologies is working along with the government to raise grants for future projects. One of Kernel's strategies is to implement TSP to accelerate organizational maturity reaching CMMi. We will focus on our clients who have adopted MoProSoft and seek continuous improvement and growth.

With our current customers the next steps is to start a TSP Pilot Project to achieve a higher maturity level in MoProSoft or to implement CMMi in their organizations. Currently we have several companies identified that are convinced that TSP is the way to improve software quality and schedule prediction. They plan to launch a TSP pilot project.

We are working to Implement PSP/TSP across Latin America with CMMi organizations. We are partnering with local organizations to provide TSP Product Internationally, currently we are talking to organizations in Colombia, Chile, Argentina, Perú and Ecuador.

We are seeking a Strategic Partnership with the SEI to address the possibility to lower TSP implementation costs to enable implementing TSP massively in small enterprises.

1.6 REFERENCES/BIBLIOGRAPHY

[2004] (Reporte PROSOFT 2004, Secretaría de Economía)

[2005] (Modelo de Procesos para la Industria de Software versión 1.3, Agosto 2005, Hanna Oktaba)

[2010] (PSP Developers Listings, July 2010, SEI)

BIOGRAPHY

Héctor Joel González Santos
PSP Instructor
Kernel Technologies Group

Héctor González is a member of the process development staff at Kernel Technologies. He was born in Monterrey Mexico. He has a BS in software development engineering. He studied mobile programming and criptology at University of Kuopio in Finland and currently he studies a MS in Information Technolgoies at Carnegie Mellon and ITESM in a dual degree program. He has over 8 years' experience programming software application implementing languages such as Java and .net in the IT Industry. He has developed as an analyst and has developed 50KLOCs BPM applications with Savvion Process Modeler and JBPM.

He is a certified PSP developer, PSP Instructor and Candidate TSP Coach.

Prior to joining Kernel as a PSP Instructor and TSP Coach, Héctor travelled to India along with the Mexican government in a Faculty Enablement Program held at Infosys in Mysore India. He studied India's software development culture and methodologies and implemented a similar framework in Kernel's Software Factory.

Héctor has also implemented MoProSoft in several organizations. At CEIS Avance, when developing Kernel's Software Factory, he implemented TSP disciplines, MoProSoft models and Agile methods to develop the software factory development process. He accelerated the process of certification for MoProsoft Level 2 through the implementation of TSP.

As a PSP Instructor he has trained around 300 students. Recently in a PSP Initiative held at Mexico City, Queretaro and Monterrey, he trained 250 engineers in PSP and TSP practices.

Actually, Héctor is working on a strategy to implement PSP/TSP massively in Latin America.

Will TSP Shape the Future of Software Development in South Africa?

Prof Barry Dwolatzky, *JCSE, Wits University, South Africa*
Lisa Lyhne, *Dariel Solutions, South Africa*
Tamasin Bossert, *Nedbank, South Africa*
Alok Goswami, *Nedbank, South Africa*

1.1 OVERVIEW OF THE SA ICT AND SOFTWARE DEVELOPMENT SECTOR

Since the dawning of the Information Age in the 1950's South Africans have had a deep and active fascination with digital technology. A measure of the level of interest in computers in those early days is the fact that the "Computer Society of South Africa" (CSSA) was established in 1957 to serve the interests of "computer professionals." This makes the CSSA only a few months younger than the world's oldest organisation representing IT professionals, namely the British Computer Society.

In the 1950's and 60's South Africans notched up several "world firsts," finding ways to use computers to automate processes in mining, manufacturing, financial services and government. As was the case in other parts of the world, most of this early software was developed in-house to meet specific requirements.

In the 1970's and 80's enforced racial segregation in South Africa, under the government of the day's Apartheid policies, met with a coordinated response from the international community. This resulted in South Africa being isolated economically and politically. Trade sanctions were imposed and multinational corporations disinvested from the South African economy. Many of the formal commercial links between South African and international companies in the Information Technology (IT) sector were cut. Companies like Unisys, IBM, DEC and others withdrew from the South African market, while newer companies (such as Apple) decided not to enter.

The South African IT sector, however, responded enthusiastically to international sanctions! Software developers were given the green light to engage in the activity that software developers do best – they re-invented the wheel! "Reverse engineering" as a means of "sanctions busting" became the mode of operation for the South African software industry. Sanctions also meant that local developers weren't required to compete with their international peers in securing contracts from South African companies. Nor were they required to measure their quality and performance against international best practice standards.

In April 1994, after the negotiated end to Apartheid, Nelson Mandela became the first democratically elected President of the "new" South Africa. International sanctions ended and, after nearly 20 years of working in isolation, the South African ICT ("Information and Communication Technology)" sector suddenly found itself needing to integrate into the highly competitive international industry.

1.2 CHALLENGES FACED BY THE SOUTH AFRICAN ICT SECTOR

The ICT sector in South Africa has a number of strengths when compared to its international competitors. It is relatively large – valued at US$ 28 billion in 2009. It is diverse – covering a broad range of sectors and technologies. It is innovative – for example: South African IT entrepreneur Mark Shuttleworth developed the certificate-based internet security protocols that support modern e-commerce[1] transactions, and South African company, Vodacom, was the first in the world to pioneer the pre-payment concept in the cell phone industry.

The South African ICT sector also has a number of significant weaknesses. These include a lack of process maturity resulting in an inability to deliver software projects predictably and with high levels of quality. In an attempt to deal with these weaknesses, the "Joburg Centre for Software Engineering" (JCSE) located at the University of the Witwatersrand (or "Wits") in Johannesburg, embarked on a programme in 2006 aimed at promoting process improvement within the South African ICT sector. With support from the South African Government, through its Department of Trade and Industry (*the dti*), the JCSE became a transition partner of the Software Engineering Institute (SEI) at Carnegie Mellon University, USA. The SEI is a world leader in the field of process improvement, having developed frameworks like the Capability Maturity Model Integration (CMMI), and methodologies like the Team Software Process (TSP) and Personal Software Process (PSP). A programme was launched to "Bring CMMI to South Africa."

While the JCSE met with some success in persuading South African companies to consider CMMI adoption, a significant concern amongst these companies was the length of time and effort required in moving from one maturity level to the next. There clearly was a need to fast-track process improvement so that the benefits of higher maturity could be realised as soon as possible. With this in mind the JCSE began to investigate the adoption of TSP and PSP in South Africa.

In 2008 the JCSE organised a study tour to Mexico and the USA. The delegation, lead by the JCSE's Director, was accompanied by the SEI, and consisted of representatives from two companies, two universities and government. The aim was to observe at first hand organisations that had adopted TSP and to understand the lessons – positive and negative – that had been learnt.

[1] Shuttleworth's company Thawte developed technology that was acquired by Verisign in 1999.

1.3 REVIEW OF BENEFITS EXPECTED FROM TSP ADOPTION

From information gleaned on the Mexico/USA study tour in 2008, together with published data and information provided by the SEI, a very compelling argument emerged for TSP adoption. The case supporting TSP adoption is summarized in the following:

- Software development projects are completed within 10% or less of the scheduled date. The industry benchmark is in the range 27% to 112%
- The final cost of development projects is within 5% or less of the original budget. The industry benchmark is in the range 17% to 85%.
- Released code is likely to have 0 to 0.2 defects per 1,000 lines of source code. The industry benchmark is 1 to 7 defects per 1,000 lines of source code.
- System testing requires between 2% and 7% of the overall effort (time or cost) associated with a development project. The industry benchmark is typically 40%.
- Companies using TSP are able to accelerate CMMI adoption dramatically. Some organisations have progressed from maturity level 1 to 4 in less than 2 years. The normal time taken is 5 years or more.

There is a further advantage of TSP adoption that is less easy to quantify precisely: members of TSP teams and their managers really like using it! In meeting with TSP/PSP practitioners in both Mexico and the USA the delegation heard many positive comments about the way in which TSP teams worked. Many said that their work-life balance had improved significantly and that TSP helped them feel empowered. One senior manager said that adopting TSP had been the best decision he had ever made.

While the delegation was convinced that TSP adoption had certainly resulted in significant benefits at those companies visited, the question that remained was whether it would have similar positive results in South Africa. The JCSE decided to initiate a pilot TSP adoption programme in South Africa.

The pilot programme started in July 2009, and this paper is the first feedback on the results that have been achieved and the lessons that have been learnt.

1.4 THE PILOT TSP ADOPTION PROGRAMME IN SOUTH AFRICA

In planning the pilot TSP adoption programme in South Africa the JCSE identified three primary objectives. These were:

a) to gain experience in using TSP in software development projects that are sufficiently representative of the types of projects undertaken by South African development teams;

b) to assess whether the benefits derived from TSP adoption by organisations elsewhere in the world are likely to be experienced in South Africa;

c) to use the pilot programme to train the first group of PSP instructors and TSP coaches in South Africa.

The following were the role-players in the pilot TSP programme:

- <u>The Software Engineering Institute (SEI) at Carnegie Mellon University:</u> The SEI is the developer of the PSP and TSP methodologies. The SEI provided TSP and PSP training and coaching to the pilot companies. It also provided training to candidate South African PSP instructors and TSP coaches. The other key role for the SEI was to quality assure the pilot, i.e. to ensure that the PSP and TSP practices were applied correctly and completely.
- <u>The Joburg Centre for Software Engineering (JCSE) at Wits University:</u> The JCSE coordinated the TSP pilot. It appointed candidate coaches and instructors to be trained as part of the pilot. The objective was for the JCSE to have, by the end of the pilot, the capacity to train and launch new TSP teams at other companies. An additional role for the JCSE was to evaluate the pilot programme and to draw conclusions as to its success or otherwise.
- <u>The Department of Trade and Industry (the dti):</u> The "Electrotechnical Unit" of the dti is the entity within the South African government that is responsible for promoting the success, both locally and internationally, of the ICT sector. the dti secured funding to cover part of the costs required in running the TSP pilot. A research report evaluating the pilot will be written by the JCSE at the end of the pilot and presented to the dti so that it can be used in guiding government policies and programmes to support the ICT sector.
- <u>The Group Technology (GT) division of Nedbank:</u> Nedbank is one of South Africa's four major banks. The GT division is responsible for IT systems within the bank. It employs over 2,000 IT professionals and runs hundreds of software development and maintenance projects each year. Nedbank sent a representative on the study tour to Mexico and the USA. After receiving a report from the study tour Nedbank's CIO agreed to participate in the TSP pilot by running two development projects using TSP. Nedbank also nominated two candidate TSP coaches who would use the pilot projects to gain experience and receive coach training. Nedbank covered most of its own TSP training and coaching costs.
- <u>Dariel Solutions:</u> Dariel is a medium-sized software development company. It undertakes complex and innovative projects for clients in a number of sectors. Dariel agreed to run one development project as part of the TSP pilot. Like Nedbank, Dariel covered most of its own training and coaching costs. There were no provisions made in the pilot to train an internal coach for Dariel.

- The Central University of Technology (CUT): CUT is located in Bloemfontein, South Africa. The School of Information Technology graduates about 150 software developers per year. Many of these seek employment after graduating in the Johannesburg area. CUT joined the pilot to explore ways of incorporating PSP training into its curriculum. CUT's aim in the pilot was to train two faculty members as PSP instructors.

The TSP Pilot was launched towards the end of July 2009 once funding from *the dti* had been secured and Nedbank had selected 2 development projects. The SEI sent two members of its TSP team to South Africa (Bill Nichols and Jim McHale) to conduct training and launch both Nedbank projects. The following training courses were presented:

- TSP Executive overview (1-day)
- Leading Development Teams (3-days), and
- PSP Fundamentals (5-days)

After providing training and launching the two Nedbank teams in South Africa, the SEI provided remote coaching (via telephone). SEI coaches also visited periodically to conduct checkpoints, postmortems and relaunches. They also provided mentorship to candidate coaches.

In October 2009 the SEI presented the PSP Fundamentals course for the Dariel team and launched their TSP project. The "PSP Advanced" course was also presented by the SEI for the candidate coaches and instructors in December 2009, followed by TSP coach training. PSP Instructor training was presented in January 2010 in Johannesburg by the SEI. This was followed by PSP Fundamentals training for a third Nedbank team. This team's project was launched by one of the candidate Nedbank coaches, observed by an SEI coach.

Both of the initial Nedbank projects had their final post-mortem in May 2010. Candidate PSP instructors wrote their PSP Certification exams in May and June 2010 as their final step in completing their training.

1.5 RESULTS OF THE TSP PILOT

As will be seen from the description given above, there was a great deal of TSP-related activity in South Africa from July 2009 to May 2010. Members of the SEI's TSP team visited frequently to provide training, coaching and mentorship. Four teams gained experience working on TSP projects and candidate instructors and coaches notched up the pre-requisites for certification.

In this section we outline at a high level the outcomes of the pilot TSP adoption programme from the perspective of each of the key roleplayers.

a) JCSE at Wits University

In its role as coordinator of the pilot the JCSE's first objective was to ensure that the programme of activities ran as smoothly as possible. The JCSE also needed to position itself to be able to continue driving TSP adoption in South Africa beyond the pilot phase. The third objective of the JCSE was to carry out a detailed assessment of the pilot and to report its findings to Government (via *the dti*) and to the ICT sector in general.

The first of these objectives was certainly achieved. The pilot ran smoothly and all required activities were completed as planned.

In terms of the second objective, the pilot programme has resulted in 5 candidate TSP coaches and 5 candidate PSP instructors completing all pre-requisite training. By the end of May 2010 there were still a few outstanding requirements in terms of completion of certification exams and elements of TSP coaching experience. These should, however, be completed soon allowing all of these candidates to begin training and coaching independently.

This paper represents part of the JCSE's third objective, namely to report results of the TSP pilot. Various other reports and presentations have been, or soon will be, completed and delivered.

The pilot has therefore been a great success for the JCSE.

b) Nedbank

Like any large organisation that has its own internal capacity to run software development projects, Nedbank faces many challenges. There is a need to ensure that projects are completed on time and within budget, while ensuring that the software produced is of the highest possible quality. Nedbank management are also eager to foster motivated and cohesive teams – Fred Swanepoel, Nedbank's Chief Information Officer (CIO), is particularly focused on achieving a positive work-life balance for his IT staff.

With strong sponsorship from Fred Swanepoel, Nedbank joined the TSP pilot as a way of evaluating how TSP may support its ongoing efforts to change both the processes and culture of the organisation.

Two projects were initially selected for TSP pilot implementation. A third project was launched in February 2010. By June 2010, Nedbank management had reached the decision to initiate a roll-out of TSP into Nedbank's Group Technology (GT) division. This decision was based on management's conclusion that the pilot was a success. Why did Nedbank management consider the pilot to be successful?

There were a number of reasons:

- Availability and quality of data: For the first time ever, GT projects were able to collect detailed and accurate data at an individual developer level that, when "rolled up" to a project level, helped in understanding the following: time spent on specific tasks; defect injection and removal rates from detailed design onwards; earned value; and process efficiency. Data was also being collected to support estimation of specific tasks and measuring the effectiveness of code reviews. A Benford analysis of time data from the pilot teams showed a high level of data accuracy (estimated to be better than 90%).

- Improved performance: The first two pilot teams significantly exceeded the 10% targets for improved time on project tasks and earned value performance by a factor of 3 or more. Figure 1 shows the task hour and earned value performance over time for one of the pilot projects. This team began with a substantial variance from plan but achieved the planned level through increased task hours and consistent re-planning.
- Improved quality: One of the pilot projects recorded a dramatic reduction in defects detected in system testing and after deployment. The first release had a small number of defects found in production and none in system test. One of these production defects was traced to a configuration problem. In later releases, no further production defects were found.. Indications are that similar quality improvements would be achieved in Nedbank's other pilot projects, although detailed system testing has not yet been done. Measured defect densities in one of the pilot projects were as follows: code review - 5 defects /KLOC; unit test – 5 defects /KLOC; inspection - 7 Defects /KLOC. Sixty percent of modules were free of defects in unit test. This is surprisingly high considering the size of the modules. Figure 2 shows the scatter plot of size versus defect counts for modules developed in this project. The graph shows a 2.5 defects/KLOC trend line.
- Estimation and planning: There was a significant improvement in the ability of the pilot teams to estimate effort, to plan and to track progress. There was also a reduction in the amount of re-work done.
- Team dynamics and positive response of team members: As would be expected, when the pilot started in July 2009, many staff at Nedbank, both those directly involved and others, expressed misgivings about TSP and its adoption at Nedbank. By June 2010 these doubts had disappeared. Many participants in the pilot were interviewed by researchers from the JCSE, and there was a unanimous view from team members that they had gained another dimension in which to think and plan the work of a project, and to perform better. Everyone had found the experience positive and was keen to continue using TSP and learning more about it. TSP's emphasis on developing self-directed teams also gave developers an opportunity to grow professionally. One of the project teams expressed disappointment at the prospect of the team members being redeployed to other (non-TSP) projects once the TSP pilot was completed. We therefore conclude that the process succeeded in building cohesive jelled teams.

TSP adoption in Nedbank was challenged by implementation and organizational issues, nonetheless, the teams managed to demonstrate success in both of the completed projects. The project teams developed plans, established planning baseline data, and improved development processes. Repeated use of TSP over several iterations on one of the projects demonstrated the ability of the team to improve process adherence, improve their planning, and improve the delivered quality. Towards the end of the TSP pilot in June 2010, the pull to implement TSP in the wider Group Technology (GT) Division has been significant. The sponsor, management, and other project teams are extremely eager to implement TSP throughout Nedbank GT. An aggressive rollout plan has been approved.

c) Dariel Solutions

The pilot project at Dariel Solutions started late in 2009 and has been finding it challenging to apply TSP correctly and collect useful data. The following have been some of the challenges:

- The project selected by Dariel is a new development using a technology stack and architecture that is relative new to the developers and the company. This has meant that software development has been done in combination with a great deal of research and investigation, making data collection and strict adherence to process very difficult;
- As a relatively small and busy company it has been difficult to retain all of the team members on the project since some have been needed to join other teams. This has led to a situation where only two of the original team members are still part of the 6-person team;
- Coaching has been difficult. In the case of Nedbank TSP coaching was done by internal candidate coaches with an SEI coach regularly available via a teleconference link. Dariel also had to rely on a remote SEI coach, but had no internal candidate coach. The JCSE's candidate coach was available but not as frequently and spontaneously as Dariel required.
- The lack of immediacy in coaching has meant that it has been hard for this new team to "own the process," and feel entitled to adapt it as required. The team initially tried to implement the PSP process as it was taught in the training course. Some aspects of this process were inappropriate and the team lacked the confidence and experience to adapt it. This created significant difficulties.

All of the above factors have made understanding TSP, following process and collecting data extremely difficult. Various corrective actions have been recently put in place. The team have defined a less ambitious implementation of TSP and PSP, which will allow for learning and constant improvement. As a target this is far more achievable and is already showing real value to the organisation.

In spite of the difficulties, developers and management at Dariel remain committed to the TSP pilot, and are keen to begin training a second team.

1.6 CONCLUSIONS

This report on the South African TSP adoption pilot has presented largely a qualitative account of the objectives and some of the outcomes. In many ways the pilot has succeeded in growing experience, training local TSP expertise and understanding more clearly some of the "non-TSP" challenges associated with TSP adoption.

As the adoption of TSP grows in South Africa the JCSE will be collecting and analysing quantitative data that will lead to a more objective data-driven assessment of the ongoing TSP pilot.

All role-players are, however, unanimous in believing that TSP and PSP are destined to shape the future of software development in South Africa.

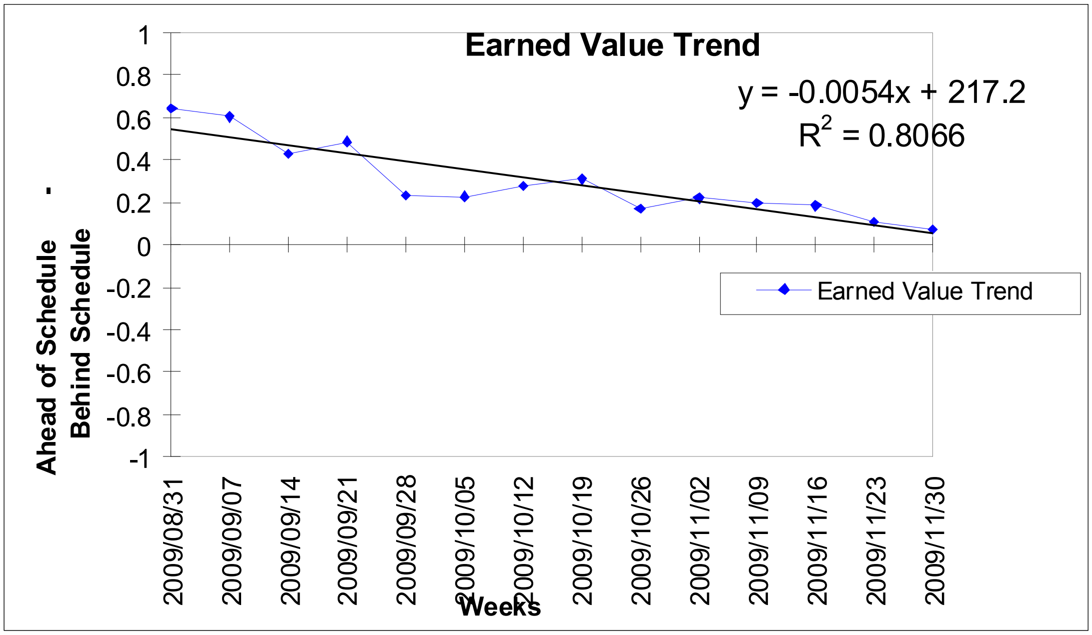


Figure 1: Earned Value Trend for one of Nedbank's Pilot Projects

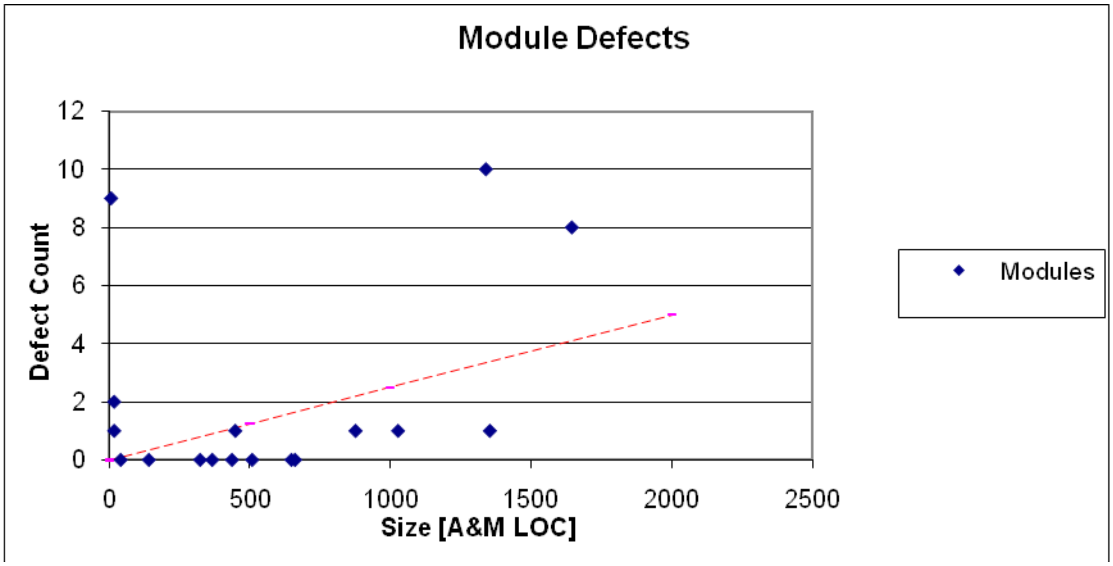


Figure 2: Defect Counts per module versus size for one of Nedbank's Pilot Projects

BIOGRAPHY

Barry Dwolatzky

Barry is Professor of Software Engineering at the University of the Witwatersrand, or Wits, in Johannesburg, South Africa. He is also Director and CEO of the Joburg Centre for Software Engineering (JCSE) at Wits.

Barry graduated as an Electrical Engineer from Wits in 1975. He stayed on to complete his PhD in 1979. Between 1980 and 1989 he lived in the United Kingdom working as a post-doctoral research associate at the University of Manchester Institute of Science and Technology (UMIST) and Imperial College, London. He joined the company GEC-Marconi as a researcher in 1985 and left in 1989 to return to South Africa. Since returning to Wits he has been on the staff of the School of Electrical and Information Engineering.

Barry has published extensively in academic journals, has presented at conferences and has successfully supervised 5 PhD's and over 30 MSc research students. His major current interest is in promoting the growth and competitiveness of the South African and African software development sector. He believes that this can be achieved by promoting the adoption of sound software engineering principles and practices.

He is chairman of the South African Chapter of the IEEE-CS. He is also a certified CMMI and PSP Instructor.

Tamasin Bossert

Tamasin is a Business Improvement Executive in the Nedbank banking group, accountable for a spectrum of initiatives encompassing Software and Systems engineering process domains. Tamasin has led and managed a multitude of key strategic initiatives for the bank. Tamasin's experience ranges across various domains from Process Analysis, Strategic Analysis, Governance, Software and Systems Lifecycle Management, Process Re-engineering, Project & Portfolio Management, and Change Management through Senior Executive Management equivalent roles. Tamasin has helped various senior management teams to develop and implement cross-functional process improvement opportunities.

Tamasin holds a specialisation in Commerce and her knowledge expertise cuts across frameworks like the Capability Maturity Model Integrated (CMMI), IDEAL, ISO 9001, PMBOK, Business Process Maturity Model (BPMM), Six Sigma, ITIL, PMBOK etc amongst others.

Alok Goswami

Alok Goswami is a Process Portfolio Manager at Nedbank, South Africa. Alok's forte is in Process Excellence and Initiatives based on international best practice frameworks. A Business Management graduate (MBA), Alok holds a spectrum of accredited and certified qualifications in various frameworks and methodologies some of which include being an SEI authorized SCAMPI B&C Team Leader (CMMI), Instructor for CMMI (Dev), EXIN certified ITIL Service Manager, Juran certified Six Sigma Green Belt, Internal Auditor for ISO 9001-2000, PMI certified PMP etc amongst others.

Alok has been a part of the core teams responsible for implementing TSP, CMMI, Six Sigma and PCMM frameworks. He has also been a trainer for various programmes in Quality Management Frameworks. As a certified Coach, Alok has facilitated development of effective leadership skills in the candidates being groomed for leadership positions at Nedbank. Alok has also published and presented papers on Process Engineering at various conferences.

Lisa Lyhne

Lisa Lyhne is the Operations Executive Director at Dariel Solutions, which is a medium-sized software development company focused on bespoke developments for corporate customers in Java and .Net. She has a passion for process improvement and software development methodologies. She has introduced CMMi and subsequently TSP to the company. She has over 20 years experience in the software development industry, working in South Africa, the UK and in Denmark. She has worked in many roles, including technical and management roles, for a variety of clients and in a variety of industries, including banking and financial, resources, medical and several others.

Lisa holds a bachelor's degree in Commerce from the University of Cape Town.

Illuminating the Intersection of TSP and CMMI High Maturity Process Performance Models

Robert W. Stoddard, *Software Engineering Institute*
Shigeru Sasao, *Carnegie Mellon University*
Dave Webb, *U.S. Air Force*
Jim VanBuren, *U.S. Air Force*

1.1 INTRODUCTION

This paper will illuminate the intersection of the Team Software Process methodology and the CMMI High Maturity concepts of process performance baselines and models. Experience with both CMMI High Maturity organizations and TSP projects have enabled this discussion to progress to the point where a body of knowledge related to the intersection is now possible. An earlier publication by the Software Engineering Institute (SEI) Software Engineering Measurement and Analysis (SEMA) team [Tamura 09] began this discussion with high level concepts of process performance models in a TSP environment. This paper and associated presentation ventures much deeper into the topic by discussing the detailed leading indicators found in the TSP body of knowledge related to leading TSP teams [Humphrey 06a]. This paper includes a very brief conceptual overview of both: 1) the TSP measures and models, and 2) the concept of CMMI Process Performance models, to set the context disregarding the reader's historical orientation. As such, it is hoped this paper and presentation will serve to motivate a number of pilots of process performance modeling within the TSP domain, as well as, pilots within CMMI High Maturity organizations not currently implementing TSP. Apart from motivating these varied pilots, the authors also hope to record the adoption and deployment experience of the pilots as a means to test the hypothesis that adoption of TSP does in fact accelerate accomplishment of CMMI High Maturity and does in fact produce superior results to CMMI High Maturity implementations without TSP.

The crux of this paper will be the discussion of the possible outcomes or performance measures that would be reasonably valuable in planning, tracking, and controlling software projects and software teams. This discussion will also include the space of interim outcomes, in addition to final project outcomes, available in TSP implementations. With the outcomes fully defined, the subsequent discussion will be on the potential "x" factors that may be logically inferred within the TSP implementation and which may serve as significant factors in predicting the different outcomes discussed. At this point, the reader will realize that much of the published work on leading and coaching TSP teams inherently identifies a rich set of factors which can be operationally defined as measures to participate in process performance modeling, disregarding whether the modeling is statistical, probabilistic, or simulation in nature. A fuller treatment of this subject with detailed models, example data sets, scenario usage of the process performance models, etc… will follow in a future SEI Technical Report.

1.2 TSP MEASURES, QUALITY PROFILES, AND PREDICTION MODELING

The Team Software Process (TSP) is a software engineering process designed to enable engineering teams to build software-intensive products more predictably and effectively [Humphrey 10]. It provides a framework for teams to plan and track their work, and to tailor their processes for continuous improvement. The TSP provides a rich set of forms and scripts, generally in a form of a software tool, to allow teams to gather data about their project. A large portion of the data is populated during the project planning session called the TSP launch, and the remainder of the data is collected throughout the project.

The TSP launch is conducted during the first week of a project. The TSP launch is a series of meetings where team members elicit the management's goals for the project, generate a plan reflecting those goals, and receives agreement and go-ahead from the management. The following is a summary of the data collected during the TSP launch:

1. Basic project information such as project name, team name and start date
2. Team members, contact information, and their roles
3. Management and team goals, along with risks associated with those goals, classified by their impact and likelihood
4. Output products (e.g. software artifacts, documentation), along with estimated sizes and estimated number of defects injected
5. Estimated schedule information and available resources (hours per week)
6. A task list with estimated hours per task along with measures required for Earned Value Analysis (EVA)
7. Quality goals and possible defect types

After agreement is made over the plan, the team members begin to execute the plan. During this period, each team member collects data about their work. The base measures collected in TSP can be largely divided into two areas: planning and tracking, and quality. For planning and tracking, each team member records in a time log the following information for each task that is being executed:

1. Start and stop time
2. Interrupt time
3. Delta time
4. Task completion date

Each task in TSP is associated with an output product and a phase. Therefore, derived measures such as time in phase can be calculated by simply collecting the above data. Also, the data collected in the time log drives the Earned Value Analysis used to track the project's schedule performance.

In terms of quality, the team members record the following information into the defect log:

1. Date recorded
2. Defect ID
3. Output product
4. Type of defect
5. Injection and removal phase
6. Fix time
7. Fix reference, in case a defect is introduced while fixing another defect
8. Description

In addition, the actual sizes of the output products are recorded as they are completed. Using the data from the time log, defect log and size, various derived measures can be calculated for the project. Some of the important measures include:

1. Process yield (the efficiency with which defects are removed from products)
2. Defect injection and removal per phase
3. Defect density (Defects per KLOC, defects per page)
4. Time in phase

Teams review their data during weekly team meetings and postmortem sessions to analyze how their process can be improved. For example, if many of the defects are being found in system test and the time in phase and/or yield is low for reviews, team members can increase the time spent in reviews and/or the review yield to try to catch defects at earlier stages. Thus, the data used in TSP are most often used as lagging indicators, compared to leading indicators for predictive modeling.

One example of predictive modeling in TSP is the PROxy-Based Estimating (PROBE) method. PROBE uses a linear regression model with size as the independent variable to estimate the effort of developing an output product [Humphrey 95].

1.3 CMMI PROCESS PERFORMANCE BASELINES AND MODELS

Discussion of the "healthy ingredients" of CMMI process performance models began in 2007 with SEI presentations at SEPG conferences and was amplified in the SEI course "Understanding CMMI High Maturity Practices (UCHMP)." The healthy ingredients were first elaborated dynamically, during the conduct of SEI measurement courses in 2006 and 2007, as a means of communicating what process performance models were in concrete, practical terms. The ingredients are derived from a holistic understanding of the intent of the CMMI models. The precise nature of several of the ingredients also comes from training, experience, and practice within the Six Sigma arena. The healthy ingredients of process performance models are briefly summarized below for the benefit of the reader, as published in an earlier technical report by the SEI Software Engineering Measurement and Analysis Team [Stoddard, Goldenson 09].

1.3.1 THE MODEL IS STATISTICAL, PROBABILISTIC, OR SIMULATION-BASED.

This particular ingredient emphasizes the logical consistency of two CMMI process areas: Quantitative Project Management (QPM) and Organizational Process Performance (OPP). QPM stresses the need for understanding statistical variation of process performance factors. Additionally, QPM reinforces the need to separate assignable, special cause variation from inherent common cause variation to help understand what actions to take with respect to each type of variation. This healthy ingredient emphasizes the need for process performance models to model the uncertainty of the predictive factors and their resulting impact on the uncertainty of the behavior of the outcome factor. For this reason, deterministic models that merely perform mathematical calculations on point estimates fall short of the superior information achievable from models that are statistical, probabilistic, or simulation in nature.

1.3.2 THE MODEL PREDICTS INTERIM AND/OR FINAL PROJECT OUTCOMES.

This ingredient derives more from practical experience and management's need for real-time cycles of learning within a given project or program. To maximize real-time cycles of learning within a given project or program, managers need to predict interim performance outcomes in addition to the traditional end-of-project performance outcomes.

1.3.3 THE MODEL USES CONTROLLABLE PREDICTIVE FACTORS THAT ARE DIRECTLY TIED TO SUBPROCESSES OR WORK ACTIVITIES.

This healthy ingredient focuses on the need for process performance models to be actionable. From that standpoint, if a model does not have at least one controllable predictive factor, it does not directly promote insight of action to influence the undesirable predicted outcome. This may be a fine nuance, but project forecasting models that model only uncontrollable factors make predictions that offer little help or insight into the actions to be taken to drive a more desirable predicted outcome. Additionally, this ingredient highlights the need for the controllable factors to be detailed enough to show a clear link to a specific subprocess or work activity. This clear link enables proactive management responses.

1.3.4 THE MODEL QUANTITATIVELY CHARACTERIZES AND MODELS THE VARIATION OF THE PREDICTIVE FACTORS AND DESCRIBES THE PREDICTED RANGE, UNCERTAINTY, OR VARIATION OF THE OUTCOME PERFORMANCE MEASURES.

This ingredient is a chief overlap of CMMI high maturity and Six Sigma concepts. Recognizing that variation (i.e., risk) may very well be unbalanced and significant in the real world, the models account for this by modeling the uncertainty of the predictive factors. Numerous examples exist in industry in which analysis using only the mean or average estimate rather than the distributional information caused serious problems in predictions of schedule, performance, and other modeled factors.

1.3.5 THE MODEL ENABLES "WHAT-IF" ANALYSIS FOR PROJECT PLANNING, DYNAMIC RE-PLANNING, AND PROBLEM RESOLUTION DURING PROJECT EXECUTION.

This ingredient builds on language in the CMMI Organizational Process Performance (OPP), Quantitative Project Management (QPM), Organizational Innovation and Deployment (OID), and Causal Analysis and Resolution (CAR) process areas related to the use of process performance models to support "what-if" and sensitivity analysis. The idea is that decision makers will be able to use process performance models to analyze alternative courses of action and alternative improvement ideas. Again, this highlights a capability intended to be exercised within a given project or program execution.

1.3.6 THE MODEL CONNECTS UPSTREAM ACTIVITY WITH DOWNSTREAM ACTIVITY.

This particular ingredient emphasizes the intent of process performance models to enable decision-makers to observe a prediction of the consequences of decisions made earlier in the life cycle or process. Indeed, this ingredient highlights the practical use of process performance models for transitions from phase to phase, hand-offs from one group to another, and so on. This particular ingredient enables the establishment and enforcement of interface agreements between internal groups and/or external groups by providing models that predict the readiness and maturity of an artifact or work product to proceed to the next step. For example, many organizations employ such models to predict defects entering system test while the code is still with the development team. Others use models to predict readiness of design or code to enter an inspection. Still other organizations use models in this fashion to determine if product and software re-quirements are sufficiently mature and stable to begin intense development.

1.3.7 THE MODEL ENABLES PROJECTS TO ACHIEVE MID-COURSE CORRECTIONS TO ENSURE PROJECT SUCCESS.

This ingredient highlights a very significant aspect that may be read into the usage of process performance models in CMMI. Specifically, within the QPM process area, process performance models may be used to anticipate undesirable performance with enough lead time to proactively influence the situation toward a successful outcome. Industry experience with this aspect is quite strong, especially in the use of critical parameter management in the Design-for-Six Sigma (DFSS) community. The notion is that models of critical parameters of the product design foster early insight into issues in products and processes enabling management to take corrective and preventive action. For this reason, organizations employ a collection of process performance models to cover their needs throughout the project life cycle.

1.4 POTENTIAL INTERIM AND FINAL PERFORMANCE OUTCOME MEASURES TO BE PREDICTED

Identifying the performance outcome measures to be predicted remains a critical first step, and often the most challenging step, in process performance modeling. Projects need to begin with a complete understanding of the organizational goals and objectives, in addition to the goals and objectives at the project level which are often heavily influenced by specific project stakeholder and customer needs. Essentially, the performance outcome measures must be aligned and traceable to higher level goals and objectives to ensure that the subsequent process performance models focus on predicting high-value performance outcomes. Generally, performance outcomes related to cost, schedule, and quality rank high in the list of outcomes to predict. Additionally, other outcome measures may be important to predict including: productivity, customer satisfaction, revenue, market share, customer loyalty, brand image, etc. Figure 1 depicts many industry-proven outcome measures at the project and/or organizational level.

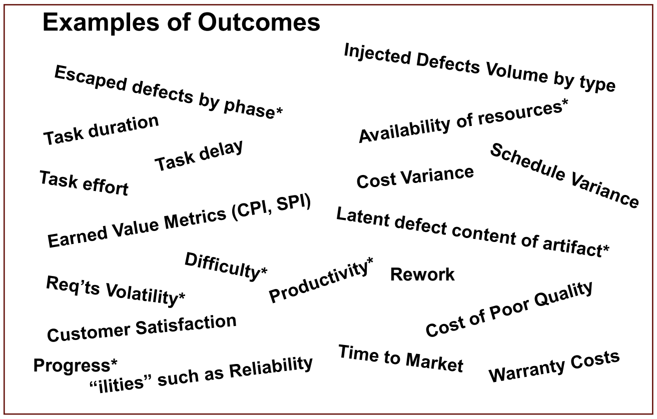


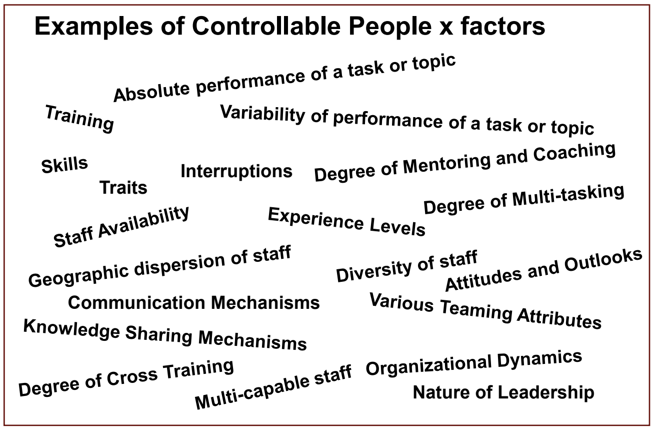
Figure 1: Examples of Outcome Measures

Many of these outcome measures may also be defined in more granular terms and for use during the development lifecycle. For example, quality may be further defined to be a measure of the number of defects by type entering system test. In this example, such models provide the system test group with invaluable information prior to the commencement of system test, thereby informing the nature and sequence of testing, as well as, an expectation of the needed duration and effort of system testing. System test groups may also use such a prediction model as a screen to determine if the software is in sufficient condition to enter system test. As such, the prediction model becomes a handshake contract between the software developers and system testers, so that software is not prematurely cast into the system test cycle.

As such, outcome measures for TSP software teams could include any predicted information items that would be useful to the TSP team, the TSP leader or the TSP coach during the conduct of the software development lifecycle in support of project monitoring, controlling, analyzing, correcting, preventing and reporting.

1.5 LEADING INDICATORS AND OTHER CAUSAL FACTORS PREDICTING OUTCOMES

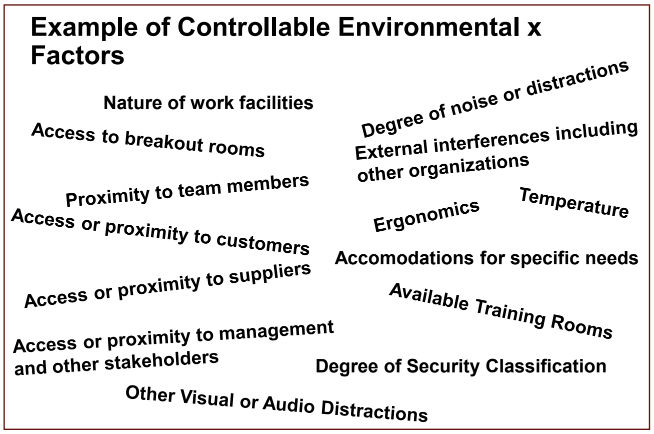
The crux of this paper rests with the rich ideas of leading indicators that arise from reading Watts Humphrey's books on leading and coaching TSP teams. [Humphrey 06a] [Humphrey 06b] Conference presentations and training by the SEI measurement team have outlined sets of potential leading indicator measures that have served well in process performance models at both the project and organizational level. These measures may be seen in the following 6 figures, which are slide extractions from previous SEI CMMI High Maturity presentations.



Examples of Controllable People x factors

Absolute performance of a task or topic
Training
Variability of performance of a task or topic
Skills
Traits
Interruptions
Degree of Mentoring and Coaching
Degree of Multi-tasking
Staff Availability
Experience Levels
Geographic dispersion of staff
Diversity of staff
Attitudes and Outlooks
Communication Mechanisms
Various Teaming Attributes
Knowledge Sharing Mechanisms
Degree of Cross Training
Multi-capable staff
Organizational Dynamics
Nature of Leadership

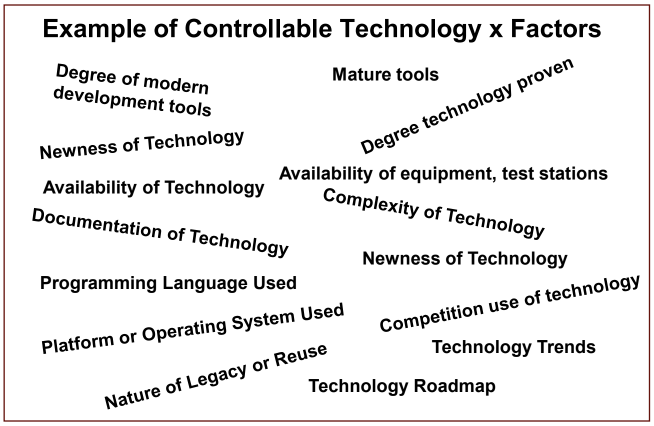
Figure 2: Examples of Controllable People Factors



Example of Controllable Environmental x Factors

Nature of work facilities
Degree of noise or distractions
Access to breakout rooms
External interferences including other organizations
Proximity to team members
Temperature
Access or proximity to customers
Ergonomics
Accomodations for specific needs
Access or proximity to suppliers
Available Training Rooms
Access or proximity to management and other stakeholders
Degree of Security Classification
Other Visual or Audio Distractions

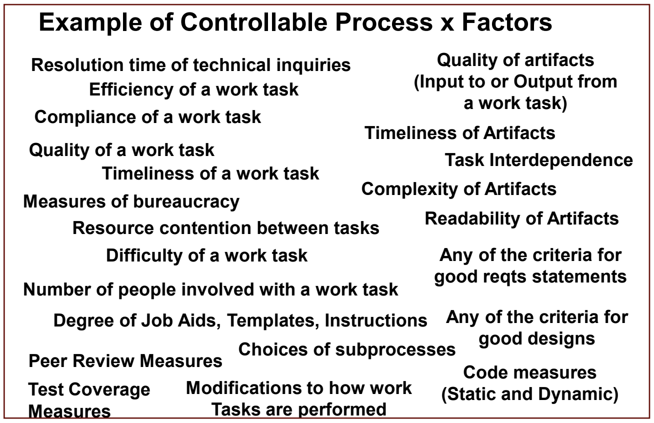
Figure 3: Examples of Controllable Environmental Factors



Example of Controllable Technology x Factors

Degree of modern development tools
Mature tools
Degree technology proven
Newness of Technology
Availability of Technology
Availability of equipment, test stations
Complexity of Technology
Documentation of Technology
Newness of Technology
Programming Language Used
Competition use of technology
Platform or Operating System Used
Technology Trends
Nature of Legacy or Reuse
Technology Roadmap

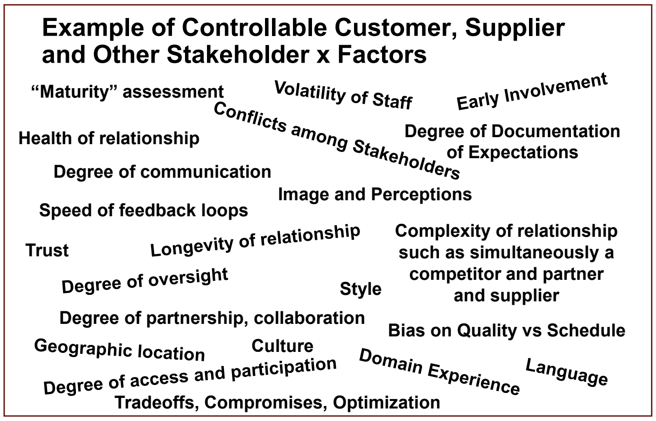
Figure 4: Examples of Controllable Technology Factors



Example of Controllable Process x Factors

Resolution time of technical inquiries
Efficiency of a work task
Quality of artifacts (Input to or Output from a work task)
Compliance of a work task
Timeliness of Artifacts
Quality of a work task
Timeliness of a work task
Task Interdependence
Measures of bureaucracy
Complexity of Artifacts
Resource contention between tasks
Readability of Artifacts
Difficulty of a work task
Any of the criteria for good reqts statements
Number of people involved with a work task
Degree of Job Aids, Templates, Instructions
Any of the criteria for good designs
Choices of subprocesses
Peer Review Measures
Code measures (Static and Dynamic)
Test Coverage Measures
Modifications to how work Tasks are performed

Figure 5: Examples of Controllable Process Factors



Example of Controllable Customer, Supplier and Other Stakeholder x Factors

"Maturity" assessment
Volatility of Staff
Early Involvement
Conflicts among Stakeholders
Health of relationship
Degree of Documentation of Expectations
Degree of communication
Image and Perceptions
Speed of feedback loops
Trust
Longevity of relationship
Complexity of relationship such as simultaneously a competitor and partner and supplier
Degree of oversight
Style
Degree of partnership, collaboration
Bias on Quality vs Schedule
Geographic location
Culture
Domain Experience
Language
Degree of access and participation
Tradeoffs, Compromises, Optimization

Figure 6: Examples of Controllable Customer, Supplier and Other Stakeholder Factors

However, in addition to the above potential leading indicators, the authors believe a rich set of untapped leading indicators may be derived directly from the two books written by Watts Humphrey on leading and coaching TSP teams [Humphrey 06a] [Humphrey 06b]. The following table summarizes these categories of leading indicators as referenced by corresponding chapters in the "TSP Leading a Development Team" book [Humphrey 06a].

TSP Topic	"TSP Leading a Development Team" Reference Chapter	Rationale for the Leading Indicators
Leadership	Chapter 2	The attributes of the leadership of TSP teams and of organizations possessing TSP teams may be significantly predictive of the performance of the TSP teams.
Team Attributes	Chapter 3	The attributes and nature of self-directed teams may also correlate and provide leading insight to the performance of TSP teams.
Team Motivation	Chapter 4	The specific aspects and components of motivation of individuals and teams also may correlate and provide leading insight to the performance of TSP teams.
Building Teams	Chapters 5-7	The manner in which software teams are formed and launched may also correlate and provide leading insight to the performance of TSP teams.
Teamworking	Chapters 8-11	The manner in which software teams manage their work to the plan, deal with changing requirements, track progress, overcome obstacles, follow their processes and manage quality, collectively, may correlate and provide leading insight to the performance of TSP teams.
Relating to Management	Chapters 12-14	The degree and character of management support for software teams, as well as the nature and health of the relationship of the software teams to upper management, may also correlate and provide leading insight to the performance of TSP teams.
Maintaining the Team	Chapters 15-18	The specific aspects and attributes associated with developing and coaching software teams and individuals may also correlate and provide leading insight to the performance of TSP teams.

Table 1: Categories of TSP Leading Indicators

The above 7 categories of potential leading indicator measures are now separately defined in the following 7 tables in which specific example candidate leading indicator measures are identified, along with one or more potential approaches to operationally measuring each candidate. At this time, the column depicting the outcome measure(s) to be predicted is intentionally left blank to depict the work in progress. On-going work with TSP teams creating process performance models with these leading indicators will help identify the beneficial outcomes to be predicted. It is hoped that the reader will now grasp the rich ideas for leading indicators arising out of the body of knowledge related to the TSP, and would be able to hypothesize which of the leading indicators, listed here, might be most promising and measurable in their own software team environments.

Table 2 below discusses ideas for additional leading indicators for the Leadership category. Although this table reflects the operational measures at a conceptual level, the next step would be to more fully populate a template such as the SEI Indicator template and/or a TSP script to support the necessary detail for consistent and repeatable implementation.

Candidate Leading Indicators	Potential Operational Measures
Effective and timely decision-making	Number of missed or late decisions; Impact of missed or late decisions
Leadership Vision	Vision articulated and communicated; Percentage of team unsure of vision; Percentage of stakeholders unsure of vision
Setting Direction via Goals	Goals clearly articulated and communicated; Percentage of team unsure of goals; Percentage of stakeholders unsure of goals
Leadership Motivation	Survey result of team members motivation by the team leader; Team Leader's self assessment of success of motivating team
Leadership Personal Commitment & Enthusiasm	Survey result of team members and stakeholders assessing the team leader personal commitment and enthusiasm; Survey and/or interview results of senior management assessment of the team leader personal commitment and enthusiasm
Leadership Taking Charge	Degree of well-organized and well-run team meetings; Degree of team crises embraced immediately by the team leader; Survey results of perceived leadership "take-charge" attribute
Leadership Leveraging Expertise within their team	Number of missed opportunities by the team leader to leverage expertise within the team; Degree to which team members perceive their expertise is not leveraged

Table 2: Factors related to Leadership

The next table, Table 3, discusses ideas for leading indicators related to the attributes of the TSP team itself. Depending on the situation and specific TSP team, different subsets of these attributes may be more indicative of team performance outcomes.

Candidate Leading Indicators	Potential Operational Measures	Potential Outcomes to be Predicted (work in progress)
Common Goal	Degree to which all team members understand and can state the team common goal	
Well defined team member roles	Degree to which team members perceive team member roles are well defined; Number of issues occurring due to a lack of well defined team member roles	
Team trust and cohesion	Survey results of individual team members with regards to team trust and cohesion; Number of issues arising from insufficient team trust and cohesion	
Sense of membership	Degree of positive feelings of team members regarding their team membership; Team member attrition initiated by the individual team member	
Ownership of the process and plan	Survey results of team members evaluating their ownership of the process and plan; Number of team member unresolved issues voiced about the team process and plan; Degree to which team members feel free to voice dissent regarding the process and plan	
Skill to make a plan	Number and percentage of team members skilled at making a team plan; A quantified total team experience level in years at making team plans	
Discipline to follow the plan	Number of instances in which team members do not follow the plan; Degree to which team members exert peer pressure on other members to follow the plan	
Dedication to excellence	Degree to which team members overtly subscribe to a dedication to excellence; Degree to which team members can quantify their personal improvement in the past 6-12 months	
Team member training	Degree to which team member skills do not match their work assignments; Number of days of professional development achieved by team members during a given year	

Table 3: Factors related to Team Attributes

The next table, Table 4, discusses ideas for leading indicators related to the motivation within the team. Again, depending on the situation and nature of the specific TSP team, different subsets of these leading indicators will be significant.

Candidate Leading Indicators	**Potential Operational Measures**	**Potential Outcomes to be Predicted (work in progress)**
Team placement on Maslow's Hierarchy of Needs	Position to which team members, the team leader and the team coach place the team on Maslow's hierarchy of needs; Degree to which non-self-fulfillment activity occupies the team member's focus, energy and time	
Cognitive Dissonance	Survey result of team members; Evaluation results of external team coach	
Feedback provided to team members	Survey result of team members; Number of improvement actions initiated by team member feedback	
Fear and greed vs commitment, as motivation	Team member self evaluation via survey; team leader independent assessment; external coach assessment	
Degree of negotiation within team	Team member self evaluation via survey; team leader independent assessment; external coach assessment; degree of time to reach team consensus; team members' attitudes toward negotiation	
Degree of Agreement within team	Team member self evaluation via survey; team leader independent assessment; external coach assessment; Degree of issues resulting from a lack of team agreement	
Degree of Performance within the team	Various objective measures of performance to include quality, schedule, budget, 360 degree evaluations	
Voluntary team member commitment	Degree to which open discussion occurs leading up to commitment; body language as assessed by team leaders and coaches	
Visible team member commitment	Pro-active actions by team members exhibiting individual commitment; Degree to which team members help build commitment in each other	
Credible team member commitments	Team member self evaluation via survey; team leader independent assessment; external coach assessment	
Individual team member ownership of the plan	Degree to which team members exhibit ownership of the plan; Degree to which team members communicate and sell the plan to other stakeholders	
Convert milestones into inchstones	Number or percentage of milestones that are planned with predecessor inchstones	
Identify steps for each inchstone	Number or percentage of inchstones planned with further detail of steps of work	
Regularly review team's progress against plan	Frequency of team progress reviews; Actions recorded by analyzing progress to plan; Survey of team members indicating satisfaction of frequency of reviews	
Provide regular feedback on inchstones	Frequency of inchstone reviews; Actions recorded by analyzing progress to plan of inchstones; Survey of team members indicating satisfaction of frequency of reviews of inchstones	
Take action to keep team perception that commitment is achievable	Coach evaluation of team leaders actions on this; Team leader self assessment of this; Survey results of team members satisfaction of team leader's actions to convince them the commitment is achievable; Number of times that the team perceives the commitment is not achievable	

Table 4: Factors related to Team Motivation

The next table, Table 5, consists of a lengthy list of ideas for leading indicators of team performance related to aspects of the process and effectiveness of the team building. Abundant literature from the self-directed and self-managed domains exists substantiating that this category seems to have the greatest potential for leading indicators of team performance. Consequently, organizations implementing self-directed or self-managed teams invest significant time and resources into team building activities, not only during initial team formations but on a continuous basis to sustain team operations.

Candidate Leading Indicators	Potential Operational Measures	Potential Outcomes to be Predicted (work in progress)
Secure Management Agreement to needed resources	Degree of management agreement to needed resources perceived by the team leader; by the coach; by other stakeholders	
Identify technical skills needed on team (Application domain, Product technology, Tools and Methods)	Degree to which the necessary skills for the project are identified in advance; Number of times the team finds itself short-handed from a skills standpoint; Impact of skills gap with the project needs in terms of budget, schedule, quality, etc…	
Identify teamwork skills needed on team (Estimating and Planning, Quality Management, Interpersonal Behavior)	Degree to which the team leader assesses the teamwork skills needed on the team (percentage from a standard list of skills); Number of teamwork shills identified as a source of problems later in the lifecycle	
Recruitment of team members with necessary skills	Degree to which recruitment of new team members is based on a skills checklist; Team member perceptions of skills match of new recruits; Recruit's reflections of their knowledge of the needed skills for the open position	
Performance of Launch step 1: Establish Product and Business Goals	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	
Performance of Launch step 2: Assign Roles and Define Team Goals	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	
Performance of Launch step 3: Produce Development Strategy	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	
Performance of Launch step 4: Build Overall and Next Phase Plans	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	
Performance of Launch step 5: Develop the Quality Plan	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	
Performance of Launch step 6: Build Detailed and Consolidated Plans	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	
Performance of Launch step 7: Conduct Risk Assessment	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	
Performance of Launch step 8: Prepare Management Briefing and Launch Report	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	
Performance of Launch step 9: Hold Management Review	Process compliance checklist; Survey participants and stakeholders for evaluation of the step; Number of actions arising from the launch step; Number of outstanding actions from previous launch steps when conducting this launch step; Coach evaluation of launch step	

Candidate Leading Indicators	Potential Operational Measures	Potential Outcomes to be Predicted (work in progress)
Performance of the Launch Postmortem	Timeliness of the launch postmortem; Participation in the postmortem; number of issues and actions identified in the postmortem; degree to which lessons learned are ignored and re-experienced; impacts of not adhering to previous lessons learned	
Team Performance of Data Gathering	Number or percentage of data gathering issues; degree of data quality issues; timeliness of data gathering within the team	
Team Performance of Plan Tracking	Degree to which the plan is tracked against actual team performance; Survey results of satisfaction of team members and team leader with respect to this	
Team Performance of Team Feedback	Frequency and quality of feedback provided to the team from external stakeholders; frequency and quality of feedback provided among team members	
Team Performance of Load Balancing	Degree to which load balancing occurs or re-occurs; Number of team member complaints about load balancing issues; Degree to which load imbalances cause issues (qty and impact)	
Team Performance of Replanning	Degree to which replanning occurs; Time since the last replan; time since the last request for a replan by a team member or stakeholder	
Team members trained in PSP	Number or percentage trained in PSP; Number of years PSP experience within the team as a whole	
Quality of the Team Member Selection Process	Degree to which existing team members participated in the team member selection process; degree to which the selection process was objective; degree to which a large net was cast in search of new team members; stability and longevity of team members once selected; new team member reaction to the selection process	
Degree of trust built up when leader inherits a team	Degree to which the new leader builds trust with the team; Amount of face time a new leader has with the inherited team; Surveyed self assessments of trust from both the leader and the team members; Number of actions that exhibit trust	
Team Member Skills Assessed	Degree to which skill assessment or testing is used; Degree to which solid references of skill performance are researched	
Team Member Aptitudes Assessed	Degree to which team member aptitude is assessed; Degree to which solid references of individual aptitudes are researched	
Team Member Interests Assessed	Degree to which team member interests are assessed; Degree to which solid references of individual interests are researched	
Degree of cooperation among team members	Team leader and team member individual survey results of satisfaction of existing team member cooperation; number or percentage of time team member cooperation doesn't exist; number or percentage of issues caused by internal team cooperation issues	
Degree to which leadership develops team members	Degree of 1-1 face time between team leaders and team members; number of development actions communicated to team members from team leaders; degree to which team member development is funded and supported	
Degree to which team members are promoted and advanced	Degree to which team members are promoted or advanced as compared to filling with external candidates; time since last promotion or advancement of team members (individually and/or collectively); Degree of team member expression of dissatisfaction related to promotion or advancement; attrition rates related to this issue	
Degree to which team building exercises used when needed	Number and type of team building exercises used within a team; survey results from team members on their satisfaction of sufficient team building exercises; number of times that lack of teaming is brought up as an issue during the lifecycle	
Degree to which the team receives timely and effective coaching	Periodicity of feedback from a coach; quality of the feedback; corrective actions enabled from such feedback; team member assessment of the value of the coaching	

Table 5: Factors related to Building Teams

In the next table, Table 6, a number of ideas for leading indicators of team performance related to teamworking are listed. As opposed to the formation and building of teams, this category includes leading indicators from the on-going operation of the team.

Candidate Leading Indicators	Potential Operational Measures	Potential Outcomes to be Predicted (work in progress)
Process compliance under stress	Number of process steps sacrificed during the lifecycle; during times of stress; impact of such violations	
Dynamic planning when needed	Degree to which replanning is implemented when needed; Degree to which current plan is perceived by team members to be unrealistic due to lack of replanning	
Impact analysis for all req'ts changes	Number or percentage of req'ts changes that are not accompanied by an impact analysis; Number of times that project issues arise due to improperly handled req'ts changes	
Progressive elaboration of plans	Degree to which underplanning and overplanning are minimized in accordance with the principle of progressive elaboration; Number of points in which progressive elaboration occurs; the effort and time incurred with replanning due to lack of progressive elaboration	
Workload balancing within the team	The number of times that workload imbalances cause team disruption, conflict or poor team performance; time required to rebalance the team; resistance of team members to workload balancing; number of times the team takes the initiative to look at workload balancing	
Tracking team progress with EV and task hours	Degree to which EV and task hours are not used to track team progress; Degree to which lack of team progress information prevented timely team leader and/or management action to prevent undesirable outcomes	
Obtaining help for the team	Number of times or situations in which the team needs help; Number of times that help is acquired; Number of times that requested help is not provided	
Definition of Success by the team	Survey results of team members, leader and coach regarding the satisfactory definition of Success by the team	
Setting and Maintaining Priorities	Number or percentage of the time that priorities are not set; Number or percentage of the time that team members perceive that priorities are not established	
Establishing Short Term Goals	Number of Long term goals without corresponding short term goals; Degree to which team members, leader and coach do not perceive adequate short term goals in place	
Overcoming Obstacles	Number and percentage of documented obstacles encountered by the team and overcome	
Changing Direction	The number of times that the team leader worked to change direction of the team when needed; The number of times that the direction was changed unnecessarily	
Involving the Customer	The amount of customer involvement, via face time, meeting time, telecons, number of inquiries or consults, technical inquiries with the customer	
Process Fidelity (accuracy of following the process)	Using checklists, the number or percentage of items faithfully followed; Impact of negative outcomes resulting from process infidelity	
Handling Process Problems	Number and percentage of process problems successfully handled monthly or per lifecycle phase; age of open process problems; time to resolve a process problem	
Quality as top priority	Number and percentage of decisions in which quality was sacrificed or traded off; survey results of team member and leader perception of quality as top priority	
Measurement of quality	Number and percentage of time that quality measures are not collected	
Individual ownership of quality	Degree to which individuals on the team collect their own personal quality data and take action based on the analysis	
Team ownership of quality	Degree to which the team collects quality data and takes action based on the analysis	
Quality reviews planned	The number and percentage of quality reviews planned vs total possible; The degree of time planned for each team member to participate in quality reviews	
Design and Coding Standards Used	The degree the standards are trained, communicated, used, monitored and updated	
Quality reviews held	The number and percentage of quality reviews held; the number and type of actions resulting from the quality reviews	
Defect reviews of test results	The number, type and percentage of defects found in testing; The number and type of defects predicted to be latent in the code	
Quality analysis conducted	The frequency and completeness of quality analysis throughout the lifecycle; The number and percentage of effort hours expended on quality analysis; The number of actions resulting from the quality analysis	
Reporting of Quality Data	The degree and timeliness to which quality data and results are reported to stakeholders; The frequency of the reporting; The stakeholder feedback on the usefulness of the reporting	

Table 6: Factors related to Teamworking

In the next table, Table 7, a number of ideas for leading indicators of team performance related to the interface with management are presented. Even the best of self-directed teams may be negatively impacted by an unhealthy relationship with management. Management support and advocacy remains important, and as such, this relationship must be developed, nurtured and maintained. First-hand experience by the authors confirms that self-directed teams may often be more at risk and impacted by unhealthy relationships with management than other organizational structures. As such, this category should not be neglected in the quest for leading indicators for team performance.

Candidate Leading Indicators	Potential Operational Measures	Potential Outcomes to be Predicted (work in progress)
Management Perception of Loss of Control over TSP team	Survey results from both team leaders and managers regarding the health of the relationship; The degree of friction or conflict between the team and the management	
Management perception of insufficient resources	The degree of resources needed vs currently in place; the probability of the team exceeding the resource request; the track record for TSP team resource overruns in the past; the difficulty in attracting additional resources when needed	
Management support for PSP training	The number and percentage of team members not trained in PSP; the training budget allocated for the team; the number of days per year allocated for team member training	
Networking as a mechanism to resolve management issues	The number and percentage of significant issues that the team leader communicates and/or solves via networking within the organization; Number of issues not solved via networking	
Management communication of team goals	The face time of management communicating with the team; The degree to which management communicates the importance of the team goals to the organization; Survey results on the team member evaluations of management communication	
Management trust of the software team	Survey of management's trust in the software team; The number of times that management expresses a lack of trust in the software team; Survey results from the organization regarding management's trust in the software team	
Periodic reports to Management	The number, frequency, timing, quality and usefulness of periodic reports to management	
Communicating solutions corresponding to problems	The number and percentage of the problem situations when a problem is communicated without an accompanying proposed solution	
Reports to Management meet their needs	Survey results of management satisfaction of team reports; Actions and decisions facilitated by the team's reports	
Management requests are handled properly by the team	The number of management requests placed on the team; the number handled vs not handled; the impact of servicing the dynamic requests; the degree to which management or the team leader suffered a surprise	
Multi-tasking imposed on team members	The number of tasks handled on average by the team members; The number of changed tasks in a given work day; the estimated amount of lost time due to changing tasks	
Team member training available and utilized	The amount of training afforded to team members internally vs externally on an annual basis	
Workspace	The degree to which team members raise complaints about the workspace; the lost time, rework, etc… resulting from workspace issues; the attrition of team members due to workspace issues; the cycle time required to remedy a workspace issue; the degree of preventive measures taken related to carpal tunnel syndrome, etc…	
Data Confidentiality	The amount of training related to data confidentiality; the amount of process addressing data confidentiality; the number of breaches or misuses of data; team member perception of the degree of data confidentiality or lack thereof	
TSP Leader balances priorities	The analysis of the coach and team members of the team leader's ability to balance priorities; the number and percentage of the time that the team leader fails to balance priorities; the impact of unbalanced priorities; the degree that the team leader seeks help to balance priorities	

Table 7: Factors related to Relationship with Management

In the last table, Table 8, ideas for leading indicators of team performance related to the maintenance of the team are listed. This list supplements the previous lists associated with building, motivating and operating teams. Team maintenance remains a significant need and challenge. The authors have witnessed organizations so sensitive to this issue, that they enforced a policy of off-site team building activities any time the membership of the leadership team occurred. Although sometimes expensive, this policy ensured leadership teams operated in a healthy fashion and prevented the significant negative consequences and business down-turns due to dysfunctional leadership teams.

Candidate Leading Indicators	Potential Operational Measures	Potential Outcomes to be Predicted (work in progress)
Team reassessments of common sense of membership	The frequency of revisiting the common sense of membership via survey or interview of team members; the degree to which signs exist of a lack of common sense of membership	
Team Communication	The frequency and nature of team communications; the degree to which urgent vs non-urgent communication is conducted; the degree to which miscommunications disrupt team operations and cause conflict; the time spent by team members each day in communication	
Frequent Team Meetings	The planning, efficient conduct and results of team meetings	
Team Openly Resolving Issues	The number and percentage of issues not resolved after the first team meeting discussing the specific issues	
Common Workspace	The degree that the team is collocated; sharing facilities; using common platforms and technology; the degree that workspace issues cause problems with the team operation and performance	
Team reassessments of team goals	The frequency and need to reassess team goals; team member perceptions that team goals are overdue for reassessment	
SMART and visual goals	The degree to which the team goals meet or don’t meet the SMART criteria; The degree to which the team goals are depicted with status in a visual way, in the team work area	
Team reassessments of team ownership	The frequency with which the team conducts a reassessment; the degree of team member perception that the reassessment is overdue	
Team reassessments of team planning	The frequency with which the team conducts a reassessment; the degree of team member perception that the reassessment is overdue	
Team reassessments of team quality commitment	The frequency with which the team conducts a reassessment; the degree of team member perception that the reassessment is overdue	
Interest and Competence	The team leader and/or coach determination of the degree to which team member interest and competence remain high	
Burnout	The degree of team member overtime; the degree to which team members eat meals in the office; the degree of team member attrition due to workload; the degree of stress that team members appear to be suffering; the degree to which abnormal and/or simple errors are made	
Challenging Work	Survey results of team members depicting the degree to which their work challenges them; The degree of challenging tasks that team members assume outside of the current team’s responsibilities	
Professional Discipline	The degree to which team members view software engineering as a discipline; the degree to which team members participate in professional societies	
Fairness	Survey result of perception from team members	
Evaluations based on task and relationship maturity	From the coach and team member standpoint, the degree to which the team leader uses the proper style based on the task and relationship maturity of the situation	
Individual measurement causing counterproductive behavior	The degree of counterproductive behavior occurring due to unwise or ill conceived measurement; the degree to which planned measures are subject to an FMEA analysis or Poka Yoke mistake proofing analysis	
Coaching provided to individuals	Survey results from team members indicating their satisfaction of coaching provided by the team leader	
Difficult team members properly handled	The effort and time expended to deal with difficult team members; the delay in dealing with difficult team members; the degree of successful conclusions in dealing with difficult team members	
Handling poor performers	The degree of time that poor performers are dealt with properly by the team leader	
Setting team improvement goals	The degree of participation of team members in establishing improvement goals; the freshness vs stagnation of improvement goals; the degree to which improvement goals challenge and stretch the team	
Adopting a team improvement strategy and process	The degree of definition, documentation, communication, training and use of a team improvement strategy and process	

Candidate Leading Indicators	Potential Operational Measures	Potential Outcomes to be Predicted (work in progress)
Developing a team improvement plan with resources	The degree to which a team improvement plan has measureable and testable criteria; the degree to which the improvement plans are resourced and successfully achieve their goals; the degree to which management and the team willingly invest resources in future improvement plans	
Providing team improvement measures and feedback	The degree that the improvement measures address the critical improvement needs; the clarity and timeliness of feedback such that team members may take early action; the degree to which the feedback is compelling and specific and actionable	
Team benchmarking (measures, dynamic)	The frequency and target of benchmarking to meet organizational and team needs; The degree that benchmarking positively motivates the team and their performance	
Strength of the Team Leader	Survey results of the team coach and team members; the track record of the team leader in leading successful projects; the difficult experiences that the team leader has under his/her belt	

Table 8: Factors related to Maintaining the Team

1.6 CALL FOR ACTION VIA PILOTS OF PROCESS PERFORMANCE MODELS WITHIN THE TSP DOMAIN

The authors would like to solicit TSP teams' collaboration and participation in the pilot measurement of some of these leading indicators and in the formulation of process performance models that would enable more precise and timely prediction of both interim and final project outcomes. The SEI measurement team and Hill Air Logistics Center have already embarked on this activity and have numerous statistical regression models which use observations of one or more leading indicators to predict outcomes such as quality and cycle time. However, greater participation and publication of such modeling would enable the TSP community to share what worked and didn't work as leading indicators of performance outcomes of the TSP teams. The authors additionally remain excited about the opportunity to use this modeling to further cement the connection between TSP team performance and the overall organizational and business outcomes.

1.7 CONCLUSION

We hope the reader gained a greater appreciation for the nature of possible leading indicators that may be easily collected as part of the existing TSP process and implementation. Additionally, the authors hope that this paper and presentation may motivate existing TSP coaches, leaders and teams to participate in the piloting and sharing of data for these leading indicators. The TSP team implementations, compared to traditional software development structures, appear better positioned to implement these types of leading indicators and benefit from the richer experience of such a toolkit of process performance models of interim and final team performance outcomes. Lastly, we believe that the lists of potential leading indicators discussed in this paper, indeed, represent the most likely controllable factors for TSP team performance. The authors must give credit to Watts Humphrey's insight of these leading indicators as discussed in his book on Leading TSP Teams [Humphrey 06a]. In this paper, we merely proposed potential operational measures of these leading indicators and described their use within the current body of knowledge of CMMI process performance models. The authors plan a subsequent paper and presentation, similar to this paper, but at the level of the Personal Software Process in which leading indicators at the individual level will be discussed. Additionally, the complete treatment will be published in a future SEI Technical Report.

1.8 REFERENCES/BIBLIOGRAPHY

[Humphrey 95] *A Discipline for Software Engineering,* by Humphrey, Watts, Addison-Wesley, 1995.

[Humphrey 06a] *TSP Leading a Development Team,* by Humphrey, Watts, Pearson Publishing, 2006.

[Humphrey 06b] *TSP Coaching Development Teams,* by Humphrey, Watts, Pearson Publishing, 2006.

[Humphrey 10] *Team Software Process (TSP) Body of Knowledge (BOK),* by Humphrey, Watts S.; Chick, Timothy A.; Nichols, William; Pomeroy-Huff, Marsha, CMU/SEI-2010-TR-020, Carnegie Mellon University, 2010.

[Stoddard, Goldenson 09] *Approaches to Process Performance Modeling: A Summary from the SEI Series of Workshops on CMMI High Maturity Measurement and Analysis,* by Stoddard II, Robert W.; Goldenson, Dennis R., CMU/SEI-2009-TR-021, 2010.

[Stoddard 10] *Approaches to Process Performance Modeling: A Summary from the SEI Series of Workshops on CMMI High Maturity Measurement and Analysis,* by Stoddard II, Robert W.; Goldenson, Dennis R., CMU/SEI-2009-TR-021, 2010.

[Tamura 09] *Integrating CMMI and TSP/PSP: Using TSP Data to Create Process Performance Models,* by Tamura, Shurei, CMU/SEI-2009-TN-033, 2009.

BIOGRAPHY

Robert W. Stoddard

Robert Stoddard currently serves as a Senior Member of the Technical Staff within the Carnegie Mellon University Software Engineering Institute (SEI), after 24 years leadership in Software Quality and Reliability with both Texas Instruments and Motorola. Robert is responsible for the development and delivery of advanced training on software measurement and analysis, specifically in regards to CMMI High Maturity performance modeling. Robert continues to promote the integrated use of Six Sigma and CMMI as exemplified in a 2008 published book entitled “CMMI and Six Sigma: Partners in Process Improvement” by Addison Wesley. Robert holds certifications with the ASQ on five topics including Six Sigma Black Belt, Reliability Engineering, Quality Engineering, Software Quality Engineering and Quality Audits. Robert became a Motorola-certified Six Sigma Black Belt in 1993 and Master Black Belt in 2003. Robert earned a B.S in Finance and Account-ing from the University of Maine, an M.S. in Systems Management from the University of Southern California, and has com-pleted most course work towards a PhD in Reliability Engineering from the University of Maryland at College Park.

Shigeru Sasao

Shigeru Sasao is a research associate at Carnegie Mellon University (CMU). Before joining CMU, Sa-sao was a software engineer for Deutsche Bank in Tokyo. There, he was responsible for developing and maintaining fixed in-come trading systems used across major financial cities in Asia. Sasao holds a Master of Software Engineering (MSE) degree from CMU.

Dave Webb

David R. Webb is a Technical Director for the 520th Software Maintenance Squadron of the 309th Software Maintenance Group at Hill Air Force Base in Utah, a CMMI Level 5 software organization. David is a project management and process improvement specialist with 23 years of technical, program management, and process improvement experience on Air Force software. He is a SEI authorized instructor of the Personal Software Process, a certified Team Software Process launch coach, and he has worked as an Air Force manager, SEPG member, systems software engineer, lead software engineer and test engineer. He is a frequent contributor to technical journals and symposiums. David holds a degree in Electrical and Computer Engineering from Brigham Young University.

Jim VanBuren

Jim Van Buren is a Program Manager for the Charles Stark Draper Laboratory currently supporting the Modular Control Equipment (MCE) project team at Hill Air Force Base. He is a SEI authorized PSP instructor and TSP coach. He holds a BS in Computer Science from Cornell University and has over 30 years of software development and management experience.

New Team Software Process Paths: Systems Engineering Team Uses TSP

Daniel M. Wilson, *Wyle Laboratories, Inc.*

The E-2C Systems Engineering Team launched a Team Software Process (TSP) project in March 2009, to develop Systems Engineering test documentation and to support the software development to integrate Automatic Identification System (AIS) onboard the E-2C aircraft. We worked in parallel with the Advanced Control Indicator Set (ACIS) software development team, to create the AIS software package Graphical User Interface (GUI) development. Our tasking was separate with a different lower-management structure; however we contributed into the same weekly rollup and shared an integrated team meeting. Additionally, we worked with the Mission Computer (MC) team although we did not share any meetings and the communication with them was far less. The software development teams are made up of a mix of different contractors as well as many Government employees. The goal was to find out if there was a productivity benefit to a systems engineering team utilizing TSP.

1.1 E-2C SYSTEMS ENGINEERING TEAM

Our Systems Engineering (SE) team consists of 12 engineers, made up of mostly Wyle employees, with the majority having more than 10 years experience in the E-2C community, and more than 15-20 years experience in system test and evaluation. Most of our systems engineers are former Navy personnel which gives them operational knowledge of military software as well as the tactical significance. Planning and developing software test products that exercise these different areas of experience is very familiar to this experienced group of professionals. Traditionally the E-2C used informal processes to develop test procedures. All were unique for obvious reasons, yet the systems engineers required little direction and were comfortable working autonomously. Their work has consistently been highly rated by the customer and they are known for getting the job done.

1.2 PREVIOUS METHOD

In past development, our systems engineers did not become involved with a new project until the Preliminary Design Review (PDR) for that project. After PDR the software developers would invite systems engineering to Meeting 1 and 9 as they launched the development phase. The systems engineers would then start reading requirements and preparing for the first release of software, at which point they could begin to capture the steps required to test the functionality. We were often faced with limited functionality, usually playing catch-up when it came to bringing our Software Test Document to maturity prior to the Functional Evaluation Test and the Acceptance Test events.

1.3 TSP EXPERIENCE

Prior to launching our own product development project, our systems engineers had worked on an integrated TSP requirements development team made up of themselves, the ACIS and Mission Computer (MC) teams. One engineer had performed the quality role even though it was our first TSP participation. Our tasks were limited to higher-level requirement writing and inspection of the lower-level requirements. The team we participated with had a great deal of experience using TSP and they provided a key formative role in developing our own systems engineering specific processes. With this experience we began gaining knowledge about TSP though it was a challenging learning curve.

1.4 THE LAUNCH

Planning for our project was challenging as expected in a first-time launch. The lack of historical data referenced by the TSP techniques, lead to some difficulties. We had to figure out how to divide the tasking between two systems engineers for our physical product – the Software Test Description (STD). In the past, this was usually written by one engineer with the other engineer(s) offering suggestions, etc. We did, however, have the advantage that we had previously participated in the requirements development with an experienced TSP team – the ACIS team. Testing has a natural fit with the GUI interface and this allowed us to define our tasks along the same lines. We divided the STD into individual test cases and defined those test cases by the same divisions we had used in the requirements development. We were then able to generate an accurate number of requirements per test case and therefore define a size estimate for each one. We didn't realize it at the time, but this methodology would result in huge benefits down the road.

Next, we had to give serious thought to what we would use as a measure. As systems engineers, we don't usually utilize Source Lines of Code (SLOC) as a unit of measure. We ended up choosing the number of pages per test case for sizing. The most challenging decision was defining the non-product tasks such as Bench Testing time. We arbitrarily chose a Bench Testing task to validate the STD after final integration of the individual test cases.

Our launch was challenging because we had to rely on unsupported estimation for most aspects of the project due to a lack of historical documentation for sizing and level of effort. We took the divisions used during the requirements development, calculated the percentage of the total then applied that percentage to the time we previously would have estimated to develop the STD. Surprisingly, our estimates proved to be fairly accurate and we made very few changes to our workbooks overall. The hidden benefit in this was that the tasks in our workbooks ended up mimicking the ACIS group's development tasks almost identically. This kept both teams in parallel and made questions between the teams more manageable due to similar project information being "fresh" in the minds of each team.

1.6 EXECUTION

The development of our product worked extremely well in the first half of our project. Having fully participated in the requirements development as inspectors, we each had valuable insight and knowledge of the developing software that we had never enjoyed before. Furthermore, software engineers would seek clarification of requirements with our engineers as a courtesy. We had built a team trust that crossed group boundaries and opened up communication that simply had not existed previously in a software development phase. As mentioned earlier, our test cases followed the same divisions as the software development. Each group invited the other to product and code inspections which led to the test cases being developed in parallel with the code that it was designed to verify. This made our tests more accurate and produced more robust software as a result. Each group's expectations of how something worked was corrected early and often, preventing defects from being found later in more costly phases of production. We also noted in the combined team meetings that many tasks assigned to each team, were worked on simultaneously. TSP is very linear in its orientation with the focus being completing tasks and then moving on. Our test cases needed to be built simultaneously with a portion of the simulation scenario that catered to it. This opened the scenario task earlier than anticipated and kept it open for a longer period of time. We were also challenged by inspecting the scenario. We could not quantify inspection criteria and we basically verified that it worked with the test cases and therefore needed no changes.

1.7 PROBLEMS

TSP has an entire infrastructure that needs to be supported. We did not have any of the products needed to conduct the analysis and inspections that are necessary.

As our first review approached, we realized that we didn't have a personal inspection criteria sheet or an inspection spreadsheet. We had never developed a software test description in pieces as a team, therefore, we had to design our criteria "on the fly" to make sure we could assemble the pieces as far as content and structure. The test cases had to conform or we risked having a final disjointed and dissimilar document. We had to produce a personal review, inspection form, and a template for the test case level of a Software Test Description. We also realized we would need an additional personal review for the Software Test Description once we integrated the test case into the single document. A great deal of off-task time was incurred building the TSP support structure that had not been foreseen.

The next issue we encountered was that our SE team consisted of two people for this project. The TSP roles outnumbered us four-to-one. With the additional task of building the TSP support products along the way, we were simply unable to accomplish the roles in a meaningful way. This was offset somewhat by sharing the weekly meeting with the ACIS team. They performed their roles as experienced professionals and provided the necessary insight into the project as a whole. The lack of role fulfillment will hurt us more in planning for the next project. It is worth noting that as TSP branches into other territories beyond software engineering, the same volume of personnel may not be available to fully staff the TSP roles on smaller projects.

Finally, we found the timeline of our development shifted drastically due to the efficiency of TSP and aligning ourselves with the software developers. We completed our test cases prior to the first drop of software, therefore putting us ahead of schedule and having to wait for the MC team before moving forward. We had never before completed the test cases that early and our next tasks required testing the software and using our test cases. We found ourselves with a five-week gap waiting for the software development teams to complete the first builds. They weren't behind; we were simply that far ahead. We made use of the time by moving our simulations scenario task ahead and we also spent many off task hours supporting the MC team with iterative non formal testing. This helped them to achieve their release date and helped us get an earlier look at the software. In hindsight, we also should have moved development of some training materials we normally produce at the end of a project to this point in the process. You have to be flexible and willing to disregard all previous expectations and adapt to unforeseen success.

1.8 RESULTS

Systems Engineering:

- Produced test cases months earlier than with previous methods
- Forced a standardization of our products with templates and review criteria
- Formalized our processes
- Increased the buy-in of all engineers on a product by giving each a portion to develop
- Systems engineers obtained an outstanding awareness of the implementation schedule

Integrated Team:

- Increased communication, knowledge and trust, by having an integrated Team meeting and forced an alignment of effort that reduce responses to questions
- Had only two System Trouble Reports (STR) with our first configured release of software – an astonishingly low number

There is no doubt about the results. TSP usage by a systems engineering team can not only improve productivity and quality internally, but carry those same gains throughout the entire software organization involved in software development.

BIOGRAPHY

Daniel M. Wilson
Systems Engineer
Wyle
Supporting PMA 231

Daniel Wilson is a Wyle Employee and a member of the systems engineering staff in the Product Support Activity (PSA) for the E-2C Hawkeye aircraft, supporting NAVAIR PMA 231 in Patuxent River, Maryland.

Wilson also is a member of the SEPG board within that organization. Before Joining Wyle and the PSA, Wilson was in the fields of CBT programming, Systems Training, and a former U.S.Navy Aircrewman.

Wilson is new to TSP, having participated in one launch as a member of a team that has done TSP for 7 years.

- - - - - - - - -

NAVAIR Public Release 10-(882)

Distribution: Statement A- "Approved for public release; distribution is unlimited"

Achieving Academic Success Using the Team Software Process

Berin Babcock-McConnell, *Carnegie Mellon University*
Saurabh Gupta, *Carnegie Mellon University*
Jonathan Hartje, *Carnegie Mellon University*
Marsha Pomeroy-Huff, *Software Engineering Institute*
Shigeru Sasao, *Carnegie Mellon University*
Sidharth Surana, *Carnegie Mellon University*

ABSTRACT

Team VdashNeg, a student group in the Master of Software Engineering program at Carnegie Mellon University, was tasked to build software to autonomously control a robot for a real-world industry project. The team was having difficulty creating a project plan which could effectively track their progress, and decided to try the Team Software Process (TSP). By using TSP, the team delivered the product to the clients a week ahead of schedule, with only two documented defects at system test in over 20K LOC. Most importantly, TSP taught the team how the software engineering puzzle fits together, and helped the team to mature as engineers.

1.1 INTRODUCTION

The Master of Software Engineering (MSE) degree is a sixteen-month graduate program offered at Carnegie Mellon University. The curriculum consists of five core software engineering courses, four electives, and the studio project, and prepares graduates to enter the software engineering field as project managers and software architects. In the core courses, students learn techniques in formal models, requirements gathering, project management, and architecture design, among other topics in software engineering. For example, during the Management of Software Development class, students may learn how to construct a work breakdown structure, conduct estimation, and use earned value analysis to track project progress [Pressman 05], [Tsui 07]. These techniques can be applied directly to the studio project, which runs continuously throughout the duration of the MSE program and provides students with a practical workspace on an actual industrial software engineering project provided by corporate sponsors. Studio teams consist of four or five students each who work with their project customer to analyze, design, implement, and deliver a working solution to the customer's business needs. Coupled with support, feedback, and critical analysis from the MSE faculty, the studio project provides students with a supportive environment in which to organize and practice their software engineering craft without the pressures or consequences brought on by real-world business markets. Faculty, customers and students alike often cite the studio project as the cornerstone of the MSE program and believe it is largely responsible for the high quality performance of its graduates.

The VdashNeg team operated during the August 2008-December 2009 academic cycle, and was composed of five individuals with diverse professional and academic backgrounds. Their studio project, which was sponsored by the Software Engineering Institute (SEI), required the team to create a reference implementation demonstrating the capabilities of an SEI technology called Predictable Assembly from Certifiable Components (PACC) [Hissam 04]. The PACC technology enables developers to create software components that are predictable in both behavior and performance. These components then can be combined to create a system that also exhibits predictable behavior and performance. The clients asked the student team to use the PACC technology in building a software system that could autonomously control a commercially-available robot called the SRV-1 Surveyor Robot [SRV1 10]. This posed a major challenge to the team, since none of the team members had prior experience with the key technologies used in the project, which were robots, image processing, and the PACC technology. Another challenge was that the client had no firm requirements for tasks that the robot would perform, so the team also had to establish its own set of project requirements. The team decided to build a "search and destroy" mission controller for the SRV-1 robot.

Four months into the project, after several failed attempts at using different planning techniques, the team was still having difficulty creating a project plan which could effectively track their progress. The team heard that the Team Software Process (TSP) might address many of the issues that they were facing, so they decided to try applying TSP to their project.

1.2 USE OF THE TSP

The major issues that led the team to choose TSP as their development process were as follows.

- **Inability to map team goals and milestones to tasks:** At the beginning of both the fall 2008 and spring 2009 semesters, the team decided on a series of goals for that semester, and those goals were further divided into milestones. However, the team had difficulty decomposing the milestones into a more granular list of subordinate tasks. For example, one of the semester goals was to develop a system prototype. To develop a useful prototype, the team needed to refine the requirements and comprehensively study the domain. These sub-goals were used as high-level milestones, but the team did not know how to convert those milestones into tasks that could be mapped to a weekly or daily schedule. This difficulty caused the team to struggle with their project planning and tracking.

- **Incomplete software process:** The team had selected the Architecture-Centric Design Methodology (ACDM) as its guiding process [Lattanze 08]. ACDM is a methodology for developing a software design, but it provides no detailed guidance in the areas of planning, tracking, risk management, and quality control. The team selected different techniques that they learned in the core courses to address these missing areas. However, because they had no comprehensive framework for the whole project lifecycle, the team struggled to tie together the different pieces to create one cohesive process.

Once these problems were recognized, the team decided to use TSP because the members agreed that it could provide a comprehensive process framework for the project, as well as detailed planning and tracking mechanisms. One of the team's mentors, a certified TSP coach, agreed to coach the team and guided them through the week-long launch process in the middle of the spring 2009 semester.

The launch helped the team to reevaluate and clarify the project goals. Instead of having a comprehensive "laundry list" of goals, the goals were divided into subsets of client goals, team internal goals (specific to processes, team performance, and the like), and team external goals (specific to meeting the clients' stated and implied goals). Because these goals were achievable, measurable, and had deadlines associated with them, they helped to form the skeleton for the rest of the project plan. In the remainder of the launch, the team made a detailed task list that could be accomplished in the budgeted available time and balanced the workload equally among the team members. Each task on the list had an associated estimate for completion date. The task/schedule list, when coupled with earned value analysis and dynamic plan rebalancing during the TSP weekly meetings, enabled the team to effectively track their progress over the remainder of the project.

Prior to using TSP, risks were identified and documented but not well managed. Mitigating strategies had not been documented, and risks were not being effectively assessed, or, if necessary, mitigated. After the switch to TSP, the team actively tracked the issues and risks in the Issues and Risk Tracking Log (IRTL). Due to the prescriptive and active risk tracking system, the team became very effective in tracking, mitigating, and managing risks that might have endangered or derailed the project if they had occurred before the switch to TSP.

TSP also helped the team to be more organized. TSP has team manager roles with very specific responsibilities and minimal overlap between them. Using the roles prescribed by TSP ensured that the responsibilities were shared equally among the team members and were promptly addressed. The team also used the TSP meeting scripts to make their weekly team meetings more focused and productive than before.

The most significant changes after the switch to TSP were in the team's planning and tracking process and the quality control. These changes are discussed in detail in the subsequent sections.

1.3 PROJECT PLANNING AND TRACKING

Before using TSP, team members did not know which tasks to complete in a given week, nor did they have any idea as to when the remaining tasks would be accomplished. As mentioned above, this issue was the main reason that the team decided to switch to TSP. After the first launch in February 2009, the team had a shared understanding of the project goals, a list of risks evaluated by impact and likelihood, a set of tangible outputs to be produced for the project, and most importantly, a detailed and balanced plan that could be used to track the progress of the work. The important lesson learned from the launch was that there is a process that needs to be followed in order to create a high-quality project plan. First, the team must understand the business goals of the client. Second, the team must understand their own goals, which are derived from the business goals. Third, the team must identify both the tangible outputs that correspond to the team goals and the processes that will be used to produce those outputs. Finally, the team can create a task list, determine the tasks' feasibility in relation to the team's available resources, identify which goals can be reasonably achieved within the parameters of identified assets and constraints, and refine the plan as necessary. The plan must also include steps to help to ensure high product quality by including reviews and inspections at appropriate times during product development. This project planning process, which is documented in detail in the TSP launch scripts, was key to creating the high-quality plan that enabled the success of the project.

The term "period of uncertainty" is defined as the time spent in the project prior to establishing a software architecture [Lattanze 08]. Until the end of the spring 2009 semester, the team was in the period of uncertainty. The team finalized the software architecture at the end of the spring semester and entered the period of certainty at the beginning of the summer semester, following the team relaunch in mid-May. The nature of the plans before and after the completion of the architecture was drastically different. While the spring semester plan concentrated on finalizing the detailed requirements and eliciting architectural drivers such as quality attributes and technical constraints, the summer semester plan was focused on implementing the product. The conceptual design used to build the summer semester plan was based on the dynamic view of the software architecture, which is shown in Figure 1. The assemblies in the TSP's Form SUMS were mapped directly to the architectural components in the diagram.

The summer implementation work was divided into four iterations of three weeks each to allow the team to continuously reflect and improve their plans, estimations, and processes. During the first iteration, the team noticed that tasks assigned to different team members were more highly interdependent than had been the case during the spring semester. For example, Team Member A needed to complete coding a particular component before the team could conduct

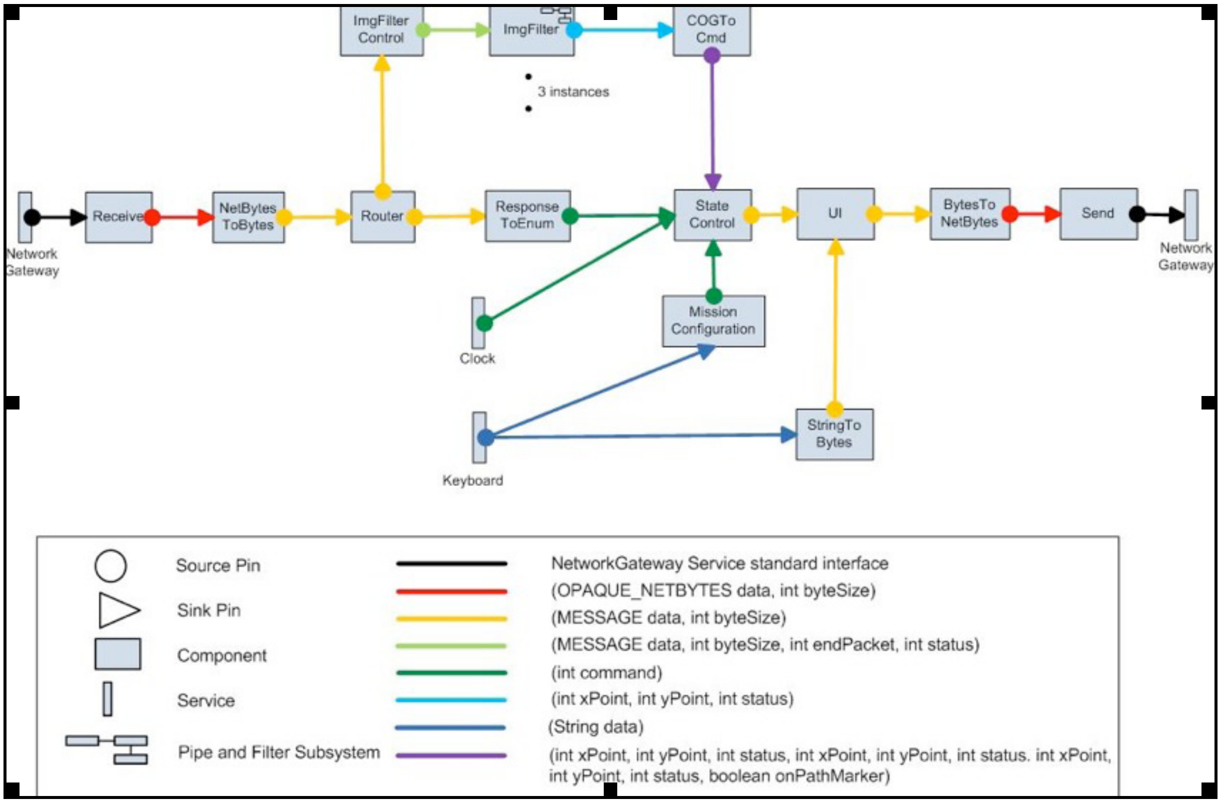

ImgFilter Control
ImgFilter
COGTo Cmd
3 instances
Network Gateway
Receive
NetBytes ToBytes
Router
Response ToEnum
State Control
UI
BytesTo NetBytes
Send
Network Gateway
Clock
Mission Configuration
Keyboard
StringTo Bytes
Source Pin
Sink Pin
Component
Service
Pipe and Filter Subsystem
NetworkGateway Service standard interface
(OPAQUE_NETBYTES data, int byteSize)
(MESSAGE data, int byteSize)
(MESSAGE data, int byteSize, int endPacket, int status)
(int command)
(int xPoint, int yPoint, int status)
(String data)
(int xPoint, int yPoint, int status, int xPoint, int yPoint, int status. int xPoint, int yPoint, int status, boolean onPathMarker)

Figure 1: Dynamic perspective of the software architecture

the code inspection of that component. Therefore, the team developed a custom dependency-tracking process to supplement the planning and tracking done using the SEI-developed TSP tool. The team developed what they called "the matrix," a table in which the architectural components were shown as rows and the development phases as columns. As team members completed tasks (such as the design review for component A), they marked the completion of that task in the table by filling in the appropriate cell. The matrix was posted in the common working area and served as a notice board, signaling other team members when they could begin work on their dependent tasks.

The following sections will discuss estimation and project tracking during the summer period.

1.3.1 ESTIMATION

The team overestimated the schedule for the project work during the summer semester. This is in contrast to the common notion that engineers tend be overly optimistic about their productivity and therefore usually underestimate the time needed for the work [Buehler 94], [Jorgensen 07]. Because the team was newly formed, they had no historical data on which to base their estimates. Also, the project required the use of a proprietary programming language called CCL, which made it difficult to use industry data for estimating. Therefore, the effort estimations for the first two iterations of the summer semester were conducted using Wideband Delphi estimation. Wideband Delphi is an expert judgment estimation technique, which attempts to reduce variations in estimations from individual team members by having multiple rounds of estimation [Humphreys 95]. During the first two iterations, the team overestimated their development tasks by 134%.

For the third iteration, the team chose to use PROBE estimation, using historical data from the first two iterations. PROBE is a parametric estimation technique using linear regression [Humphreys 95]. The sizes of the architectural components were used as the dependent variable for predicting effort. The team created several linear regression models using a combination of physical lines of code (LOC) or logical LOC. Out of the 16 components developed in the first two iterations, 14 data points were used for the model and 2 were set aside for validation. The team chose the physical LOC model excluding one outlier point, which produced the best validation result and the highest linear correlation.

After the model was created, the team conducted a Wideband Delphi for the sizes of the components to be developed in iteration 3. These size estimates were used as inputs into the model to estimate the development effort for individual components. However, the team was risk averse and decided (against their coach's advice) to pad the estimates produced by the model. At the end of iteration 3, the padded estimates from the PROBE model were 52% higher than the actual component sizes, representing a significant improvement over the 134% overestimation in the first two iterations. However, if the team had used the output from the PROBE

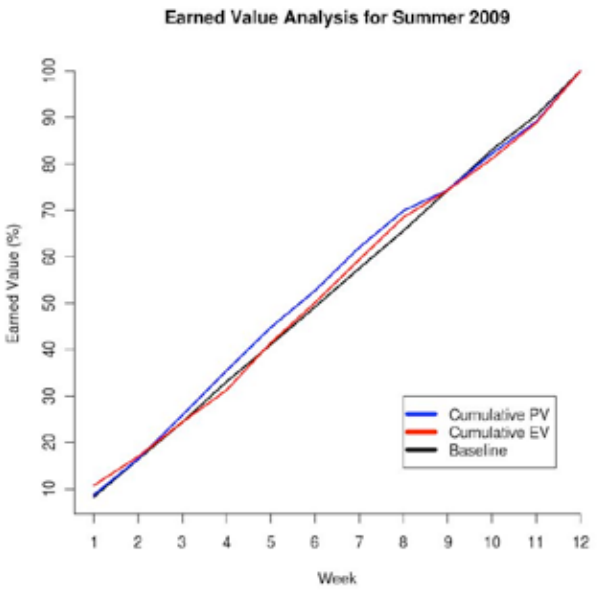


Figure 2: Earned value chart for summer 2009 semester

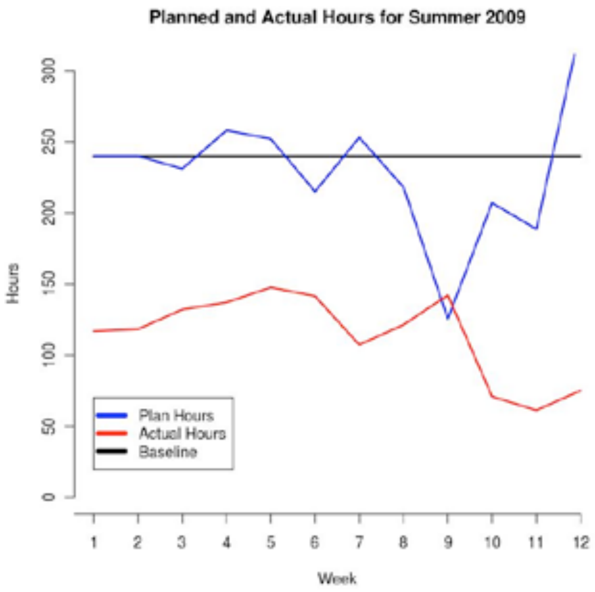


Figure 3: Planned vs actual hours per week for summer 2009 semester

model without padding, the estimation would have been too high by only 19%.

1.3.2 EARNED VALUE ANALYSIS

Figure 2 shows the earned value chart for the summer semester. The 10% earned value gained in the first week includes the TSP launch. Figure 3 shows the actual and planned effort per week for the same period. While the small deviation between the lines in Figure 2 show that the team was able to meet their weekly planned values, Figure 3 shows that the team spent less effort than planned to complete their tasks. This can be attributed to two factors:

- The team spent most of their available effort resource of 12 hours per person per week during the spring semester on on-task time. When the available resources for effort increased in the summer semester to 48 hours per person per week, the team's plan used the same assumption that 100% of the time available hours would be spent on project tasks.
- As mentioned above, the team overestimated the effort required for implementation tasks.

The team used the earned value (EV) analysis effectively to track the team's progress during the summer semester. Each week, the team held a status meeting using the TSP weekly meeting script. During the meeting, the team discussed both the individual EV status and the consolidated earned value for the team, along with any risks that threatened to impact the team goals. Because of their attention to maintaining progress against the schedule and quality plans and the timely mitigation of risks, the team beat its projected schedule and actually delivered the product one week earlier than planned.

1.4 QUALITY

1.4.1 DATA FROM THE SUMMER 2009 SEMESTER

Because the metrics used to guide the quality decisions are primarily lagging indicators, the team spent the first iteration somewhat in the dark as to its quality performance. However, at the conclusion of the first iteration, some quality data started to become available and the team's quality profile began to emerge. Because of the short time available for the project iteration and the fast pace required to meet the schedule goals, the relevant phase of the work was often complete by the time that clear quality trends became visible. Even with the limited timeframe, however, it was still possible to analyze the data and use the results to improve some of the team's processes. Notably, the team was able to refine the checklists used for design and code reviews and inspections.

1.4.2 THE FILTERS

At the conclusion of the summer semester, a clear image of the team's quality profile became visible. A portion of this profile is discussed below.

Table 1 describes the summer cycle defect data. The data show that 54 defects were injected during the detailed design phase, and none were removed during the detailed design phase; therefore, 54 defects remained at the end of detailed design. During the detailed design review phase, no defects were injected and 31 defects were removed; thus, at the conclusion of the detailed design review phase, the defect total had dropped to 23 defects remaining in the product. This table also shows that 85 defects were injected during detailed design and code, and all 85 defects were removed by the end of the system test phase.

Phase	Defects In	Defects Out	Running
Net Total			
Detailed Design	54		54
DLD Review		31	23
DLD Inspection		11	12
Code	31	3	40
Code Review		14	26
Code Inspection		12	14
Unit Test		10	4
Build & Int. Test		2	2
System Test		2	0
Total	85	85	0

Table 1: Defect data for summer 2009 semester

Based on the defect data shown in Table 1, phase and process yields for the summer cycle were calculated. These are shown in tables 2 and 3.

DLD Review	57%
DLD Inspections	48%
Code	7%
Code Review	35%
Code Inspection	46%
Unit Test	71%
Build and Integration Test	50%
System Test	100%

Table 2: Phase yields for summer 2009 semester

% Before Compile	69%
% Before Unit Test	84%
% Before Build & Int. Test	95%
% Before System Test	98%
% Before Acceptance Test	100%

Table 3: Process yields for summer 2009 semester

Table 2 shows the phase yields for the summer cycle. For example, the detailed design review phase captured 57% of the defects present at the beginning of that phase. Table 3

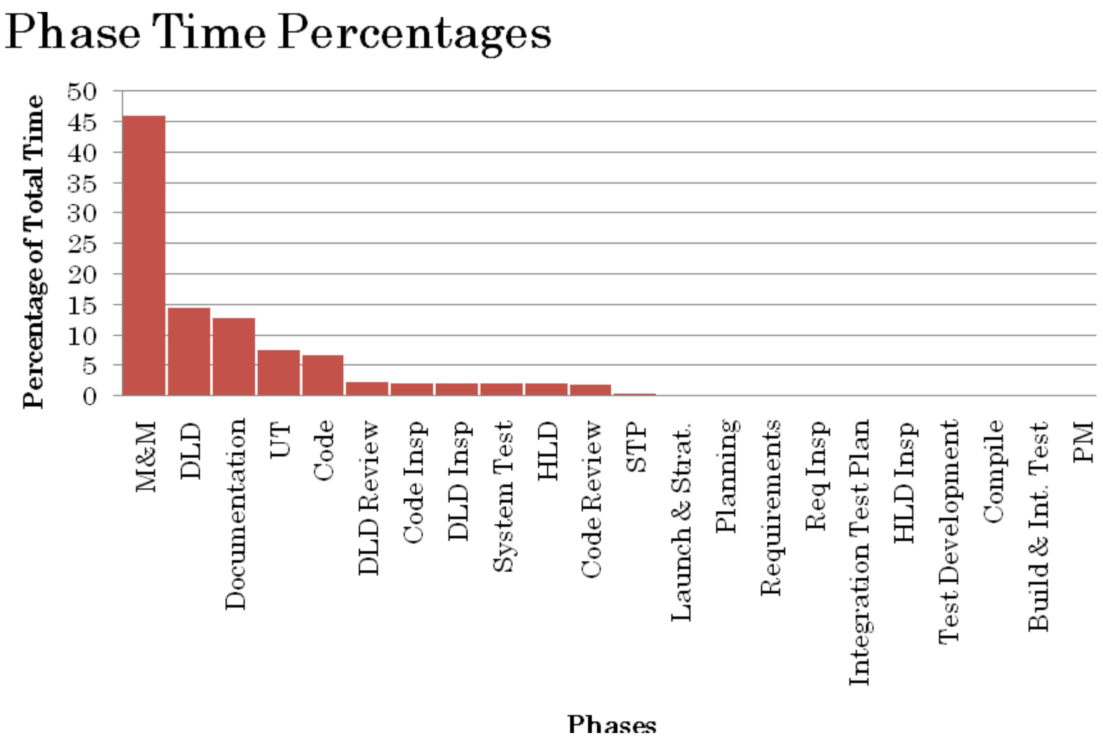


Figure 4: Phase time percentages

shows the process yields for the summer cycle. For example, 69% of the defects were removed prior to the compile phase. Taken together, these two tables clearly illustrate the defect-filtering effect of the TSP phases. As the defects flow through the process, each phase removes a percentage of the defects (phase yield), resulting in an increasingly defect-free product (process yield). The effectiveness of these filters is highlighted by Table 3, which shows a process yield of 100% before acceptance test. No defects have been reported by the clients after delivery. It is established in literature that it is cheaper to catch defects earlier in the lifecycle, and defects found in later stages such as system test take more work to fix [Humphreys 95]. The filtering effect of the TSP phases resulted in only 2 defects remaining in system test; the attention to product quality contributed greatly to the team delivering one week ahead of schedule.

1.4.3 TIME IN PHASE / FOCUS ON DETAILED DESIGN

Although the team released the product with zero defects, the team's phase and process yields were below the TSP quality planning guidelines. In the postmortem, the team examined the time-in-phase data as a way to find areas for improvement in its phase and process performances.

Figure 4 shows the percentage of the total summer cycle hours spent in each phase. The phases have been arranged from left to right in descending order. This makes it easy to see the relative amounts of time consumed by each phase. For example, the Management and Miscellaneous phase consumed the most time, which represents 46.0% (630.7 hours) of the total hours (1372 hours) used in the summer cycle.

Table 2 shows that the phase yield for the detailed design review phase was 57%; however, the TSP quality planning guidelines suggest a phase yield of 70%. As shown in Figure 4, the percentage of time spent in the detailed level design review phase was significantly lower that the time spent in the detailed level design phase. The team spent 199 hours in the detailed level design phase and 31.3 hours in the detailed level design review phase. This yields a DLD Review-to-Detailed-Design ratio of 0.16. However, the TSP quality planning guidelines suggest a ratio of 0.5 or greater. Thus, in the future, the team might consider spending more time in the detailed level design review phase, with the goal of improving both the phase yield for the detailed design review phase and the team's overall process.

1.5 CONCLUSIONS

In their journey through the studio project, the team felt that TSP was not only a valuable process management tool, but also a powerful method for showing them how the pieces of the software engineering puzzle fit together. TSP provided multiple techniques in one cohesive package, allowing the team members to understand software engineering at a much deeper level. For example, TSP showed the team how and where they could gather data and how they could analyze that data to improve their process. Because these metrics are lagging indicators, the team did not always find it possible to improve their process within the bounds of the twelve-week implementation cycle. However, the TSP has provided the team with concrete data which they can use to improve their process for future cycles.

The team also delivered the software product to their clients a week ahead of schedule. During final system testing, the system had only two documented defects in system test over 20K LOC. The team also eliminated maintenance costs because there were no enhancements or bug fixes to be made. By contrast, the other teams in the MSE studio program spent an additional two months in the fall 2009 semester on bug fixes and enhancements. The TSP enabled Team VdashNeg to turn a struggling project into a well-planned, well-managed learning experience in which the team members could see how the engineering principles and methods presented in their core courses could be meshed into a cohesive process framework in which a high-quality software product was developed on time and within budget.

1.6 REFERENCES

[Buehler 94]
Buehler, Roger, *Exploring the "Planning Fallacy": Why People Underestimate Their Task Completion Times*, American Psychological Association, Inc. 1994.

[Hissam 04]
Hissam, Scott; Klein, Mark; Lehoczky, John; Merson, Paulo; Moreno, Gabriel; & Wallnau, Kurt, *Performance Property Theories for Predictable Assembly from Certifiable Components (PACC)*, CMU/SEI-2004-TR-017, Carnegie Mellon University, 2004.

[Humphreys 95]
Humphrey, Watts S., *A Discipline for Software Engineering*, Addison-Wesley, 1995.

[Jorgensen 07]
Jorgensen, Magne; Faugli, Bjorn; & Gruschke, Tanja, *Characteristics of Software Engineers with Optimistic Predictions*, Journal of Systems and Software, 2007.

[Lattanze 08]
Lattanze, Anthony J., *Architecting Software Intensive Systems: A Practitioner's Guide*, Boca Raton: Auerbach, 2008.

[Pressman 05]
Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, R.S. Pressman and Associates, 2005.

[SRV1 10]
http://www.surveyor.com/SRV_info.html, Surveyor Corporation, 2010.

[Tsui 07]
Tsui, Frank; & Karam, Orlando, *Essentials of Software Engineering*, Jones and Bartlett Publishers, Inc., 2007.

BIOGRAPHY

Berin Babcock-McConnell

Berin Babcock-McConnell is a graduate of the Master of Software Engineering (MSE) program at Carnegie Mellon University.

Before joining the MSE program, Babcock-McConnell was a software engineer for NEC Communication Systems in Tokyo. There he was involved in the development of application software for third generation cell phones. His responsibilities covered a broad range from requirements analysis to architecture design to implementation and testing.

Babcock-McConnell holds a BS in Computer Science from Rutgers University and an MSE from Carnegie Mellon University.

Saurabh Gupta
Software Engineer
Nextag Inc

Saurabh Gupta is a software engineer at Nextag Inc.

Saurabh holds a Master of Software Engineering degree from Carnegie Mellon University and his bachelor's degree in Computer Science and Engineering from Jawarlal Nehru Technological University, India.

Prior to Carnegie Mellon, he worked for 2 years at Adea Inc. as a software engineer. He was involved in developing software through the stages of the SDLC.

Jonathan Hartje
Project Manager
Applied Research Laboratories: The University of Texas at Austin

Jonathan Hartje is a Project Manager and Software Architect at The Applied Research Laboratories: The University of Texas at Austin (ARL:UT).

Hartje is responsible for designing large-scale, distributed software systems for acoustic signal processing, scientific and autonomous behavior computing. He manages the software engineering personnel and activities at ARL:UT resulting in the design, implementation, testing and integration of multiple U.S. Navy Sonar programs hosted on manned and autonomous underwater vehicles.

Hartje holds a B.S. in Computer Science from the University of Texas at Austin and an M.S.E from Carnegie Mellon University.

Marsha Pomeroy-Huff, Ed.D.
Member of the Technical Staff
Software Engineering Institute

Dr. Marsha Pomeroy-Huff joined the SEI in 1992 and is currently a member of the Professional Certification Program, where she oversees the SEI's TSP Mentor Coach Certification program. She also serves as an adjunct lecturer, studio team mentor, and TSP Coach for student teams in Master of Software Engineering program at CMU.

As a member of the PSP/TSP Initiative Team from 1997 to 2007, she coordinated and contributed to planning, developing, and maintaining the various courses in the PSP and TSP product suite. Dr. Pomeroy-Huff is the primary author of PSP Body of Knowledge and a co-author of the TSP Body of Knowledge and the TSP Mentor Coach Guidebook. She holds a doctorate degree in instructional design from the University of Pittsburgh and is an SEI-Certified PSP Developer and SEI-Certified TSP Coach.

Shigeru Sasao
Research Associate
Carnegie Mellon University

Shigeru Sasao is a research associate at Carnegie Mellon University (CMU).

Before joining CMU, Sasao was a software engineer for Deutsche Bank in Tokyo. There, he was responsible for developing and maintaining fixed income trading systems used across major financial cities in Asia.

Sasao holds a Master of Software Engineering (MSE) degree from CMU.

Sidharth Surana
Member of Technical Staff
VMware Inc.

Sidharth Surana is a Member of Technical Staff at VMware, Inc (NYSE: VMW)

Before joining VMware, Surana was a lead software engineer for ExpenseAnywhere Inc. in Monroeville, PA. There, he was responsible for leading the product development team developing the next generation expense management solution for the company.

Surana holds a Master of Software Engineering (MSE) degree from Carnegie Mellon University.

AIM Case Study: Moving from TSP to CMMI ML3

Oscar A. Mondragón, *ITESM Mexico*
Edgar D. Fernández, *SILAC*

This paper describes the experience of a small Mexican company, SILAC, which is using the Team Software Process (TSP) as the foundation for implementing the Capability Maturity Model Integration (CMMI) [CMMI Product Team, 2006] through Maturity Level 3 (ML3). The company is working with the Software Industry Excellence Center (SIE Center) of Tecnológico de Monterrey (Tec) and the Software Engineering Institute (SEI) to pilot the Accelerated Improvement Method (AIM), a formalization of methods used by SEI customers to combine the best of two great technologies. The main idea behind this approach is to decrease the time to reach CMMI ML3 by obtaining the synergy of combining both methodologies: self directed teams, focus in strong personal quality, strong commitments from engineers, short testing cycles, high performance, organizational process assets, and managing continuous process improvement.

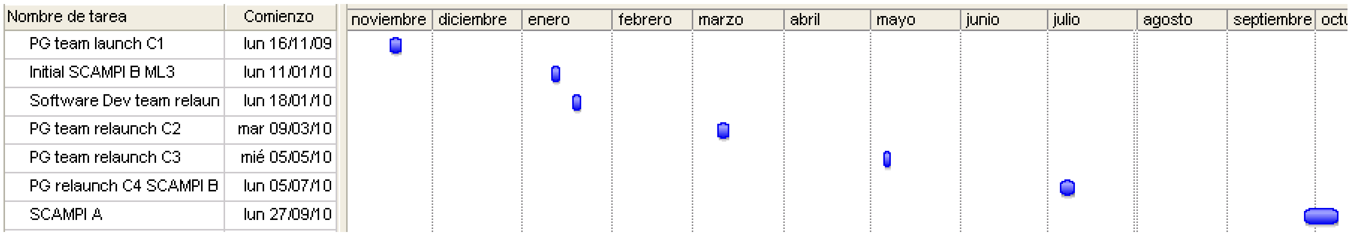
The scope of the project is to achieve CMMI ML3 processes through the use of Team Software Process (TSP) as a foundation. As a result, a guiding principle was established: missing processes should be developed, modified, or refined using the TSP philosophy and resources. A process group was created to handle this software process improvement project and this group had to be managed as a TSP team. This paper describes the project context, current project status, managing the process group, the implementation strategy for missing processes, the high-level- design document and lessons learned. The paper mainly addresses the first two project cycles describing activities, problems, and opportunity areas.

1.1 PROJECT CONTEXT

SILAC began operations in May 2006 in Zacatecas, Mexico. Its main business was customizing web applications for administrative domains, for government and private customers. A typical software project last between 6 and 12 months, has a team of 4 and 7 full time engineers, and has 5,000 to 13,000 Software Lines of Code (SLOC) in size. SILAC began using Personal Software Process (PSP) trained developers in June 2006, and began implementing TSP in December 2007. Regarding organizational assets, SILAC did not have an Organizational Set of Standard Processes (OSSP). Nowadays, SILAC has 15 people including 10 full time engineers. Mainly, one TSP software development project is implemented at a time.

The AIM project started in November 2009 with the launch of the SILAC's Process Group (PG) using the "TSP 2009.09" release and process elements from the unreleased "TSPm 2008.09." TSPm is a TSP version for managing projects with multiple teams. TSPm also considers the constitution of a process group and defines new roles to handle these new responsibilities. TSPm have also included new process elements that address some CMMI practices for process areas such as configuration management, organizational process focus, organizational training, risk management, and organizational process definition. These new process elements are the main reason for using the TSPm as starting point to reach CMMI ML3.

For this project, the PG defined four implementation cycles (around 3 months each) and the following goals: a Standard CMMI Appraisal Method for Process Implementation (SCAMPI) B and a TSP evaluation in June 2010, and SCAMPI A [SCAMPI Upgrade Team, 2006] ML3 in September 2010. A TSP evaluation has an organizational scope and characterizes aspects such as: the ratio of PSP-trained engineers; TSP project quality results; and customer reviews to TSP projects. Figure 1 shows the project cycles and milestones.



Nombre de tarea	Comienzo
PG team launch C1	lun 16/11/09
Initial SCAMPI B ML3	lun 11/01/10
Software Dev team relaun	lun 18/01/10
PG team relaunch C2	mar 09/03/10
PG team relaunch C3	mié 05/05/10
PG relaunch C4 SCAMPI B	lun 05/07/10
SCAMPI A	lun 27/09/10

Figure 1: Project cycles and milestones.

This paper describes the work accomplished through the first two cycles. The project includes two TSP teams: one Process Group (PG) team with four people with different time availability and a Software Development (SD) team with six full time engineers working in an 8 months project. SILAC was using the Process Dashboard tool [The Software Process Dashboard Project Team, 2010] for its TSP projects.

1.2 SILAC TSP BACKGROUND

SILAC started up with two programmers having no process for doing software development. Projects were web-based and mid-sized having lots of defects in user acceptance test. The small team has code control by allowing only one developer accessing the code at a time. After the first PSP training, SILAC found a major change in the way it was doing things. Schedules began to shrink and quality improved. There was one PSP-trained developer and the code was kept under control. SILAC growth its expectations so that new developers should keep the quality and control of the code. Even though SILAC found PSP training very valuable, it needed team discipline and coordination. SILAC was also interested in achieving a process certification. One of the stated requirements for SILAC was to establish a process without lacking flexibility. That is having a process without excessive bureaucracy. SILAC found TSP as a natural step for achieving this.

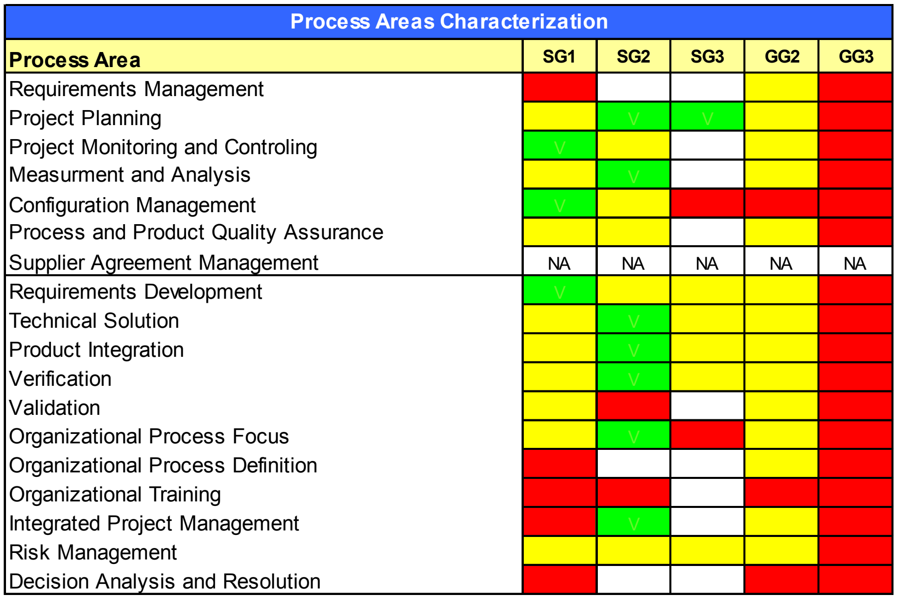
1.3 PROJECT STATUS

At the beginning of the AIM project, it was assumed that TSP process elements were fully deployed in the organization. There was a strong belief that software engineers and PG Members understood and used TSP and PSP processes (scripts, forms, and role descriptions); therefore, the selected strategy was to build a plan based on the SEI TSP to CMMI mapping [McHale 2005].

Regarding personnel skills, some engineers were new to the organization, with less than two months in the job. Some experienced engineers had PSP training, but most of them had not completed the PSP for Engineers course. Two members of the PG team had no software engineering background and the other two had it. Although PG members have been trained in TSP, they did not fully understand many of the technical vocabulary. There were English language barriers (most of the engineers speak very little English). Furthermore, it was a lack of documented guidelines - tools and policies- where engineers could map TSP concepts to their daily activities. Everything was well known only by two staff members who had the expertise in processes and tools. Unfortunately, they left this knowledge neither documented nor communicated.

Regarding the coaches, the project started with one internal TSP coach, one external TSP coach, one CMMI consultant and the SEI support for piloting TSPm. SILAC had an internal TSP Certified Coach, who offered coaching services to the development teams. He was the Coach for the software development team which was launched using the "TSP 2009.09" release two months after the PG team was launched. For the AIM project, SILAC internal coach was intended to assist TSP teams for pilots. A second external coach was also participating with the PG team. This coach started the project by launching the PG team in November 2009 with SEI assistance and he was responsible for guiding and coaching the process group during the project. The external coach remotely attended weekly PG meetings. Because of the existence of an internal coach in the organization, it was unclear, for some months, which one should provide post-launch coaching. As a result, the PG team was left almost alone. Coaches did not schedule a checkpoint during cycle one.

Two months after the PG launch in November 2009, the CMMI consultant did an informal SCAMPI B ML3 to get an idea of SILAC processes being used. The software development team and the process group were interviewed during 2 days. A major outcome was found: SILAC engineers did not understand several TSP process elements and the gap to fulfill CMMI ML3 appeared bigger than initially expected. Figure 2 shows SCAMPI B ML3 findings.



Process Areas Characterization					
Process Area	**SG1**	**SG2**	**SG3**	**GG2**	**GG3**
Requirements Management	🟥			🟨	🟥
Project Planning	🟨	🟩 V	🟩 V	🟨	🟥
Project Monitoring and Controling	🟩 V	🟨		🟨	🟥
Measurment and Analysis	🟨	🟩 V		🟨	🟥
Configuration Management	🟩 V	🟨	🟥	🟥	🟥
Process and Product Quality Assurance	🟨	🟨		🟨	🟥
Supplier Agreement Management	NA	NA	NA	NA	NA
Requirements Development	🟩 V	🟨	🟨	🟨	🟥
Technical Solution	🟨	🟩 V	🟨	🟨	🟥
Product Integration	🟨	🟩 V	🟨	🟨	🟥
Verification	🟨	🟩 V	🟨	🟨	🟥
Validation	🟨	🟥		🟨	🟥
Organizational Process Focus	🟨	🟩 V	🟥	🟨	🟥
Organizational Process Definition	🟥			🟨	🟥
Organizational Training	🟥	🟥		🟥	🟥
Integrated Project Management	🟥	🟩 V		🟨	🟥
Risk Management	🟨	🟨	🟨	🟨	🟥
Decision Analysis and Resolution	🟥			🟥	🟥

Figure 2: SCAMPI B ML3 findings

While TSP coaches did not have a deep understanding of the CMMI model, the CMMI consultant did not have any TSP knowledge. Therefore, CMMI consultant started training in December 2009 and it was decided that he would be the new PG coach after cycle one. The PG team was relaunched in February 2010 by the new external coach (the CMMI consultant). The new external coach assigned his effort to train the PG in both: CMMI process areas and managing organizational change concepts. The internal coach helped PG members to: use the dashboard tool, capture task data, and to consolidate calendars for weekly meetings. Although both coaches were attending PG weekly meetings, none of them provided enough post-launch coaching. As a natural consequence, PG members were having troubles in both: meeting weekly task hours and finishing assigned tasks.

Two months after the PG relaunch (April 2010, cycle 2), the new external coach decided to perform a checkpoint to the PG team. Results shows a calendar slip (as illustrated in Figure 3) due to the following factors:

- Misunderstanding of TSP: PG members deviated constantly from TSP scripts.
- Lack of Software Engineering background: Two of the original Sr. technical members left the process group and they were replaced by two junior-technical members with very little or no development experience.
- Not enough TSP post launch support to PG members: Coaches were not giving enough time.
- Poor management involvement: management did not request any formal project status. SEI neither checked coaches' activities nor checked project status.

The external coach recommendation was to relaunch. He also started providing the required post launch coaching to the PG members.

Monitoring problems in cycle one and two could have being mitigated with proper coaching and proper pilot support. Similarly, some PG performance problems could have being mitigated with proper coaching and proper training.

1.4 MANAGING THE PG

An important aspect of TSPm 2008.09 is the institution of a process group to handle software process improvement (SPI) activities at the organizational level. The PG is launched as a TSP team with the main purpose that SPI activities received the same rigor as any TSP team. TSPm also includes process elements for launching a process group as a TSP team and new roles for managing the process group. The new roles have responsibilities for creating and managing the organization's set of standard processes (OSSP), reporting process nonconformance issues, organizational training, tracking coaching activities, and other related issues. Having a process group is a good common practice for managing process improvement projects [McFeeley, 1996]; for instance, implementing CMMI or managing the organizational assets for TSP projects.

There are new roles for the process group: Coaching Manager, Process Asset and Data Repository Manager (PADRM), Training Manager. Additionally, there are new responsibilities for the Team Leader, Support Manager, and Quality Manager. Both quality and support managers provide an alternative reporting chain to management of non-compliance issues that are not being directly addressed and resolved by the team or team leader during operations. Support manager is the configuration management coordinator for the OSSP. PADRM is responsible for establishing and maintaining all aspects of the Project Notebook, which is stored as part of the OSSP.

The initial staffing of the PG included two people with nonsoftware management skills and two part-time technical people. During the first cycle, time was allocated to: become familiar with TSP process elements; provide training in process improvement and some CMMI process areas; and conduct an informal SCAMPI B with the software development team and the PG team. It was assumed that the PG team understood PSP, TSP and SILAC software development processes. Although the external coach participated in several weekly meetings, no post-launch training was provided to the PG members. The PG team had problems with the tool and members were not properly following TSP scripts. During preparation for the post mortem, the internal coach found that quality data and size data was poorly collected.

For the second cycle, there was no historical data for the quality plan and problems with gathering other project data prevented the team in having better estimates. The SCAMPI

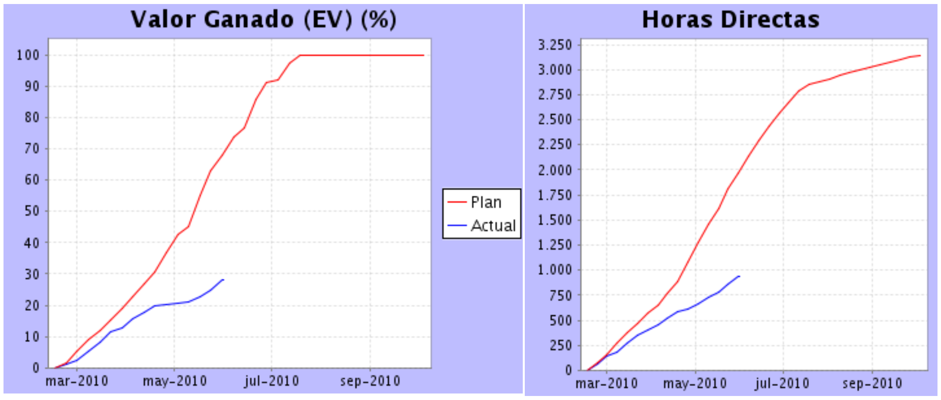


Figure 3: TC-AIM Earned Value and Effort

B findings were used to make effort estimates. Regarding the PG staffing, the two technical experts left the project and two junior technical staff members were assigned to the project. A new team leader for the PG was also assigned. The training for the process areas selected for this cycle was provided by the CMMI consultant. The project progress started to fall behind. The PG had problems developing the High Level Design (HLD) documentation to comply with CMMI process areas. The structure of the HLD document is described later on in this paper. Some HLD documents were redone several times and others got in limbo. The PG did not know SILAC software processes and they could not determine if the SCAMPI B findings were really missing processes in SILAC or just the lack of experience from the interviewed engineers. The PG was also getting behind their task hours. A checkpoint conducted by the new external coach made it clear that PG was having problems understanding PSP and TSP as well as SILAC software development processes. As a checkpoint recommendation, management provided a PSP review and a talk explaining how TSP is being applied and or modified in their software development processes. In addition, the SILAC technical expert was trained in the CMMI process areas covered in cycle 1 and 2. He also reviewed the SCAMPI B results and the design documents with PG members. Other corrective actions that resulted from the checkpoint were to relaunch cycle 2 and to provide post-launch coaching to the process group.

1.5 THE IMPLEMENTATION STRATEGY

The scope of the project is the implementation of CMMI ML3 processes using TSPm process elements as a foundation. Therefore, a guiding principle was established: missing processes should be developed, modified, or refined using the PSP/TSP philosophy and resources. A PG team was created to handle this SPI project and it should be managed as a TSP team. To address this project, four cycles were identified:

a.) Building the infrastructure: The first step was to provide basic training in process improvement projects and CMMI. The standards to define/modify process elements were established as well as naming conventions. A repository for the PG was created and the tool for publishing the OSSP was identified. A spreadsheet for handling launch forms was also developed.

b.) Process and project management process areas: The second step was to relaunch the operations team using TSPm scripts and the spreadsheet. The PG started the implementation of the Organizational Process Definition (OPD) and the Organizational Process Focus (OPF) Process Areas (PA) to help it manage the SPI project. The process to define and refine processes was documented and the priority for implementing processes was established. Similarly, the PG started the Organizational Training (OT) PA to handle training needs and register training records. Project management PAs (Project Planning –PP, project monitoring and controlling –PMC, and Risk Management –RSKM), needed small refinements. Integrated Project Management (IPM) PA helps TSP teams using and maintaining the OSSP. The Requirement Management (REQM) PA was reviewed and a tool was identified. For the support category, the Process and Product Quality Assurance (PPQA) and the Configuration Management areas were included in this cycle. PPQA was reviewed from both perspectives: as a process area and as a generic practice. In CM, the new TSPm scripts and forms were reviewed and some CM tools for handling the configuration management system were identified. In addition, TSP processes elements were captured in the OSSP tool.

c.) Engineering and advance project management process areas: The third step was to pilot new process elements from step 2 and to continue with the engineering and remaining process areas. These PA have major gaps in TSP implementations and require a major effort; therefore, they were left last to allow the PG mature their skills, infrastructure and communication channels. The Requirements Development (RD), Technical Solution (TS), Product Integration (PI), Verification (VER) and Validation (VAL) PAs are to be reviewed with the REQ, DEV, HLD, IMP, TEST1..3, and INS scripts. The scripts are to be updated to reflect SILAC operations and to add missing processes. The Decision Analysis and Resolution (DAR) PA helps TSP team members to take informed decisions of mayor project problems. The Measurement and Analysis (MA) PA helps developing indicators to fulfill project and organizational information needs. Some indicators needed to be documented and others to be developed. This cycle should finish covering all the PAs at CMMI ML3.

d.) Maturing and refining processes: The fourth step was to pilot new process elements from step 3 and to conduct both an SCAMPI B and a TSP Evaluation of the company. Areas of opportunity and Process Improvement Proposals (PIPs) resulting from both evaluations shall be addressed in this cycle so that a SCAMPI A can be conducted.

1.6 THE HIGH LEVEL DESIGN DOCUMENT

There is a need to identify and register required changes to TSP process elements as a result of fulfilling the gaps with CMMI ML3. It is also useful to document TSP-process-elements relationships with a CMMI process area as well as the relations among these process elements. A custom High Level Design (HLD) template was developed to capture these needs and the design rational. There must be a HLD document for each of the process areas at CMMI ML 3. The HLD template includes sections to specify: process-area practices and any finding registered during the informal SCAMPI B (CMMI gap analysis); a description of the association between a TSP process element within a TSP life cycle, a description of the association between a TSP process element (role, script, form, specification, guideline, checklist) and the process-area practices; a change description for each identified TSP process element;

a description for a new process element if need it; and a flow diagram describing relations among TSP process elements. Fulfilling this document allows people to identify relations considering several points of views and to attain a major understanding of TSP process elements and CMMI process area objectives.

1.7 LESSONS LEARNED

In order to be successful with an AIM implementation the following aspects should be considered:

Process group membership: it must include process people with experience in implementing software process improvement initiatives. Also, it must include expert-technical people that make use of company processes. They can provide value added amendments to current processes.

Process group training: members must have hands on experience in both TSP processes and company-software-development processes. Some members must have prior software development experience. Members must have being trained in the CMMI process model. Additionally, training in managing software process improvement projects is highly recommended.

Coach training: it is highly recommended that the process-group coach has previous hands-on-experience in coaching TSP teams. Similarly, the process-group coach should have previous hands-on-experience implementing CMMI process improvement projects. Having this mixed expertise allow keeping to the principles of TSP while both fulfilling CMMI requirements and adding value to the organization with the new process elements.

Handling internal and external coaches: Coaching responsibilities must be explicit and documented at the beginning of the project. Coaches should have frequent status meetings. Cross checkpoints among TSP teams should also be considered.

AIM Implementation: A TSP evaluation and a SCAMPI B shall be conducted as first steps of the project. These assessment findings provide a real and current status of the organizational-process use as well as the bases to estimate the effort to comply with both reference models: TSP and CMMI. In our case, it was assumed that TSP was fully implemented and project plans were made using this incorrect assumption. High Level Design (HLD) Document: this document addresses several important factors before you modify or add process elements: process needs or new process requirements; TSP life cycle phase where the new process need will be used, the process element where the new process need will be implemented, the relations among affected process elements, a change description on how process elements will be modified. Before the process group modify current process elements, the HLD document is a good mean to ensure that the process group understands both CMMI process needs and TSP current processes.

Post Launch Coaching: New process groups are exposed to PSP, TSP, CMMI, and process improvement concepts. If some members of the process group are new to any of these technologies, then they need a lot of support in using the tool; registering task, effort, and defect data; following roles responsibilities; balancing personal calendars; conducting weekly meetings; doing management status reports; conducting inspections; doing postmortems; interpreting project data; establishing and maintaining a process asset library; managing an organizational change; and understanding and interpreting the CMMI model and process areas.

AIM or TSPm support: SEI should provide implementation guidelines for these pilots as well as a close monitoring to avoid or mitigate potential problems. The essence of the TSP process elements should be documented so that CMMI people trying to understand the TSP philosophy can have a guiding document, avoiding misinterpretations and duplication of tasks when implementing AIM to reach CMMI ML3.

TSP and CMMI are complementary technologies that combined provide an amazing synergy. Combining both technologies is not an easy task, though. When practitioners of one technology try to assimilate the other technology, they can get confuse and may misunderstand the concepts and philosophies behind the technology. This issue may be true especially for TSP coaches or CMMI consultants trying for first time the other technology. Because these technologies are sometimes perceived as opponent technologies; proper objective support is need it to overcome this issue.

REFERENCES

McHale, Jim and Wall S., Daniel (2005). *Mapping TSP to CMMI* (CMU/SEI-2004-TR-0014), Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.

SCAMPI Upgrade Team (2006). *Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.2: Method Definition Document* (CMU/SEI-2006-HB-002), Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.

CMMI Product Team (2006). *CMMI for Development, Version 1.2* (CMU/SEI-2006-TR-008), Pittsburgh, PA: Carnegie Mellon University. Available at: http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr008.pdf

McFeeley, Bob (1996), *IDEAL: A User's Guide for Software Process Improvement* (CMU/SEI-96-HB-001), Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.

The Software Process Dashboard Initiative (2010). Downloaded in May 2010 at http://www.processdash.com/home.

BIOGRAPHY

Oscar A. Mondragón

Oscar A. Mondragon is a member of the technical staff at the SIECenter – ITESM Mexico since May 2004. Mondragon works as a sr. consultant for software quality models and process improvement methods. Mondragon is a SEI Certified instructor for the Intro to CMMI DEV, a PSP professional, and recently a TSP coach . In Latin American, Mondragon guides companies in their software process improvement projects using the CMMI model, develops SCAMPI B, prepares EPGs, and evaluates pilots. Before joining the SIECenter, Mondragon was as a research assistant at the System and Software Engineering Affinity Laboratory at the University of Texas at El Paso (UTEP), had two research appointments at the Real-time System Lab at the University of Pennsylvania and an internship at the AT&T labs at Middletown. Also, Mondragon was a research professor at ITESM where he customized seminars for Mexican companies in quality assurance, independent verification and validation, and software testing; tought undergraduate courses in Software Engineering; and did join research with the Electrical and Engineering department at UTEP. Dr. Mondragon holds a Ph.D. degree in Computer Engineering from UTEP and a Masters degree in Computer Science with a specialization in Software Engineering from the California State University at Sacramento. He did his master and doctoral studies as a Fulbright and CONACYT grantee, respectively. Dr. Mondragon is an IEEE Certified Software Developer Professional (CSDP) and a member of the core committee for CSDA/CSDP. Dr. Mondragon has published journals and articles in formal specification, verification & validation, requirements engineering, and process improvements in small settings.

Edgar D. Fernandez

Edgar D. Fernandez is SILAC co-founder and Operations Manager since May 2006. Fernandez is a PSP Certified Developer and TSP Certified Coach. As a developer, Fernandez was part of two important projects for Mexico's and Zacatecas' Government, using PHP, MySQL and Open Source platforms. Fernandez was part of Zacatecas' Government IT program, by participating in conferences and PSP/TSP diffusion. Fernandez is also BPoC for the SEI TSP Suite, and offers PSP/TSP services in Mexico and Latin America. Fernandez has also a Master Degree in Software Engineering by CIMAT. Fernandez participates, nowadays, as a TSP Coach for SILAC development teams and process mentor for the EPG, who is implementing TC-AIM method into the organization, and offers Team Leader and Coaching services for Quarksoft S. A., a mexican software company.

www.sei.cmu.edu/tspsymposium