

Carnegie Mellon University
Software Engineering Institute

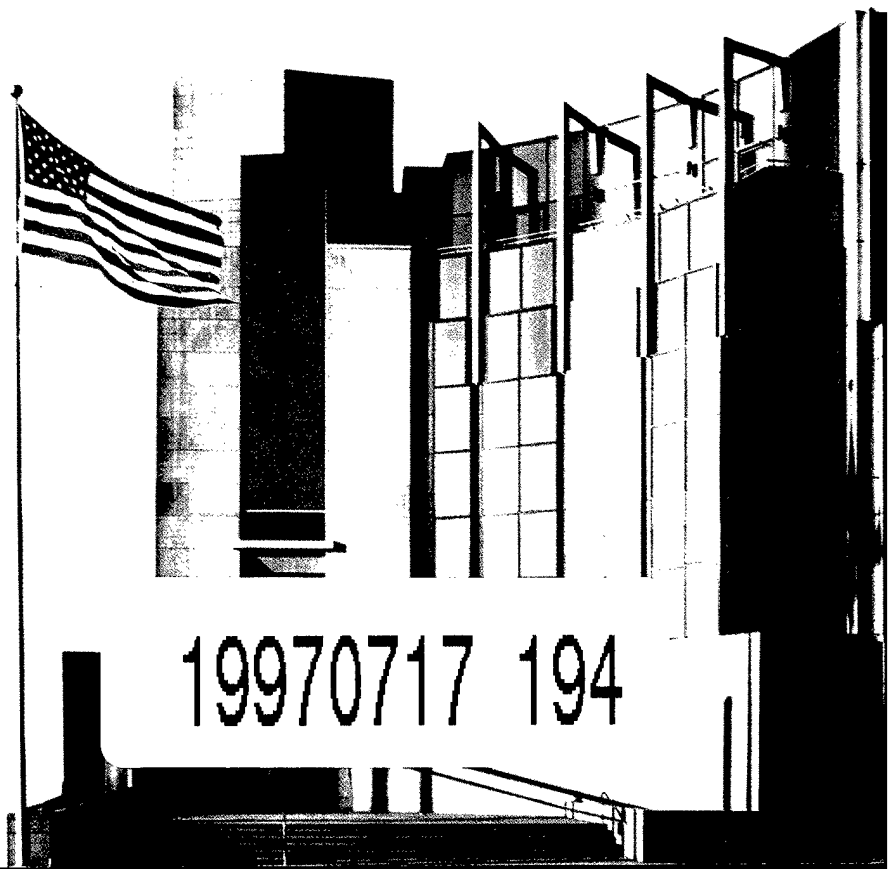
Radical Improvements Require Radical Actions: Simulating a High- Maturity Software Organization

Steven Burke
June 1997

TR

TECHNICAL REPORT
CMU/SEI-96-TR-024
ESC-TR-96-024

DTIC QUALITY INSPECTED 3



19970717 194

Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of, "Don't ask, don't tell, don't pursue," excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Technical Report

CMU/SEI-96-TR-024

ESC-TR-96-024

June 1997

Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization



Steven Burke

Resident Affiliate,
Computer Sciences Corporation

Process Program

[DTIC QUALITY INSPECTED 3]

Unlimited distribution subject to the copyright

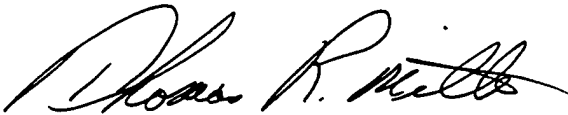
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/AXS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Copyright © 1997 by Carnegie Mellon University.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

Requests for permission to reproduce this document or to prepare derivative works of this document for external and commercial use should be addressed to the SEI Licensing Agent.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This work was created in the performance of Federal Government Contract Number F19628-95-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

This document is available through SAIC/ASSET: 1350 Earl L. Core Road; PO Box 3305; Morgantown, West Virginia 26505 / Phone: (304) 284-9000 / FAX: (304) 284-9001 / World Wide Web: <http://www.saic.com/contact.html> / e-mail: webmaster@cpqm.saic.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center / Attn: BRR / 8725 John J. Kingman Road / Suite 0944 / Ft. Belvoir, VA 22060-6218. Phone: (703) 767-8274 or toll-free in the U.S. — 1-800 225-3842).

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Table of Contents

Acknowledgments.....	vii
Executive Summary of Findings	ix
1. Introduction	1
2. Systems Thinking Background.....	5
3. Prior Published Findings	9
4. The Study's Methodology.....	11
4.1 Initial Model	11
4.2 Selecting an Organization and Relating It to the CMM.....	12
4.3 The Subjective Survey Needed to Determine Feedback Loops	13
4.4 Coordinating with Other High-Maturity Organizations	16
4.5 Developing, Testing, and Reporting the Simulation	16
5. Simulation Design Overview	19
5.1 Life-Cycle Process	19
5.2 The People Process	21
5.3 Processing of Maturity Level 4 and 5 KPAs	23
6. Findings	27
7. Validation of Results by SEL and Others.....	29
8. Conclusion.....	31
Appendix A: iThink Simulation Diagrams and Low-Level Design	33
Appendix B: Simulation Testing Design and Results	37
Appendix C: Test Output Tables.....	49
Appendix D: Definitions of Terms and Acronyms.....	55
References	57
Bibliography	59
SEL Reports.....	59
SEI Reports and Tools	59
Software Process Improvement	60

List of Tables

Table 1: Example Input and Output Values for the Starting Values and Two Combinations	39
Table 2: Value of Varying SUG and OUT	40
Table 3: Value of Various Attitudinal Mixes	40
Table 4: Value of Active Hiring of Pro-SPI People	41
Table 5: Results of Actively Hiring Pro-SPI Staff	44
Table 6: The Impact of Low ROI and How It Can Be Overcome	45
Table 7: Actively Replacing Cons Can Rapidly Overcome Bad Initial Mix	45
Table 8: Large KDLOC Case	46
Table 9: "Cut-Out" Cases	46
Table 10: Value of Varying ROI, SUG, and OUT	49
Table 11: Value of Various Attitudinal Mixes and the Value of Actively Hiring Pros	50
Table 12: More Cases of the Value of Various Attitudinal Mixes and the Value of Actively Hiring Pros	50
Table 13: Impact of Low ROI and How It Can Be Overcome	51
Table 14: Active Firing of Cons Can Rapidly Overcome Bad Initial Mix	52
Table 15: The Large Project Size Case	52
Table 16: The "Cut-Out" Cases	53

List of Figures

Figure 1. High-Level Diagram of SEL Feedback Relationships	17
Figure 2. Relative Size to Error Rate Relationship for 11 Projects	20
Figure 3. Relative Effort to Schedule Relationship for 11 Projects	20
Figure 4. Derived Relationship Between Pro/Con Ratio and Personality Mix Effectiveness	22
Figure 5. The Life-Cycle Subsystem	34
Figure 6. The People (Staff) Subsystem	35
Figure 7. The KPA Processing Subsystem	36
Figure 8. Test-Run Values for Effort, Errors, Schedule, and Size for the SEL Baseline Case	42
Figure 9. Flow of the Three Attitude Types Across Time for the SEL Baseline Case	42
Figure 10. Backlog of Pro-SPI New Hires Related to Maturity for SEL Baseline Case	43
Figure 11. Pros Increase Before High Maturity Achieved for Active Hiring	44
Figure 12. The Impact of the Cut-Out Case	47

Acknowledgments

I would like to deeply thank and acknowledge the tremendous support I've received from the NASA GSFC SEL. Sharon Waligora, Bill Decker, Steve Condon, Mari Sykes, Myrna Regardie, Cindy Brown, Warren Steger, Ray Luczak, Dick Wood, Frank McGarry, Dave Kistler, and Rick Coon provided access to technical reports and vast quantities of metrics, and gave their time to be part of the survey. Without them, this work would not have been accomplished.

I would also like to thank the SEI staff. Dave Zubrow, Marc Kellner, Will Hayes, Jim Hart, Betty Deimel, and Bill Hefley provided expert review and guidance of this research.

Finally, I would like to thank Professor Manny Lehman, of the Imperial College of London, for sharing his in-work papers on this topic.

If you have any questions about this report or would like further information, I can be contacted at

email: sburke@cscmail.csc.com

phone: (301) 794-1813

Executive Summary of Findings

Background

The following summary of findings is provided for those who may not be interested in the details of how the research was conducted or the details of the simulation design.

This report describes the methodology used to create a simulation of a high-maturity software organization and the results of this simulation. The approach of the study evolved as follows: The original intent was to simulate a mature organization with the activities of its cross-project, organization-level process improvement staff. The study would work backwards by "cutting out" the elements of the simulation that related to these high-maturity activities. The assumed regression in size, effort, schedule, and quality would be a reasonable indicator of the value of high maturity. As the study progressed, management "policy-control variables" arose that could also be simulated to reveal some of the organizational dynamics of a high-maturity organization. Finally, the results of the analyses were fed back to Computer Science Corporation (CSC) Software Engineering Lab (SEL) staff. This feedback triggered additional learning about high-maturity organizations that the simulation could not reveal by itself. The author realized that the goal of this study was not to design a perfect simulation right from the beginning, but to learn about high-maturity organizations.

The causal loop system of the National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC) SEL software life cycle and process improvement processes, as described below, served as a high-level basis for the simulation design. Size, effort, schedule, and quality metrics from 11 projects spanning over 10 years were used to derive equations and graphical relations between the following attributes: size to quality, size and quality to effort, and effort to schedule. The study used return on investment (ROI), or benefit, improvement data from carefully documented software process improvement (SPI) studies on reuse, cleanroom, and test process improvements.

Interviews with a cross section of personnel helped define the people elements of the simulation. An important people issue to remember is that there were three major attitude stances of the staff: people who supported process improvement (*pro-SPI*), people who were against process improvement (*con-SPI*), and people who didn't care (*no-cares*). The no-cares were modeled to be influenced by the pro-SPI or con-SPI people.

The following causal loop example is the basis of the simulation described in this report. It represents the software development and process improvement processes used by the NASA GSFC SEL.¹ The author of this report, a CSC employee, interfaced with CSC SEL personnel throughout this study to access the data, to conduct interviews of SPI and project personnel, and to review the results.

The SEL's high-level causal loop description of its software development and process improvement processes is as follows:

1. Major SPI efforts are piloted and deployed based on project cycle time (i.e., pilot on one cycle, and deploy on subsequent projects).
2. Major SPIs increase maturity (the probability of successfully achieving your goals).
3. Increased maturity attracts new hires and retains experienced staff that are "pro SPI" (i.e., they support and participate in SPI activities and are attracted to success and innovation).
4. Pro-SPI staff make minor SPI suggestions.
5. Major and minor SPIs decrease cycle time.
6. Decreased cycle time enables more major and minor SPIs to be accomplished.
7. Go back to 1 and repeat the cycle.

The hiring, reassignment, and turnover of staff are only a few means to control the aggregate attitudinal mix of the staff. The aggregate attitudinal mix of the staff has an impact on how many people adopt a SPI and, therefore, has an impact on the overall potential benefit of that SPI. Rewards, recognition, sanctions, communication, and support (tools, training, etc.) are other incentives to encourage staff to convert and/or adapt to the pro-SPI environment. The SEL process improvement staff managed most of these additional means. But hiring, reassignment, and turnover of staff, which were controlled by project managers and not SEL staff, were the means mentioned by the SEL as an important factor in maximizing the potential SPI benefit. They were also the only ones that could be reasonably quantified. The others were assumed to be constant across SPI efforts due to the careful planning and execution of process improvement activities by the SEL's process improvement staff. The key is to focus on managing the attitudinal mix of the staff.

¹ The SEL is a partnership among NASA Goddard Flight Dynamics Division, the University of Maryland Department of Computer Science, and Computer Sciences Corporation (CSC), NASA Goddard's primary contractor for building flight dynamics ground support systems. The SEL was founded in 1976 and is dedicated to software process and product improvements. The SEL has a mature software improvement program including an extensive measurement program. Although the SEL has never had a formal software capability evaluation, it has been cited as an optimizing organization, winning the first IEEE Computer Society Software Process Achievement Award in 1994 (McGarry and Pajerski 1994).

Variables Tested

Seven management "policy-control variables" were tested in various combinations to see what management decisions produced the most value. The values were based on SEL performance data and the SEL survey. Ranges above and below those SEL values were also selected. Only the seventh variable tested the "cut-out" case, which was the original intent of this study. The other six test project attributes, the process improvement process variables, and people issues.

The following management "policy-control variables" were tested (possible values for each variable are shown in parentheses):

1. Amount of return on investment (ROI) improvement (benefit) due to minor SPIs (small suggestions) on either size, schedule, effort, or quality (1%, 0.5% or 0.25%). That is, each suggestion can improve size, effort, etc., by 1%, 0.5%, or 0.25%.
2. Amount of SPI suggestions per pro-SPI staff member per project cycle (0.2 or 0.4 suggestions per pro-SPI person per project cycle). All suggestions made are also implemented. This is based on SEL project experience.
3. Percent of total suggestions that were lessons learned from other projects within the organization (Organization Sharing - Defect Prevention Program (DPP)). Possible values were 60%, 30%, or 0.
4. Initial aggregate attitudinal mix—the pro/con ratio (1.5, 1, 0.66 or 0.5). A value of 1.5 means that there were 1.5 times as many pro-SPI staff as con-SPI staff.
5. Personnel policies that set clear expectations about staff attitudes towards SPI. Either let increased maturity passively attract pro-SPI new hires or let the managers actively hire pro-SPI new hires even when starting at a low maturity.
6. Project size (100 or 150 KDLOC [thousand lines of developed code]).
7. "Cut-out" versus non "cut-out" cases. In the cut-out cases, projects accomplish SPI activities in isolation with no sharing of lessons. The non cut-out case has a fully staffed group that would accomplish organization-level SPI activities across projects using the Capability Maturity Modelsm (CMMsm) Maturity Level 4 and 5 key process area activities as a guide to pilot and deploy major SPIs across projects, extract lessons from projects and communicate them to other projects, and staff an organization-level metrics program. The SEL calls their staff the "Experience Factory" staff.

Findings

A baseline case was defined as the SEL performance when the values for the above variables were set at 0.5%, 0.2, 60%, 1.5, passive, 100 KDLOC, and non-cut out, respectively. When all of the significant combinations of the above 7 variables were run for a simulated period of 500 weeks (about 10 years), the simulation indicated the following major findings:

sm CMM and Capability Maturity Model are service marks of Carnegie Mellon University.

1. In the "cut-out" case, the value of going from CMM Level 3 (medium maturity) to Level 5 (high maturity), in the SEL context (*yours will be different*), resulted in potential improvements over a 10-year period in effort, error rate, schedule, and size of 215%, 135%, 192%, and 168% from the baseline case. The exact numbers are not important; what is important is that the numbers are very large. Furthermore, the difference between a low-maturity organization (that does no SPI efforts) and a high-maturity organization is much greater. Potential improvements over a 10-year period of 480%, 143%, 592%, and 290% for effort, error rate, schedule, and size, respectively, are possible in that case. These percentages can also be viewed as the cost of staying at low maturity.
2. For the "cut-out" case of highly motivated individuals (high SPI suggestion rate, high SPI ROI, good pro-SPI mix, etc.), the lost potential of not having the support of centralized organization-level process improvement (or Experience Factory) staff was greater than the "cut out" for the baseline case. Effort, error rate, schedule, and size could have been improved by 242%, 173%, ~2000%, and 240%, respectively, over a 10-year period. This contradicts some people's belief that using formal procedures inhibits their ability to excel. The simulation shows that good people lose *more* potential than mediocre people when they are not supported by the seemingly "formal" Level 4 or 5 KPA procedures, activities, and staff!
3. Over time, high maturity brings you to the point where your current challenges (100 KDLOC projects in the same domain) become less challenging in terms of effort, error rate, schedule, and size requirements. High maturity pushes and enables you to solve more complex problems and build more complex systems.
4. Minor SPI suggestion rates and the DPP variable (learning from other projects' mistakes) have a stronger influence on performance than the ROI for minor SPIs.
5. A small project size, which requires a small schedule, will show quicker success and produce more minor SPI suggestions than choosing a large project to begin major process improvement efforts.
6. Items 4 and 5 imply that the SEPG (or Experience Factory staff) needs to shift its focus from tailoring an organizational process for a project. The new focus should be global optimization of sharing lessons and SPIs across projects, designing a minor SPI suggestion process that encourages frequent suggestions from the technical staff, and choosing small projects to pilot major SPIs.
7. The initial pro/con ratio (attitudinal mix) is the control variable with the strongest influence. An initial unfavorable ratio can sink desires to achieve high maturity. You may need to assess your aggregate attitudinal mix before starting a large improvement effort. Active hiring (and/or conversion) of pro-SPI people before you start your drive for high maturity can help overcome an initial high-risk mix. Active hiring of pro-SPI people can accelerate achieving high maturity by four years as compared to the passive case. Active replacement of con-SPI and even "no-care" people can accelerate achieving high maturity by another whole year and a half or more from the active hiring case.

Radical improvements require radical actions. In true Systems Thinking fashion, the strongest relationships were not the intuitive issues of "How effective is the ROI of your SPI efforts?" or "How well are your procedures documented?", but "What is the aggregate attitudinal mix of your staff towards SPI?" and "How can you influence the aggregate attitudinal mix?" Achieving radical improvements requires a battle for the hearts and minds of the "no-cares" in order to maximize penetration, or adoption, of the process improvements. The ratio of the pro-SPI people, who inspire, to the con-SPI people, who impede, is crucial. The selective staffing findings should be abstracted up one level, to show management that hiring and replacement

are only two means to the end of actively managing the aggregate attitudinal mix. Addressing the cons, converting the no-cares, carefully selecting leaders, using turnover to your advantage, modifying rewards and recognition structures, communicating the SPI message, and funding appropriate support are other means to manage the attitudinal mix.

An important part of the learning process was the presentation of these simulation findings to key SEL personnel. The presentation triggered new learning that the simulation did not uncover, and that would not have been as well articulated if the simulation had not been done:

- Managers need to plan future projects so that the staff can use the new skills and technology brought on by process and technology maturity. If staff members are not able to use their new skills, they will go elsewhere.
- Although counterintuitive, it is the "no-cares" that managers should focus their attention on because they may change their attitudes more easily than the "cons." The pros and cons are mostly set in their ways.
- Team leaders have a strong influence on whether their staff attitudes are pro- or con-SPI. Turnover in the team leads can cause wild swings in the pro/con mix. In the subjective survey described later in this report, some people felt that if the senior manager did not support SPI, then the rest of the organization would not either.
- Cons are often heroes and opinion leaders that are looked up to by others. The cons do not think they are getting value from SPI and may perceive that they have a lot to lose. It is important to identify the cons and try to involve them in SPI because diversity can strengthen the end result. However, poorly planned SPI efforts may increase polarization of cons and pros.

Although SEL personnel agreed with the overall qualitative findings, some of the quantities used in the simulation were difficult to validate against reality. They pointed out that the study did not model the negative feedback due to reorganizations, budget cuts, and the resulting low morale. SEL personnel pointed out other detailed points. Although not verified yet, it is felt that these missing details will change the quantities of the simulation runs, but not the qualitative findings.

The author of this report can be contacted at
email: sburke@cscmail.csc.com
phone: (301) 794-1813

Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization

Abstract: This report describes the methodology used to create a simulation of a high-maturity software organization and the results of this simulation. The goal of this research was to find the quantitative value of improving from Capability Maturity Modelsm (CMMsm) Level 3 to Level 5. The method was to simulate a high-maturity organization using its actual empirical data and then "cut out" the high-maturity elements of the simulation. The resulting change in software size, effort, schedule, and quality would be a more accurate measure of the value of high maturity than working forward with a low- or medium-maturity organization and merely hypothesizing the activities and values of high maturity. The author used computer simulations based on Systems Thinking and Systems Dynamics, which reasonably modeled the "soft variables" of the people aspects of an organization (personnel attitudes, learning curve, participation in software process improvement, etc.). The simulation also related the soft variables to the "hard variables" of a software organization's life-cycle process (software size, effort, schedule, quality, etc.).

1. Introduction

What is a reasonable prediction of the future performance of your organization? What is the impact that your policy decisions have on the dynamic relationships of your organization, and therefore your organization's performance? What is the reasonable long-term value of achieving a high maturity level of the Software Capability Maturity Modelsm (CMMsm)? How does the aggregate mix of attitudes held by your technical staff affect your organization's maturity and performance levels?

"Systems Thinking" or "Systems Dynamics" computer simulations² can help you answer these questions. The simulations can also help you determine what policy decisions to make in order to improve performance and prevent blind tampering with your organization and its processes. Relatively inexpensive actions such as assessing your staff's attitude towards improvement activities and rewarding strong commitment to improvement while sanctioning indifference or defiance can have a surprisingly large impact on performance.

sm CMM and Capability Maturity Model are service marks of Carnegie Mellon University.

² An explanation of Systems Thinking, Systems Dynamics, and the iThink tool is given in Chapter 2.

Setting down expectations regarding employee commitment and taking appropriate human resource (HR) actions seems radical in our current "I'm OK, you're OK" politically correct world. But radical performance improvements require radical actions. Even if you spend resources to reengineer your processes or to continuously improve them, you will lose tremendous potential if a large majority of your staff (and managers) are apathetic or antagonistic towards improvement.

System Dynamics simulations can reasonably model the "soft variables" of the people aspects of an organization [personnel attitudes, learning curve, participation in software process improvement (SPI), etc.]. The simulation can also relate the soft variables to the "hard variables" of a software organization's life-cycle process (software size, effort, schedule, quality, etc.). A methodology used to create a simulation of a high-maturity software organization and the results of this simulation are reported in this paper.

The original goal of this research was to find the quantitative value of improving from SW-CMM Maturity Level 3 to level 5. The method was to simulate a high-maturity organization using its actual empirical data and work backwards by "cutting out" the high-maturity elements of the simulation. The change in software size, effort, schedule, and quality would be a more accurate measure of the value of high maturity than working forwards with a low- or medium-maturity organization and merely hypothesizing the activities and values of high maturity.

Some descriptions of low-, medium-, and high-maturity organizations may help. *Low-maturity organizations* are organizations with a low to zero probability of successfully achieving their goals (deliver on time, within cost, with quality). Their processes are unstable, probably undocumented, and probably not followed if they were documented. Low-maturity organizations depend on heroes to save the project. If the hero leaves, the project is in trouble—hence the instability. Major improvements are not planned, nor are they often successfully implemented. Staff participation in SPI is minimal to none as they are too busy resolving crises.

Medium-maturity organizations have a good probability of successfully achieving their cost, schedule, and quality goals even for projects in different application domains—due to the ability of tailoring organization-level processes to each project. They can repeat previous performance and are not dependent on heroes for success. However, they do not quantitatively improve their current processes in a systematic way. Process and performance improvement is still led and performed by some dedicated staff outside of the project technical staff.

High-maturity organizations spend the resources to have an infrastructure that supports the systematic, quantitative, and continual improvement of their products and processes. They have a high probability of not only successfully accomplishing project goals, but also successfully accomplishing performance improvement goals. Members of the project staff contribute and own process improvement efforts.

Other terms and acronyms used in this report are defined in Appendix D.

As the research progressed, new goals of determining the impact of various management-level policy decisions were discovered and then modeled. This is an important point. The value of conducting simulations is not just in answering original questions, but in discovering new questions to answer that you would not have even known to ask until you started the simulation effort. Even after the simulation, presenting the simulation's findings to the people in the organization that was simulated triggered new learning that the simulation did not consider.

2. Systems Thinking Background

A short background in Systems Thinking, adapted from the book, "The Fifth Discipline" is necessary (Senge 1990). Today's generation has seen an exponential increase in knowledge and competition. Just 25 years ago, the patent numbers issued by the U.S. Patent Office numbered in the 2 millions. That is, it took 200 years for the country to register 2 million different inventions or patents. Now patents number in the four millions. Just as many patents or inventions were registered in the last 25 years as there were in the prior 200 years. If an organization is to maintain a competitive advantage, it must increase its ability to learn faster than the competition. Systems Thinking can help us learn faster because it is a discipline for seeing wholes rather than parts and dynamic patterns of change rather than static snapshots. Systems Thinking treats the whole system, including the people involved, as both the cause and effect of observed phenomena. Systems Thinking describes the causes and effects in circular fashion rather than linear cause-effect diagrams. Systems have delays and feedback that are not obvious when linear thinking is used. A systemic or endogenous view changes our mindset from blaming outside forces and merely reacting to those forces. People are part of the cause and effect and can take responsibility and participate in the improvement of a system. Very small nonobvious forces often have the leverage needed to improve a system, rather than tamper with a seemingly obvious force and cause more problems.

Some examples may help to explain the concept of Systems Thinking. The Cold War arms race progressed as follows: The USSR had nuclear weapons, which was perceived as a threat to the U.S. The U.S., then, increased its weapons, which was perceived as a threat by the USSR, which increased its weapons, and the cycle continued. The cyclic system was the problem, not a linear solution of building a better weapon. The harder we push, the harder the system pushes back. This is called compensating feedback. To solve world hunger, wealthy countries provided agricultural assistance to poor countries, which reduced death by starvation, but increased the population. This, in turn, caused more starvation. Many companies are surprised when they experience rapid growth and success that exceeds their capacity to serve the high demand. So they react by spending capital to increase capacity, which takes a long time. This delay turns customers off, which rapidly decreases revenue and can cause bankruptcy if the company cannot make the payments for the capital invested. The question is not, "What factors influence performance?", but "What relationships generate performance?" Systems Thinking, using Systems Dynamics simulation—the simulation of Systems Thinking models with computers—can help people understand the true systemic behavior of an organization, learn faster, and take truly correct corrective action.

The following description of Systems Dynamics and iThink is taken from the iThink reference manuals (HPS 1994). Systems Dynamics and iThink (the Systems Dynamics tool used in this study) are concerned with the dynamics, or the change of state over time, of closed loop processes. Systems Dynamics and iThink base their fundamental, process-simulation building blocks on the following definition of process: "A process is a sequence of activities

through which material flows for purposes of undergoing some sort of transformation which adds value.” The essence of a process is flow. Therefore, a key question when creating an accurate simulation is, “What is flowing?” Over time, it has been observed that material does *not* flow much of the time. It is often waiting for some activity to be performed on it. Systems Dynamics and iThink, therefore, make a distinction between stocks (amount of resources) and flows (the rate of change of the stock). Flows directly control the transfer, consumption, and transformation of stocks and, therefore, directly affect the level or amount, of stock.

Stocks and flows are two of the three building blocks used by iThink to represent processes. The third building block, information feedback links, provides indirect control of the stocks and flows over time. Feedback is information, not the actual flow of resources, that is provided to the decision rules that control the amount of flow over time. To illustrate this concept, take cutting a bagel as an example. The bagel is the stock being transformed. Cutting the bagel with a knife at a certain rate (e.g., one inch per second) is the flow. Feedback is the form of information from your eyes to your brain that the bagel is cut in two. This information is used to affect the decision rule that controls the cutting rate. Without this feedback, you may cut off your finger.

People try to improve processes by using a static process map or diagram to analyze the resources (stocks) and activities (flows) in isolation. Their goal is to see which ones can be removed or rearranged to speed up the flows. Removing stocks and flows that do not add value is a typical example of such a process improvement methodology. Systems Dynamics and iThink help improve this methodology and help prevent tampering with a process by giving proper focus to the feedback and decision rules elements of a process. It has been found that people are able to analyze the static structure of a process fairly well, but they have difficulty analyzing the dynamics of a process. That is why computer simulation can be effective. The state of a process can be defined as the amount of stock at any given point in time.

The evaluation of the simulation runs described in this study is based on comparing the state of key resources (size, effort, schedule, and quality) at the end point of running a simulation, with all of its dynamics, for a 500-month period. For this study, size, effort, schedule, and quality are stocks. The flow of process improvement activities, whose timing and magnitude are controlled by various people and life-cycle feedback links, changes the amount of those stocks. The state of these stocks and the the process improvement activities also affect the people and life-cycle feedback links. Size, effort, schedule, and quality also have certain flows to each other. This shows the closed loop nature of the overall process.

The following causal loop example is the basis of the simulation described in this report. It will be expanded upon later. It represents the software development and process improvement processes used by the NASA Goddard Space Flight Center (GSFC) Software Engineering Lab (SEL):

1. Major software process improvement (SPI) efforts are piloted and deployed based on project cycle time (i.e., pilot on one project, tailor and deploy on the subsequent project).

2. Major SPIs increase maturity (the probability of successfully achieving your goals).
3. Increased maturity attracts new hires and retains experienced staff that are "pro SPI" (i.e., they support and participate in SPI activities and are attracted to success and innovation).
4. Pro-SPI staff make minor SPI suggestions.
5. Major and minor SPIs decrease cycle time.
6. Decreased cycle time enables more major and minor SPIs to be accomplished.
7. Go back to 1 and repeat the cycle.

3. Prior Published Findings

Systems Dynamics and Systems Thinking are nice ideas, but they need to be successful in matching real-world events if they are to have value. The following published literature describes the accomplishments of others who used System Dynamics to simulate a software organization. Systems Dynamics has been used to successfully model and match real-world events. Please see the Reference and Bibliography for more information.

- Abdel-Hamid and Madnick, in *Software Project Dynamics*, created a very detailed simulation of the NASA GSFC Software Engineering Lab. They modeled project estimation and how manager's decisions can affect actuals. The simulation runs came very close to actual data on several SEL projects (Abdel-Hamid and Madnick 1991).
- Johnson, in *IT Organization Flight Simulator*, modeled the value of making various improvements (training, metrics, technology, and infrastructure). The model is calibrated to COCOMO models and has an excellent training handbook to help managers and staff use her model as a starting point and branch off to make their own models (Johnson 1995).
- Yourdon, Rubin, and Johnson, in "With the SEI as My Co-Pilot," modeled the effects of an organization's progress as it climbed up the five CMM maturity levels to determine the value of high maturity. Using published data on actual improvements as a base for their model, they found that productivity doubled from Level 1 to 5, the time for the new-hire learning curve was reduced by a factor of 4, and productivity dips during the first transition levels were reduced at the later levels (i.e., the organization becomes resilient to change) (Yourdon, Rubin et al. 1994).
- Madachy, in "Process Modeling with System Dynamics," created a very detailed software life-cycle process that modeled the effects of various inspection process phenomena. Simulation runs closely matched actual project data from Litton Computer Services (Madachy 1996).
- Glickman and Kopcho, in "Bellcore's Experiences Using Abdel-Hamid's System Dynamics Model," successfully used their model to track actual project estimates and to evaluate subcontractors' estimation abilities (Glickman and Kopcho 1995).
- Other works by Tvedt, Nguyen, Hansen, and Rubin also show the use of System Dynamics simulations in assisting managers to make decisions that will not backfire (Tvedt and Collofello 1995), (Nguyen and Ahmed 1995), (Hansen 1996), (Rubin 1996).

Most of the above published studies used actual industry data as a base for their models. Changes in the model, reflecting various management decisions, were then run to find at least the qualitative, or relative, value of those decisions as compared to each other. The study described in this report uses the same strategy. Some of the studies [most notably (Abdel-Hamid and Madnick 1991), (Madachy 1996), and (Glickman and Kopcho 1995)] used Systems Dynamics simulations that very closely matched quantitative values of actual projects.

4. The Study's Methodology

4.1 Initial Model

The original goal was to determine the value of high-maturity (CMM Level 4 and 5) activities. It was assumed that high-maturity organizations would have the quantitative data needed for a Systems Dynamics simulation. Simulating a high-maturity organization by using their real data and working backwards was expected to provide more valid results than simulating a lower maturity organization and working forwards by hypothesizing what the future may be. With this strategy, research was performed to determine how high-maturity activities affected the software life-cycle process, people, and each other. An initial model had each of the CMM Level 4 and 5 KPAs as entities (separate subsystems to simulate) that related to each other, the life-cycle process entity, a management entity, and a staff entity. Once these impacts and relationships were defined, the type of metrics needed for the simulation was defined. Also, a subjective survey was created to determine the "soft" variables that involved people dynamics.

In defining the metrics, we decided to collect data on how the metric value changed over the life of the project, in addition to the final value of the metric, in order to capture the dynamic nature of the metric. For example, we would want not only final total size of a product (e.g., total lines of code), but also how that total value changed during the project. The same applied to effort and quality metrics. We also thought that a grouping of the product into pieces broken up by the completion dates for development and testing was necessary. That is, for a 10,000 line of code (LOC) product, we asked: "What were the quality, effort, and schedule metrics for the first 1,000 LOC complete, then the second, third, etc.?" This allowed us to see if the first pieces had relatively large quality, effort, and schedule values as compared to the final pieces. If the final pieces were rushed or had benefited from a learning curve, their resulting quality, effort, and schedule values would be markedly different from the first pieces.

Very detailed metrics tables were derived from analyzing previously published simulations and various software metric programs. Size was to be measured in whatever units the organization wanted (LOC, function points, etc.). Size was to be grouped into 10 (or whatever number of) groups ordered by size (the average size of the largest 10% of modules, etc.). Also, size was to be ordered by the completion date for testing (size of the product for the first 10% of the schedule, second 10%, etc.). Quality, effort, and schedule data were requested for each life-cycle phase for each of the 10 (or whatever) pieces. For example, data were collected to answer the following questions: How much effort was spent in design, code, etc., for the first piece of the product; the second; etc.? How many elapsed days were spent in design for the first piece, etc.? How many defects were found in the design phase for the first piece, etc.? The intent was to see how these values changed over the time of the project as opposed to using just the final values.

As might be expected, these metrics requirements turned out to be too detailed for most organizations to keep and were also not really needed for the simulation. However, the organization chosen to be simulated did have excellent weekly size, effort, and quality metrics for 11 very similar projects that spanned 10 years. It took about one entire person-month to research the published uses of various metrics; relate those to the high-level model of how high-maturity activities affect life cycle, management, people, and each other; and create the subjective survey. This month of time was not considered a waste because it was still important to gain the understanding of how detailed metrics relate to the various processes. More discussion on how the metrics were actually used to model the dynamic changes of value over time is provided in Chapter 5 (Simulation Design Overview).

4.2 Selecting an Organization and Relating It to the CMM

After the high-level model and metrics requirements were defined, an organization (or organizations) needed to be selected to model. At first, all of the approximately 10 organizations known to be at CMM Level 4 or 5 were targeted to be asked to participate. But experience in prior studies and tasks showed that most organizations would not provide the data at the above required level of detail without establishing a strong personal relationship. Also, the six to eight months allotted for the task was only enough time to decently model one organization. Therefore, we chose the SEL because they fit the model of a mature organization as described earlier in Chapter 2 (Systems Thinking Background). They had an excellent archive of detailed *weekly* size, effort, quality, and schedule metrics for projects in the same domain spanning a 15-year period.

A quick overview of how the SEL operates and how the SEL activities relate to the CMM is necessary. The SEL uses a different process improvement mechanism than the CMM, which it calls the "Experience Factory." In a subsequent section, we describe other high-maturity organizations that used the CMM to achieve different goals. The SEL's goal is to successfully implement technological and process *change* to improve their products. They have a staff, separate from the project staff, who work on the adoption of change. They support the use of metrics, analyze those metrics, and facilitate the piloting, analysis, and deploying of major technological and/or process change *across projects*. The SEL does not continually work on the CMM Level 4 and 5 KPA activities for the activities' own sake. The SEL performs experiments. These experiments can be mapped to Level 4 and 5 KPA activities as follows:

- The SEL has an extensive metrics infrastructure to support successful implementation of change. This can correspond to the CMM Level 4 Software Quality Management and Quantitative Process Management KPAs.
- The SEL used the Experience Factory paradigm to implement the major use of Ada and object-oriented design (OOD). This can correspond to the CMM level 5 Technology Change Management KPA since they represent major technological changes used in the software life-cycle process.
- The SEL used the Experience Factory paradigm to implement the major use of cleanroom technology and to combine system test and acceptance test into one phase. These can

correspond to the CMM Level 5 Process Change Management KPA. The Ada and OOD activity also resulted in major process changes.

- Members of the SEL Experience Factory staff extract and communicate minor SPI lessons across projects. This can correspond to the CMM Level 5 Defect Prevention KPA.

The original intent of the study was to determine the value of high CMM maturity. The SEL, however, did not pattern their Experience Factory directly after the CMM. Therefore, this study shows the value of high maturity in general, the value of the Experience Factory in particular, and how the Experience Factory can relate to the CMM in a very generalized sense.

4.3 The Subjective Survey Needed to Determine Feedback Loops

The author of this report visited the SEL site to meet with SEL personnel and collect data. Weekly metric data were collected for 11 similar projects spanning a 10-year period. In addition, reports describing major experiments during that period and their benefits were collected. The third, most critical component of the data-gathering phase was to create and conduct a subjective survey that would uncover the "soft" variable relationships and best-guess values for items typically not measured (e.g., people issues). The survey was to find out if the following items were really important to process improvement:

- What impact and characteristics did turnover have on process improvement? Did change in the staff's attitudinal mix caused by this turnover have an impact on performance?
- Did managers' support and decision-making abilities in the area of process improvement have an impact on turnover?
- What caused people to participate or not participate in process improvement? Do staff attitudes, performance of the QA/SEPG staff in implementing process improvement suggestions, or outside factors (customer or new technology) affect participation in process improvement?
- How much process improvement work would continue by technical staff volunteers if the QA/SEPG staffing of formal process improvement activities were eliminated?
- How much do people benefit from learning from other peoples' lessons? How much has process improvement helped management?
- Do certain maturity levels have inertia or momentum? Does entropy increase when formal staffing of process improvement is cut?

The survey was given to managers, QA/SEPG staff, and technical staff. All of the people surveyed from all groups were also interviewed. For the most part, their answers were consistent with each other despite the different job roles. This survey validated some of the assumptions of the initial model, while it invalidated other important assumptions. For example, the role of management did not warrant the modeling of management as a separate entity. The survey validated and corrected the early design of the simulation before potentially incorrect (and therefore time-wasting) detailed design and "coding" activities took place. A major theme of this report is that the goal of Systems Thinking is to learn more, faster. The simulation is just

a part of the learning process. This survey (and the presentation of the simulation results) were also important parts of the learning process. The following interesting findings describe the organizational dynamics of a mature software development organization as they implement process and technology improvements.

All of the people interviewed strongly agreed that organizational process improvement activities must have a separate staff to work on those issues. Project technical staff may internalize process improvement only within the confines of their project. Cross-project deployment of process improvement must have its own staff or it will not get done.

In determining the impacts of turnover, achieving Maturity Level 3 (not 4 or 5) is sufficient to sustain process maturity in spite of high turnover. That is, if "mature people" left, the process would still survive in its mature state and not necessarily deteriorate *at Maturity Level 3*. Goldenson and Herbsleb describe the results of a survey of other companies, and independently confirmed that turnover had little impact on process maturity (Goldenson and Herbsleb 1995). However, maintaining Level 4 and 5 performance is like a having a fine-tuned, high-performance engine: it takes only a little dirt in the engine to cause degradation. Disruptions like reorganizations or budget cuts (layoffs) may cause a regression to Level 3. It was also noted that as "mature people" left, the organization lost the ability to explain the rationale, or the "whys" behind the process. The "whats" and "hows," on the other hand, were documented in procedures. Management can use this lesson to help sustain the ability to explain the "whys" of SPI.

Although the organization was robust enough to sustain high turnover of technical staff and middle managers, if a senior manager was replaced with a person who did not support process improvement, then the organization would also stop supporting process improvement. The senior manager or executive has a generally disproportionate impact on the success of process improvement.

As the organization progressed in its process improvement efforts, turnover decreased from about 10% a year to about 2 to 3% a year as people began to like the new environment. The people interviewed stated that pro-process improvement staff did not leave if a new manager came in who did not support process improvement, but it made their jobs harder. So, process improvement helped people stay, but regression in process improvement did not cause people to leave.

With one exception, turnover was caused by reasons unrelated to process improvement: shorter commute to work, slight increase in pay, budget cuts, etc. The exception is that if managers do not plan for their staffs to use the new skills obtained in a high-maturity organization (e.g., learning major technology changes like OOD, Ada, reuse, etc.), there will be an increase in turnover because the people will go somewhere else to use those skills.

Staff attitudes, which the initial model did not consider, were very important according to those interviewed. One interviewee said that Maturity Level 3 attracted people who liked to be led, to have rules defined, and therefore liked to have the documented procedures that Maturity

Level 3 often requires. Maturity Levels 4 and 5 attracted people with a different attitude: people who liked to have change, to learn new things and new methods, and to actively participate in change. One interviewee said it might not be a bad thing to have turnover before setting out to achieve higher maturity levels. Another interviewee stated that high maturity did not help teach new people faster; Maturity Level 3 did that. Higher maturity attracted *fast learners*, so that the net effect was the same.

The survey revealed that the management section of the initial proposed model design was not as large as originally thought. Managers basically need to support process improvement and use the metrics of Maturity Level 4 to make wise, precise decisions on implementing the major changes administered in Maturity Level 5. Managers did state that the use of metrics for project estimation and for making other types of decisions can help their accuracy by as much as 50%. Most of the interviewees stated that the manager's *technical skills* were more important than the manager's knowledge of process improvement techniques.

The survey revealed the following important features of the process improvement process: A major SPI is piloted on one project for its full cycle, analyzed, then deployed to the other projects in their cycles. That is, it takes one whole project cycle before all projects begin to use a major process improvement. This makes the simulation easy, yet interesting. Instead of trying to find out the actual calendar time it takes to roll out a process improvement, the simulation just needs to track the pilot-and-deploy sequence to cycle time. An important feedback loop is created. A major SPI is piloted and deployed, which decreases cycle time because of the improvement. This enables more SPIs to be implemented in the same fixed amount of time, and so on.

Also, interviewees stated that the learning curve for a new hire to be considered experienced is one project cycle. A new employee must work on a complete project to be considered experienced. So, we have another feedback loop. If major SPIs decrease cycle time, then the learning curve time is also decreased. But major SPIs increase maturity, which attracts people who support process improvement and who are fast learners. These people make even more major and minor (intra-project) SPI suggestions, which decrease cycle time even further, and repeats the cycle.

Different interviewees stated that as much as 60% of total minor SPI suggestions that they implement in their project were lessons learned from other projects. This can be interpreted as the value of having an organization-wide defect prevention program where intra-project lessons are extracted from a QA/SEPG staff and communicated to the other projects.

Some interesting miscellaneous items revealed in the survey were:

- Recognizing and rewarding local champions sometimes caused resentment by others and resistance to adopting the SPI efforts.
- When deploying major SPIs, the QA/SEPG staff needed to be sensitive to the different cultures that existed within each project.
- The first cleanroom SPI effort at CSC was strongly resisted by the technical staff. Then, one year later, the staff uncharacteristically embraced cleanroom methods with the

rationalization of, "Good, I don't want to do unit testing anyway." It may take a whole cycle to overcome the initial shock of a major change.

- It was hypothesized that as quality improved, the customer would change from reporting post-delivery defects to asking for major enhancements. That is, an increase in quality may result in an increase in total maintenance work due to delighted customers wanting more features. However, although enhancements made up the bulk of maintenance, the customer "backflipped" and stated that the quality was "too" good and that they wanted the products quicker, even with lower quality!

The tremendous value of the above findings clearly show the need, when designing a simulation, to conduct a subjective survey to complement the mere collection of metrics.

4.4 Coordinating with Other High-Maturity Organizations

The SEI hosted a CMM Level 4 and 5 Workshop to characterize high maturity based on real organizations in order to help update the CMM. Representatives from TRW, Raytheon, Motorola (India and U.S.), Citicorp (India), Lockheed Martin (the former IBM Federal Systems Division On-Board Space Shuttle Support Group), and other companies attended. This was an excellent opportunity to give the subjective survey described above to these representatives. From this survey, we found that each company used the CMM as a means to achieve totally different goals. The SEL's goal was to successfully implement change to improve products. One company used the CMM to determine what process activities were *not* needed so that they could make the most money. Another company used the CMM to determine what process activities were *always* needed so that they could maintain their reputation for quality, even if a particular customer wanted something fast and cheap with low quality. Another company used the CMM to estimate and maintain extremely high levels of software quality and reliability, yet had almost no tracking of effort. Other companies used the CMM to help individuals not let down the team; i.e., the CMM was used to ensure team success. *The point is that the true value of the CMM is that it increases your probability of successfully achieving your goals.* This is what it really means to be "mature." Each organization has different goals. Therefore, the quantitative results of this study should not be generalized or applied to other organizations. The qualitative results of determining the value of the CMM and of the use of system-dynamics simulations to help organizations learn more, faster can be used by many organizations. Organizations would then have to tailor the CMM and the system-dynamics simulations to their own goals.

4.5 Developing, Testing, and Reporting the Simulation

Once all of the metrics, reports, and surveys were collected from the SEL, the development of the simulation could begin. The simulation was broken up into three major subsystems: the people process, the life-cycle process, and the process improvement process (KPA processing). The simulation was iteratively developed, with each of the three subsystems comprising an iteration that was separately tested and validated before being connected to

the next subsystem. Complex simulations clearly need iterative development. Just like in regular software development, it is very costly to find an error late in the development process that could have been found in the earlier subsystem validation phase. Figure 1 shows a high-level diagram of the simulation and the key feedback relationships. The text description of these relationships was given in Chapter 2 (Systems Thinking Background). Note that in Figure 1, CT refers to cycle time.

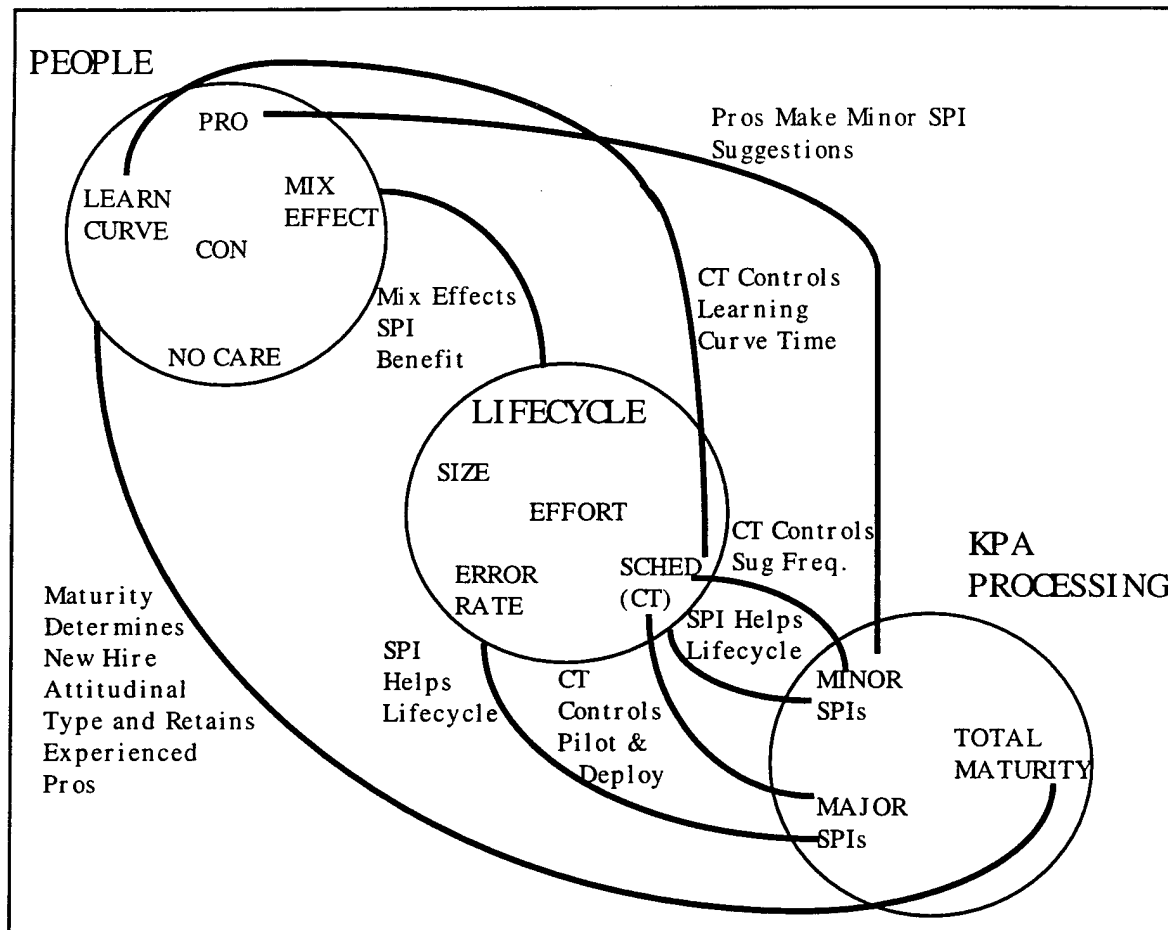


Figure 1. High-Level Diagram of SEL Feedback Relationships

Once the whole simulation was developed, all relevant combinations of the “policy control variables” were run to determine which combinations produced the best and worst values.³ The results were analyzed, presented to the SEL and others for review, and are reported

³ See the Variables Tested section in the Executive Summary or Appendix B for definitions of the policy control variables. These are different from the subsystem variables listed in Figure 1.

here. A more detailed description of the simulation design, testing, analysis, and reporting follows.

5. Simulation Design Overview

5.1 Life-Cycle Process

The life-cycle process modeled how software size, effort, quality, and schedule related to each other in order to produce a product. Those four attributes of the life cycle were quantities that could be changed by process improvements or by each other as one process improvement rippled through the four attributes. The SEL had excellent metrics for these four attributes. Based on 11 separate projects in the same application domain spanning 10 years, the following relationship existed between effort, size and quality:

$$\text{Effort (KHours)} = .153 (\text{KDLOC}) + 5.3 (\text{Development Defects/KDLOC}) - 4.68$$

Intuitively, effort is spent on doing something based on size and/or fixing something you did based on your error rate. This relationship had a correlation (r^2) of .97 and a significance probability of being only a chance relation of 1-p equal to much less than .005.

The SEL had experienced an interesting phenomenon. An increase in size not only increased total errors (as expected), but also increased error rate. You would expect the rate to be constant. It was believed reuse reduced the number of interface and integration errors by an exponential amount. If 100 modules are developed, the number of interfaces, and therefore the potential number of interface errors, is proportionate to the square of the number of modules (100*100 in this case). If reuse required only 50 modules to be developed, the number of potential interface defects is cut by one half squared, or one fourth. The SEL, upon an initial review of this finding, believes a different reason may be the cause. Basically, the smaller projects that had high reuse were done later in the 10-year time period. It was time that was the critical dimension, because better processes and a better, proactive staff were used in the later projects. The SEL believes these were the real reasons for the *cause* of reduced error rates; size is just a *correlation*, not a cause. This study reports on the results using size correlated to error rate as shown in Figure 2. In Figure 2, note that as size got very small, the error rate actually increased (error rate is shown in development errors/KDLOC). A plot of effort against error rate (not shown) revealed the same thing.

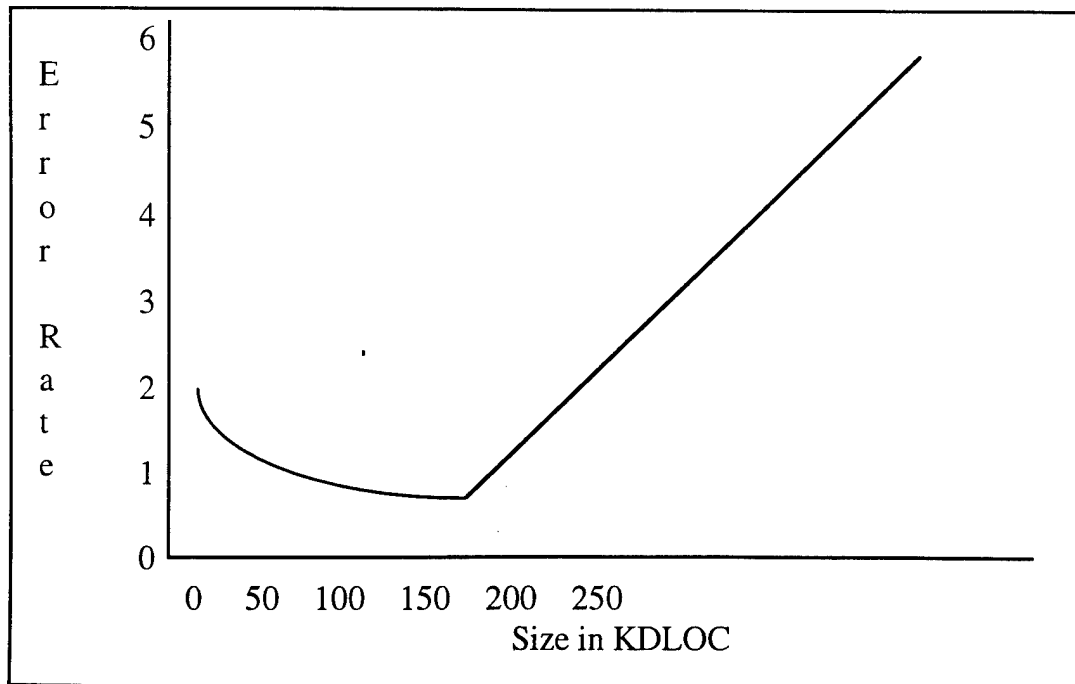


Figure 2. Relative Size to Error Rate Relationship for 11 Projects

The relationship between effort and schedule was also plotted using SEL data. The relationship shown in Figure 3 existed.

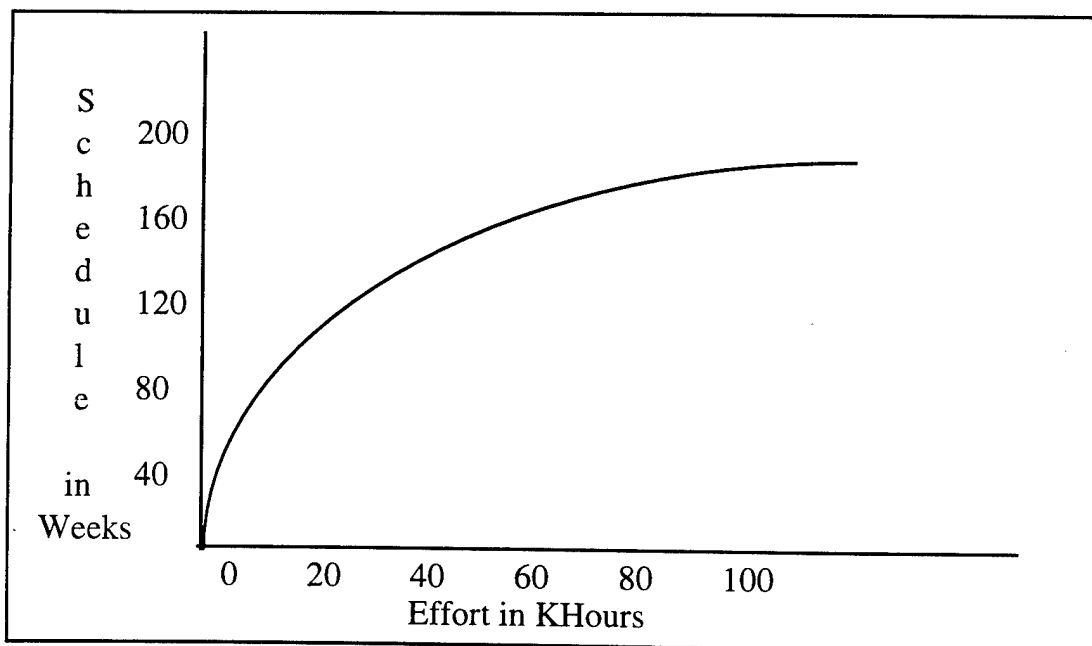


Figure 3. Relative Effort to Schedule Relationship for 11 Projects

With the above relationships derived from actual SEL life-cycle metrics, we can model the effect of a software process improvement. For example, if a major SPI, like reuse, reduces the amount of KDLOC needed to deliver the same amount of requirements, then effort is reduced according to the equation given above. Furthermore, since size can also reduce error rate according to Figure 2, effort can be reduced even further. Then, when effort is reduced, schedule is reduced according to Figure 3. SPI benefits are modeled as percent reductions in either size, effort, error rate or schedule. Major SPI efforts—such as reuse, cleanroom, and test process improvements—had their percent reductions published in SEL technical reports. These actual percents, along with the derivation of minor SPI percent reductions, are discussed later. It is important to discover the life-cycle relationships so that the full effect of a process or technology improvement can be modeled.

5.2 The People Process

The SEL said that there were three attitudes of staff that affected the potential benefit of process improvement: pro-SPI people, con-SPI people, and no-care people. Pros made almost all of the minor SPI suggestions. Based on a subjective survey conducted on an Ada/reuse major SPI effort, about 30% of the staff said they were pro-Ada, 20% were con-Ada, and 50% did not care. Both the pros and the cons were vocal in their opinions about this major improvement effort.

The attitudinal mix and the pro/con ratio can affect the overall potential benefit realized by a SPI effort. This was defined as the attitude impact. If there are more pros than cons, then more no-cares will also adopt the effort. This higher penetration and adoption will realize a higher overall benefit. If there are a lot of cons, then a lot of people may not adopt the SPI effort. Interviewees agreed that attitude affected SPI adoption and that staff members with strong attitudes affected other staff members' adoption of SPI.

In System Dynamics, finding the correct relationships among variables is more important than waiting for extensive empirically validated data. This is because the simulation can run a wide range of values to see what values are critical and if the overall results are insensitive to the particular values. The following "soft" variable relationship between pro/con ratio and personality mix effectiveness was derived using 30%, 20%, and 50% values:

1. At a 30/20 ratio, assume the 50% no-cares go along with the pros.
2. So 80% of the total staff are virtual pros. Since this was the documented case, we set their personality mix effectiveness to 1. That is, if a SPI effort claims to give a benefit of 50%, then the personality mix effectiveness given this personality mix is 50% times 1, or 50%.
3. If we want to find the effectiveness for a 25/25 pro/con ratio, then we can use the following: Assume the no-cares are also split since the pro/con ratio was even. If 80% virtual pros produced an effectiveness of 1, then having 50% virtual pros produces an effectiveness of $50\%/80\% = 0.625$.
4. If we have a 20/30 pro/con ratio, assume the no-cares are virtual cons (poisoned). So having only 20% pros produces an effectiveness of $20/80 = 0.25$. Figure 4 shows the

relationship between the pro/con ratio and SPI effectiveness for a range of values.

Personality Mix Effectiveness

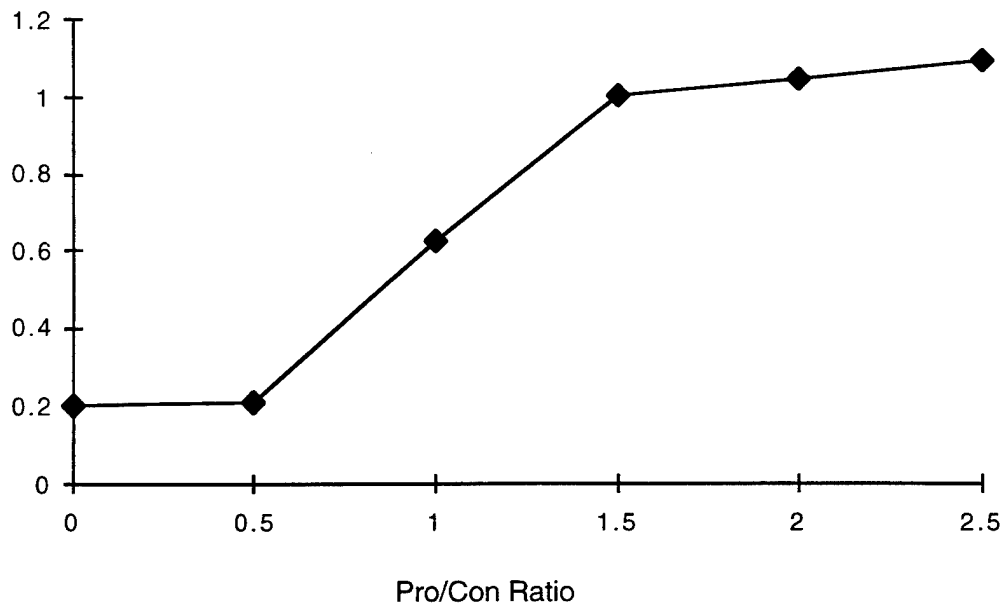


Figure 4. Derived Relationship Between Pro/Con Ratio and Personality Mix Effectiveness

In other people issues, the SEL stated that it takes one project cycle for a new hire to become considered experienced. That is, the cycle time is the learning curve. The more SPIs that are implemented, the shorter the cycle time since size and quality SPIs can reduce effort, and reduced effort reduces schedule. Therefore, the shorter the cycle time, the shorter the learning curve time. Also, increased maturity attracted more pro-SPI new hires and retained experienced pro-SPI personnel. The number of cons gradually decreased. So a number of important feedback loops are revealed. Increased SPI maturity increases the number of pros and decreases the number of cons. More pros make more SPI suggestions, which reduces cycle time and increases maturity. Also, the increased pro/con ratio increases SPI effectiveness, which reduces potential cycle time even further and enables more SPIs to be implemented. These all feed each other and can start a positive momentum in process improvement.

The above scenario is considered a passive pro-SPI hiring case. The increased maturity passively attracted pro-SPI new hires and retained more experienced pro-SPI staff basically by word of mouth. However, we can test out the effects of an active pro-SPI hiring case. Managers can actively hire pro-SPI personnel even if the current maturity is low, rather than wait for high maturity to passively attract pro-SPI staff. Test results are given later.

For a final people issue, the SEL reported that turnover (quits) started at about 10% per year, then decreased to 2-3% as maturity increased, then increased to about 6% toward the end of

the 10-year period. This final increase was due to a number of reasons. The most important from a process improvement perspective was that many of the future projects did not require the use of the new skills learned from the major SPIs. People with these new marketable skills (Ada, OO, reuse, etc.) left to find jobs that used those skills. Other important factors, such as severe budget cuts, which caused reassignment of personnel, caused a decrease in morale, which also resulted in quits.

5.3 Processing of Maturity Level 4 and 5 KPAs

This subsystem models the timing of the flow of process improvements into the life-cycle and people subsystems. There are two types of SPIs: major and minor. Major SPIs are managed by the Experience Factory staff (QA/SEPG/metrics type of personnel). The simulation models five separate projects running in parallel. The total organizational impact of a major SPI requires two project cycles. On the first cycle, the SPI is piloted on 1 project and, therefore, the impact is 20% of the total reported benefit. On the next cycle, the SPI is deployed to the other 4 projects to realize the remaining 80% of the total reported benefit. The following three major SPIs, as reported by the SEL, were used:

1. The Ada/Reuse/OO major SPI realized a potential benefit of reducing size by 55% yet providing roughly the same requirements (Waligora, Baily et al. 1995). This value of 55% was taken by averaging the percent reuse for the "low reuse" projects, averaging the percent reuse by the "high reuse" projects, and then taking the difference. The percent reuse was defined as follows: Find total lines of developed plus reused code (in KSLOC). Find total lines of developed code plus a 20% factor times black box reused modules (in KDLOC). Then, take the difference between these KSLOC and KDLOC values and then divide that difference by total KSLOC.
2. The cleanroom major SPI reduced error rates by 32% from the baseline project to the first-generation cleanroom project. Subsequent cleanroom projects also had high reuse along with using cleanroom. High reuse was a second variable besides cleanroom that affected error rates. Therefore, the reduced error rate values reported by the high-reuse projects could not be used due to having two variables (cleanroom and reuse) and only one known (Pajerski 1995).
3. The improvement reported by combining the system test and acceptance test phases reduced effort by 35% for the attitude ground system software domain (which is the domain of software used in this report) (Waligora and Coon 1995).

The most difficult part of the SEL process to simulate was the process for suggesting minor SPIs. This is because it was the least documented. According to the SEL, each major SPI spawned at least two minor SPIs. Minor SPIs were generated either by the pro-SPI people within one of the five projects (intra-project SPIs) or received as a lesson from one of the outside four projects (inter-project SPIs). Inter-project SPI "lessons" were extracted by the Experience Factory staff and communicated to the other projects. Different people stated that as many as 60% of minor SPIs implemented by a project are SPIs originating from other projects. This can be construed as part of the value of the defect prevention program. The number of intra-project SPIs was based on the number of pro-SPI staff (since only pros made suggestions), the organization's maturity (since low maturity inhibited pros from

participating), and an estimated number of suggestions per project cycle. The last of these three variables could be set to arbitrary values, such as one suggestion per project cycle or two suggestions, etc. The first two variables were determined by the whole system.

The value, in percents, of each minor SPI had to be derived. It was first arbitrarily set to 1% since it did not seem unreasonable for a minor SPI to save 100 person-hours in a 10,000 person-hour project. However, one SEL project did report that effort was reduced by 12% due to the combined staff making 1 suggestion per week for a 1-year time period. Using the compound interest formula to find what "interest rate" is needed to reduce a principle by 12% we have:

$$(1 - X\%)^{50 \text{ weeks}} = 100\% - 12\% = 88\%$$

$$X\% = 0.25\%$$

That is, this particular project made 1 SPI suggestion per week, each of which had an average benefit of 0.25%, for a 50-week period. The variable for the benefit (ROI) of a SPI was tested at values of 1%, 0.5%, and 0.25%.

The simulation was designed so that the percent improvement for a SPI suggestion will act on the current value of a variable like effort (the principal). For example, if the initial effort was 100, a 1 percent improvement in the first week would reduce the principal to 99. The new value of 99 would then be used for the second week. A second 1% improvement applied on the 99 would be 0.99, leaving a value of 99.01 for use in the third week, and so on. The percent improvement of a particular process improvement is incorporated in a discrete way at one time.

This particular project (with the 12% improvement) helped define the maturity "soft" variable. Maturity was not quantitatively measured by the SEL, but had to be quantified for the simulation. It was defined as being a composite of major SPI maturity and minor SPI maturity. Each of these was set on a scale of zero to one. Major SPI maturity reflected the ability of the organization to successfully pilot and deploy one major SPI. Since it piloted the SPI to one out of the five projects on the first project cycle, the value of major SPI maturity could only go from 0 to 0.2 for that period. For the second project cycle, the SPI was deployed to the other four projects so the value of major SPI maturity ranged from 0.2 to 1. Minor SPI maturity was different. The basic premise was that an organization was not considered mature until the project staff actively participated in process improvement. That is, an external staff, such as the Experience Factory staff, can impose a major SPI on a resistant project staff. But if the project staff does not participate in continuous process improvement, the organization should not be considered mature. It was decided that when the frequency of making minor SPI suggestions approaches one per week for a sustained period of half a project cycle, then the minor SPI maturity reaches its highest value (of 1). These decisions may seem arbitrary, but their content represents the spirit of the CMM. The spirit of the CMM is to have all of the people participate in technological and process improvement as the best means to successfully achieve your goals. This maturity variable does not directly map to the five

levels of the CMM, but uses adoption of SPIs and suggestions as surrogates for maturity levels.

The ability to derive "soft" variables like maturity into a quantity needed by the simulation is crucial for the success of the simulation. In this case, maturity was broken up into two scaled variables that were correlated to observable quantities (piloting and deploying major SPIs and minor SPI suggestion frequency). Appendix A shows the actual simulation coded in the iThink application. Appendix B describes the test design and quantitative results of particular simulation runs. Appendix C describes the quantitative test results for most other simulation runs with comments that give an interpretation of the results. The major findings of the simulation are described in the next chapter.

6. Findings

When all of the significant combinations of the 7 variables (described in the Variables Tested section of the Executive Summary and Appendix B) were run for a simulated period of 500 weeks (about 10 years), the simulation revealed the following major findings:

1. In the "cut-out" case, the value of going from CMM Level 3 (medium maturity) to Level 5 (high maturity), in the SEL context (*yours will be different*), resulted in potential improvements over a 10-year period in effort, error rate, schedule, and size of 215%, 135%, 192%, and 168%, respectively, from the baseline case. The exact numbers are not important; what is important is that the numbers are very large. Furthermore, the difference between a low-maturity organization (that does no SPI efforts) and a high-maturity organization is much greater. Potential improvements over a 10-year period of 480%, 143%, 592% and 290% for effort, error rate, schedule, and size, respectively, are possible in that case. These percentages can also be viewed as the cost of staying at low maturity.
2. For the "cut-out" case of highly motivated individuals (high SPI suggestion rate, high SPI ROI, good pro-SPI mix, etc.), the lost potential of not having the support of centralized organization-level process improvement (or Experience Factory) staff was greater than the "cut out" for the baseline case. Effort, error rate, schedule, and size could have been improved by 242%, 173%, ~2000%, and 240%, respectively, over a 10-year period. This contradicts some people's belief that using formal procedures will inhibit their ability to excel. The simulation shows that good people lose *more* potential than mediocre people when they are not supported by the seemingly "formal" Level 4 and 5 KPA procedures, activities, and staff!
3. Over time, high maturity brings you to the point where your current challenges (100 KDLOC projects in the same domain) become trivial in terms of effort, error rate, schedule, and size requirements. High maturity pushes and enables you to solve more complex problems and build more complex systems.
4. Minor SPI suggestion rates and the DPP variable (learning from other projects' mistakes) have a stronger influence on performance than the ROI for minor SPIs.
5. A small project size, which requires a small schedule, will show quicker success and produce more minor SPI suggestions than choosing a large project to begin major process improvement efforts.
6. Items 4 and 5 imply that the SEPG (or Experience Factory staff) needs to shift its focus from tailoring an organizational process for a project. The new focus should be global optimization of sharing lessons and SPIs across projects, designing a minor SPI suggestion process that encourages frequent suggestions from the technical staff, and choosing small projects to pilot major SPIs.
7. The initial pro/con ratio (attitudinal mix) is the control variable with the strongest influence. An initial unfavorable ratio can sink desires to achieve high maturity. You may need to assess your aggregate attitudinal mix *before* starting a large improvement effort. Active hiring (and/or conversion) of pro-SPI people before you start your drive for high maturity can help overcome an initial high-risk mix. Active hiring of pro-SPI people can accelerate achieving high maturity by four years as compared to the passive case. Active replacement of con-SPI and even no-care people can accelerate achieving high maturity by another whole year and a half or more from the active hiring case.

Radical improvements require radical actions. In true Systems Thinking fashion, the strongest relationship was not the intuitive issues of "How effective are the ROIs of your SPI efforts?"

or "How well are your procedures documented?", but "What is the aggregate attitudinal mix of your staff towards SPI?" and "How can you influence the aggregate attitudinal mix?" Achieving radical improvements requires a battle for the hearts and minds of the no-cares in order to maximize penetration, or adoption, of the process improvements. The ratio of the pro-SPI people, who inspire, to the con-SPI people, who impede, is crucial. The selective staffing findings should be abstracted up one level, to show management that hiring and replacement are only two means to the end of actively managing the aggregate attitudinal mix. Addressing the cons, converting the no-cares, carefully selecting leaders, using turnover to your advantage, modifying rewards and recognition structures, communicating the SPI message, and funding appropriate support are other means to manage the attitudinal mix of staff.

7. Validation of Results by SEL and Others

An important part of the learning process was the presentation of these simulation findings to key SEL personnel. The presentation triggered new learning that the simulation did not uncover, and that would not have been as well articulated if the simulation had not been done:

- Managers need to plan future projects that use the major SPI activities so that the staff can use the new skills and technology brought on by process and technology maturity. If the staff members are not able to use their new skills, they will go elsewhere.
- Although counterintuitive, it is the no-cares that managers should focus their attention on because they may change their attitudes more easily than the cons. The pros and cons are mostly set in their ways.
- Team leaders have a strong influence on whether their staff attitudes are pro- or con-SPI. This can cause wild swings in the pro/con mix if there is turnover in the team leads causing shifts in attitude. In the subjective survey described in this report, some people felt that if the senior manager did not support SPI, then the rest of the organization would not either.
- Cons are often heroes and opinion leaders that are looked up to by others. The cons do not think they are getting value from SPI and may perceive that they have a lot to lose. It is important to identify the cons and try to involve them in SPI because diversity can strengthen the end result. However, poorly planned SPI efforts may increase polarization of cons and pros. It is important not to label cons as "bad" people, but people who have different viewpoints. Cons may even be right in some cases.
- Although SEL personnel agreed with the overall qualitative findings, some of the quantities used in the simulation were difficult to validate against reality. They had difficulty accepting the pro/con to SPI effectiveness quantification. However, they agreed with the qualitative issue that attitudinal mix has a strong influence on SPI success. They pointed out that the study did not model the negative feedback due to reorganizations, budget cuts, and the resulting low morale. They also pointed out other detailed points. For example,
 - Performance differences between pros, cons, etc., were not modeled.
 - Different major SPIs may have different personality mix effectiveness relations.
 - The rate of quitting may vary by attitude type.
 - High maturity retained existing pros as well as attracting pro-SPI new hires.
 - Quality improvements did not necessarily result in immediate schedule improvements.
 - Trying to actively hire pros may produce gaming in which all new hires say they are pro-SPI.
 - Pros may get turned off with SPI if it is forced upon them.

Although not verified yet, *it is felt that these missing details will change the quantities of the simulation runs, but not the qualitative findings.* These issues pointed out by the SEL constitute even more learning, even though they were not simulated.

In validating the life-cycle values for some of the high-performance cases, the final values for schedule and effort seem too small to reflect reality. For example, in Table 10 of Appendix C,

the last 3 cases show final schedules of 1.5 to 4 weeks and effort of about 2,000 hours. These seem much too small for a real project. The simulation, although based on valid empirical SEL project metrics for the initial values and life-cycle relationships, focuses on the *relative* ordering of the final values for each input combination. The input combinations with the best or worst relative final values are the combinations that a manager can learn from. The goal of this simulation is not project estimation, as previous published simulations attempt to do. The goal of this simulation is to determine the value of high maturity and to learn which means to achieve high maturity produce the best relative performance.

After this simulation was done, a recent SEL project was completed which had extremely high reuse and very effective quality processes.⁴ This new project had an effort and schedule improvement that was 2 orders of magnitude greater than a similar project that had been worked on 10 years ago. The effort and schedule values for the recent project were in the single digits and, coincidentally, resembled the final values of the simulation. Although it would be too much to say that the simulation predicted the values of this recent project, this helps to validate the simulation to a degree. What can be said is that the relatively small values for effort and schedule produced by the simulation and demonstrated by this recent project are possible and have been achieved.

⁴ Waligora, S. Phone conversation and slide of recent project.

8. Conclusion

In the SEL context, the quantitative value of high maturity compared to medium maturity is high—on the order of 100% to 250% potential improvement in size, effort, quality, and schedule over a 10-year period. The value of high maturity compared to low maturity is even greater. The qualitative value of high maturity, as seen in this simulation as well as in interviewing other organizations (Motorola, Raytheon, etc.), is an increase in the probability of successfully achieving your goals. The feedback loops that exist in any complex organization can either amplify or dampen efforts to improve performance. A careful learning process that includes subjective surveys, System Dynamics simulations, and presentation of results to the users can effectively identify your feedback loops, including people issues, and help managers successfully implement performance improvement.

This effort took five solid person-months of one person's time working eight hours a days with less than five percent of his time interrupted by meetings, etc. In (Yourdon, Rubin et al. 1994), Ed Yourdon reports that one of his efforts to create a systems-dynamics simulation of a software process took one of his staff members two to four person-months. These simulations do take a lot of time to create if you want a relatively accurate and valid simulation. It is highly recommended that you retain the services of a systems-dynamics consultant to help accelerate the learning curve needed to create these simulations.

As far as future efforts are concerned, Professor M. Lehman of the U.K.'s Imperial College is embarking on a much more rigorous project to simulate the feedback phenomena of several software organizations (Lehman 1994; Lehman 1995). One of his major goals is to see how the various feedback loops in a software organization accelerate or dampen performance improvement efforts. He observes that many improvement efforts (such as high-level languages, CASE tools, etc.) provide benefit to the local process within a software organization. However, large potential organization-wide improvements are severely dampened by the feedback loops that exist between people, departments, processes, reward mechanisms, etc. His study should provide tremendous insight into how to intelligently introduce performance improvements. Staff at Motorola (both India and the U.S.), as well as other leading software companies, have also expressed interest in using Systems Dynamics to simulate their software organizations. It is hoped that the findings in this report help encourage managers to take a system-wide view of implementing performance improvement and help show others that the complex software development process can be simulated to a reasonable degree.

Appendix A: iThink Simulation Diagrams and Low-Level Design

The next three pages show the actual simulation coded in the iThink Systems Dynamics application. "Underneath" each box (stock) or circle (flow or converter) are equations and logic that control the execution of the simulation. A quick low-level design description of each diagram follows:

1. The life-cycle subsystem. Starting at the upper left hand corner and working clockwise, the personality (attitude) mix converter adjusts any of the schedule, size, etc., ROI rates based on the Pro to Con ratio. Moving over to the top center, cumulative size SPIs affect the rate of change of size. In the upper right corner, size changes affect error rate according to the graph in Figure 2. Also, a SPI may affect error rate directly. In the lower right corner, the impact that size or error rate might have on effort is modeled. In the lower center part of the diagram, changes in effort due to size or error rate changes are combined with changes in effort due to effort SPIs to model overall effort changes. Finally, in the lower left corner, effort changes affect schedule according to Figure 3. Also, schedule SPIs may also affect schedule directly.
2. The people (staff) subsystem. Starting at the right, experienced no-cares, cons and pros are separate flows that quit according to the quit rate. The quit rate itself changes with maturity. For each quit, a no-care, con or pro new hire is determined by a Montecarlo probability distribution that is also based on maturity (see the left part of the diagram). If an experienced con quits, a pro may replace him or her if the maturity is high. The time the no-cares, cons, or pros spend in training (the middle slotted boxes) is regulated by the current project schedule. As process improvements decrease cycle time, the learning curve time is also decreased. As process improvements increase maturity, the attitudinal mix based on the new pro and con totals also changes because new hires may be of a different attitude than the experienced person who quit.
3. The KPA processing subsystem. This is the most complex subsystem. The top half deals with major SPIs, the bottom half models the minor SPI suggestion process. For the major SPI section starting from the left, major SPIs are piloted and deployed as regulated by schedule (see top center for the schedule box). In the center, the "Pilot Major Cum" and "Deployed Major Cum" receive the impulse of the completed piloted or deployed SPI at the end of a cycle and store the cumulative total. In the center, the "Total Major SPI" sends out the appropriate ROI percent based on when the pilots and deploys are complete. This ROI percent is sent out to one of either the "Size Major ROI," "Effort Major ROI," "Error Major ROI," or "Schedule Major ROI" flows located on the top right corner.

For the bottom half that models the minor SPI suggestions, the bottom center shows the "Sug Frequency" being a product of "Total Maturity," number of pros, "Suggestions per Pro per cycle," and "Percent of Suggestions from Outside." The structure to the right of this causes the suggestions to be implemented in a "low-hanging fruit" fashion. That is, within a project cycle, a few suggestions are made at the beginning, many in the middle, then only a few at the end since all of the easy ones (the low-hanging fruit) were worked. The structure on the lower left corner is needed to compute the average suggestion frequency for a floating period of one half of a project cycle. As the suggestion frequency increases, the "Minor SPI Maturity" also increases because when an organization is mature, everyone is involved in making improvement suggestions.

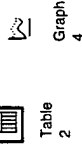


Figure 5. The Life-Cycle Subsystem

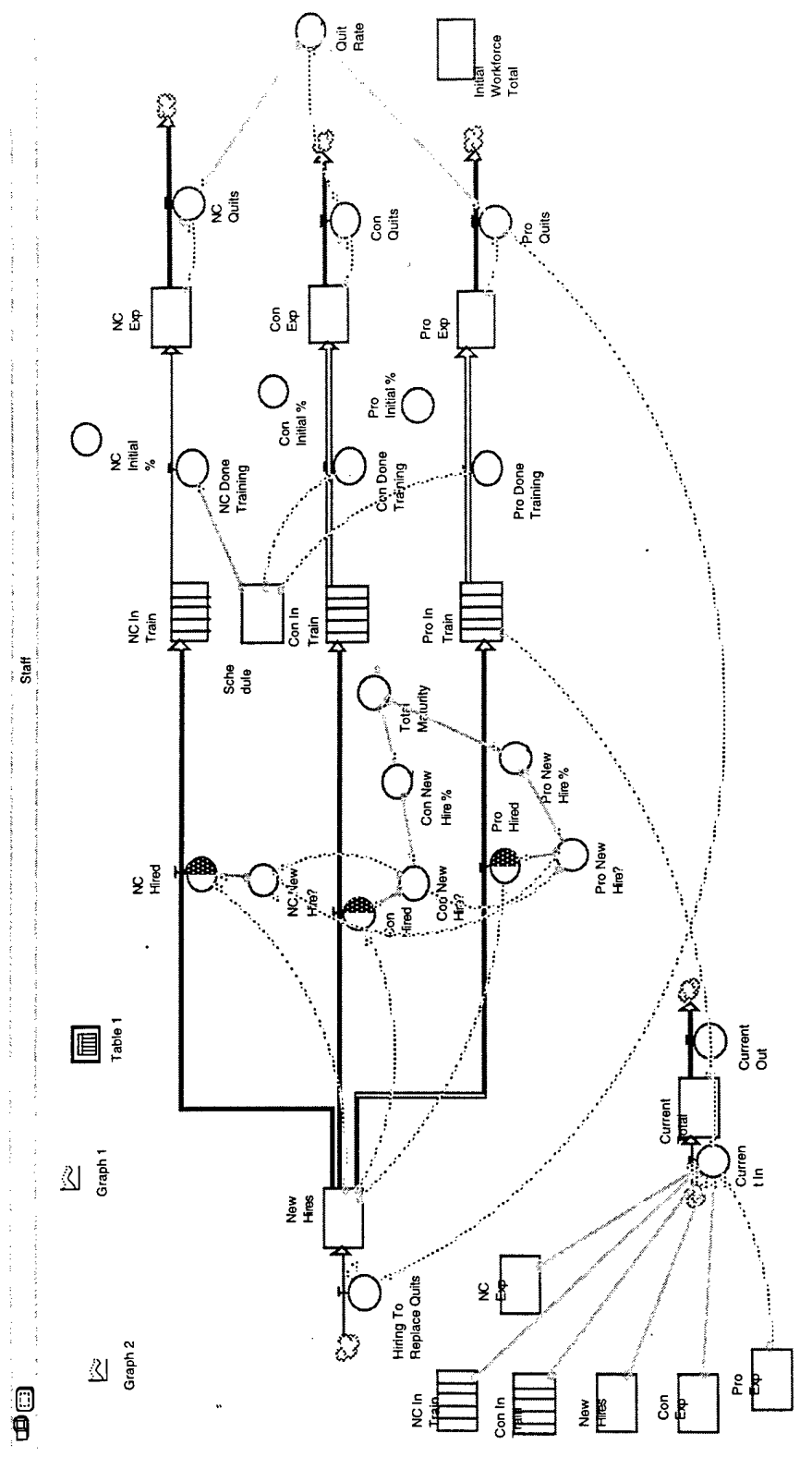


Figure 6. The People (Staff) Subsystem

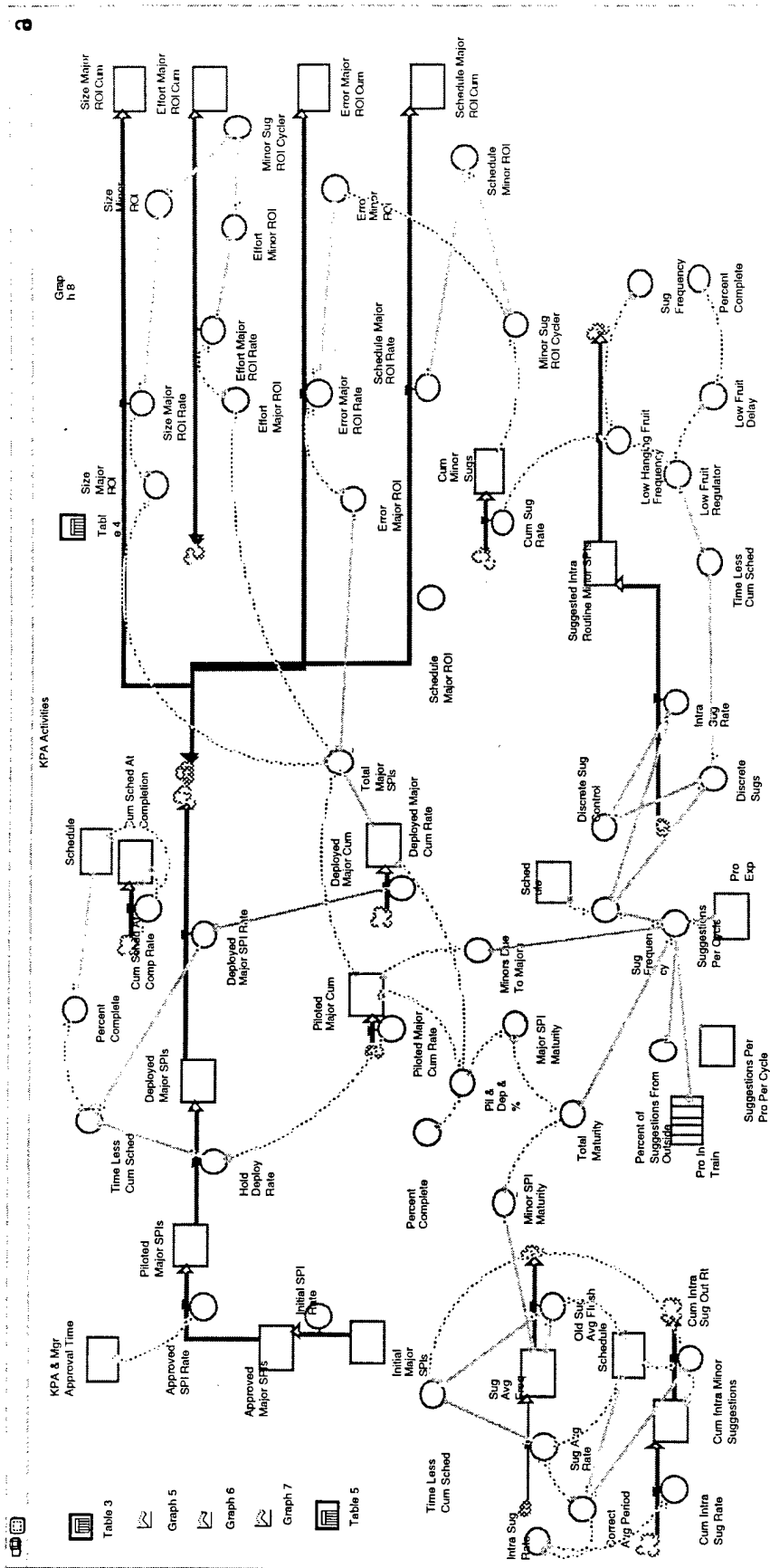


Figure 7. The KPA Processing Subsystem

Appendix B: Simulation Testing Design and Results

Once the simulation was developed and validated for correct execution, an overall test strategy had to be designed that would reveal the *relative* value of each combination of “policy-control variables,” which are defined below. An organization-level viewpoint was chosen. This viewpoint would represent the total potential benefit (or loss) of performance across the five ongoing projects within the SEL. Performance would be measured as the values of effort, error rate (quality), schedule, and final size needed to produce a product that initially required 100 KDLOC or 150 KDLOC. Each simulation test run would be executed for a 500-week (about a 10-year) period. The final values of size, effort, etc., can be compared for each combination of policy-control variables to see which combinations bring the most benefits.

The following management “policy-control variables” were tested:

1. Amount of ROI improvement (benefit) from minor SPIs (small suggestions) on either size, schedule, effort, or quality. The values used were 1%, 0.5%, and 0.25%. Discussion of these values is in Section 5.3 (Processing of Maturity Level 4 and 5 KPAs). The short name for this variable is ROI.
2. Amount of SPI suggestions per pro-SPI person per cycle. The values used were 0.2 and 0.4. The desire to test the cases of having each pro-SPI person make one or two suggestions per cycle must be normalized by the five projects since the pros are spread across them. This gives the values of 0.2 and 0.4. The short name for this variable is SUG.
3. Percent of total suggestions that were lessons learned from other outside projects (Defect Prevention Program). Values used were 60%, 30%, or 0. The use of 60% was based on conversations with SEL personnel on how much they benefited from lessons from outside projects. The 30% value was given by Motorola and Raytheon as approximate values that they have experienced. 0% represents the case of no DPP. The short name for this variable is OUT.
4. Initial personality mix—the pro/con ratio. The values used were 1.5, 1, 0.66, or 0.5. These values represent initial pro/con ratios of 30/20, 25/25, 20/30, and 16.66/33.33, respectively. The initial amount of no-cares was always set at 50%. The total SEL staffing was fixed at 300. This represented the approximate staffing level for this part of the NASA contract. The short name for this variable is PRO/CON.
5. Attitude type hiring policy. Either let increased maturity passively attract pro-SPI new hires or let the managers actively hire pro-SPI new hires even when starting at a low maturity. The short name for this variable is HIRE.
6. Project size. Values used were 100 and 150 KDLOC. Only a few runs were done for 150 KDLOC. The subsequent table of run results will have those runs marked.
7. “Cut-out” versus non “cut-out” cases. In the cut-out cases, projects accomplish SPI activities in isolation with no sharing of lessons. The non cut-out case has a fully staffed group that would accomplish organization-level SPI activities across projects using the CMM Maturity Level 4 and 5 key process area activities as a guide to pilot and deploy major SPIs across projects, extract lessons from projects, communicate them to other projects, and staff an organization-level metrics program. The SEL calls this staff the “Experience Factory” staff. The subsequent tables of run results will have “CUT OUT” labeled on the appropriate runs.

An example of how the policy control variables relate to the output values (size, effort, etc.) using the organization viewpoint is now given. During the simulation time of the first project cycle, a major SPI is piloted to only one of the five projects. So only 20% of the potential improvement of the SPI will be implemented during that time. During the time of the second project cycle, the other 80% will be implemented. The policy control variables influence the initial magnitude and frequency of the minor SPI suggestions, as well as the initial prevailing attitude mix. As the maturity and the number of pro-SPI staff increases, the number of minor SPIs implemented will also increase and show incremental improvement throughout the project cycles. A fifth output value is *time to reach high maturity*. It is measured by the number of weeks needed for the maturity variable to reach its high value range of 1.6 to 2. The frequency and amount of major and minor SPI improvements change the output values of size, effort, etc. An important output value is schedule (cycle time). As it decreases (improves), more major SPIs can be piloted and deployed. It should be emphasized that although actual SEL project metrics were used as a foundation for the simulation, the final performance values of size, effort, etc., will not match an actual SEL project. They are used to determine only the *relative* value of various management policy decisions and, therefore, to help achieve the real goal: to learn more, faster.

Table 1 shows how to determine the value of a combination of policy-control variables by comparing the inputs and outputs of each combination. ROI, SUG, OUT, PRO/CON, and HIRE are input variables. EFF, ERR, SCHD, SIZE, and MAT are final output values (end) corresponding to effort, error rate, schedule, size, and time to maturity, respectively. Please note that size can change due to reuse and other improvement methods. For example, a final size value of 34.51 KDLOC means that after all of the improvements are done, only 34.51 KDLOC is needed to be developed to satisfy the requirements that the original starting value of 100 KDLOC was intended to satisfy.

The following tables show the various combinations, their input and output values, and comments that give an interpretation of the results. The first five columns on the left show the input control variable values. The second five columns show the final output values in bold. A comment column gives a quick qualitative interpretation of the results. The bold underlined input value shows the only value that was changed from the previous case.

Input Variables					Output Values					Comments
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	
0	0	0	0	0	15.9	1.0	87.7	100	N/A	Starting values
.5	.2	0	1.5	Pass	5.1	.94	34.46	48.08	>500	No DPP
.5	.2	<u>60</u>	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL value of DPP

Table 1: Example Input and Output Values for the Starting Values and Two Combinations

In Table 1, the first case (first row) shows the starting values of effort, etc., at the beginning of the simulation run. It represents the case of a low-maturity organization that does no improvement efforts. In comparing the second and third cases, only the OUT variable was changed from 0% to 60%. The second case represents an organization that has no DPP cross-project sharing of lessons. The lower the final output values, the better. The baseline SEL combination gave values of 3.33, 0.71, 14.8, 34.51, and 255 for effort, error rate, schedule, size, and time to maturity, respectively. Note the higher values, and therefore lost potential, when there is no sharing of lessons. Sharing lessons for a 10-year period could potentially decrease effort over 30%, from 5.1 KHours to 3.33 KHours.

The next table, Table 2, shows the results from varying SUG and OUT in order of improving combinations (from bad to better). The five cases below show the benefits of increasing lesson sharing (OUT) and suggestion frequency (SUG). An organization that creates an infrastructure for its process improvement process that can handle more suggestions (0.4) with moderate lesson sharing (30%) can greatly improve performance compared to an organization that has a low suggestion rate (0.2) with moderate lesson sharing.

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	0	1.5	Pass	5.1	.94	34.46	48.08	>500	No DPP
.5	.2	<u>30</u>	1.5	Pass	4.83	.91	31.71	46.86	>500	30 vs. 0 OUT
.5	.2	<u>60</u>	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	<u>.4</u>	<u>30</u>	1.5	Pass	3.19	.69	13.1	32.94	218	Big improvement from 0.5, 0.2, 30
.5	.4	<u>60</u>	1.5	Pass	3.14	.68	12.23	31.94	198	Small improvement from baseline case

Table 2: Value of Varying SUG and OUT

The next table, Table 3, shows how an organization with a high-risk pro/con ratio (attitudinal mix) can have a severely degraded performance.

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	.2	60	<u>1</u>	Pass	7.19	.95	43.36	62.35	>500	Big drop due to bad pro/con
.5	.2	60	<u>.66</u>	Pass	9.54	.96	64.54	75.35	>500	Even bigger drop

Table 3: Value of Various Attitudinal Mixes

The next table, Table 4, shows how an organization that actively hires pro-SPI people in spite of starting at a low maturity can greatly improve its performance and increase its time to reach high maturity by four years.

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	30	1.5	Pass	4.83	.91	31.71	46.86	>500	Moderate OUT
.5	.2	30	1.5	<u>Act.</u>	3.28	.71	14.73	34.78	272	Active hiring causes big improvement

Table 4: Value of Active Hiring of Pro-SPI People

Figure 8 shows how effort, size, etc., changed over the 500-week period. Note that the Y axis has different scales for each of the different values. Each line is numbered. For example, line 2 represents the value of error rate. Its scale on the Y axis ranges from 0 to 1.5. Line 4 represents size. Its scale on the Y axis ranges from 0 to 100. The big step-drops represent the piloting and deploying of major SPIs. The gradual decrease in values in the right half of the graph show the benefits of continual implementation of minor SPI suggestions.

Figure 9 shows how the staff attitudinal mix changed over time. Although not shown until Figure 10, high maturity was achieved around week 255. After that point, a dramatic increase in the number of pro-SPI staff occurred since high maturity attracted pro-SPI new hires. It is interesting to note the counterintuitive finding that it is the no-cares who leave or convert to pros, not the cons, when the number of pros increased after week 255. This is reflected by comparing the large drop in no-cares to the relatively small drop in cons during that time period.

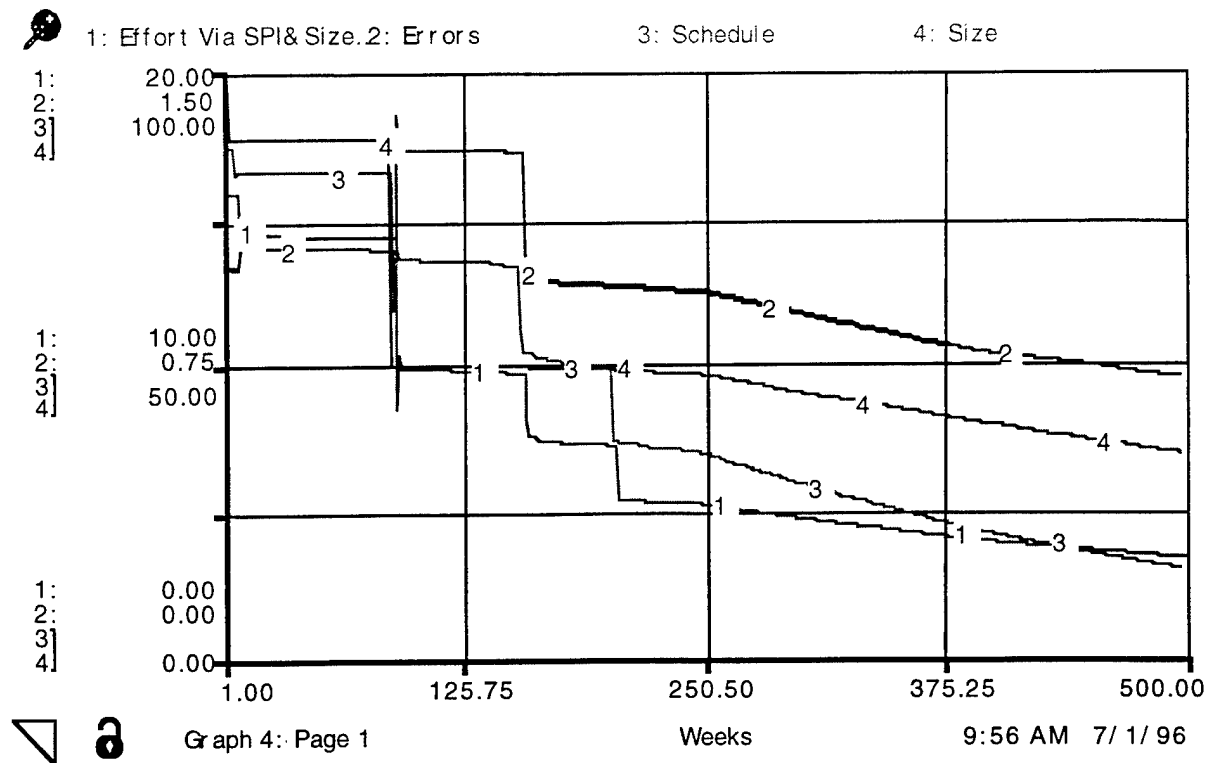


Figure 8. Test-Run Values for Effort, Errors, Schedule, and Size for the SEL Baseline Case

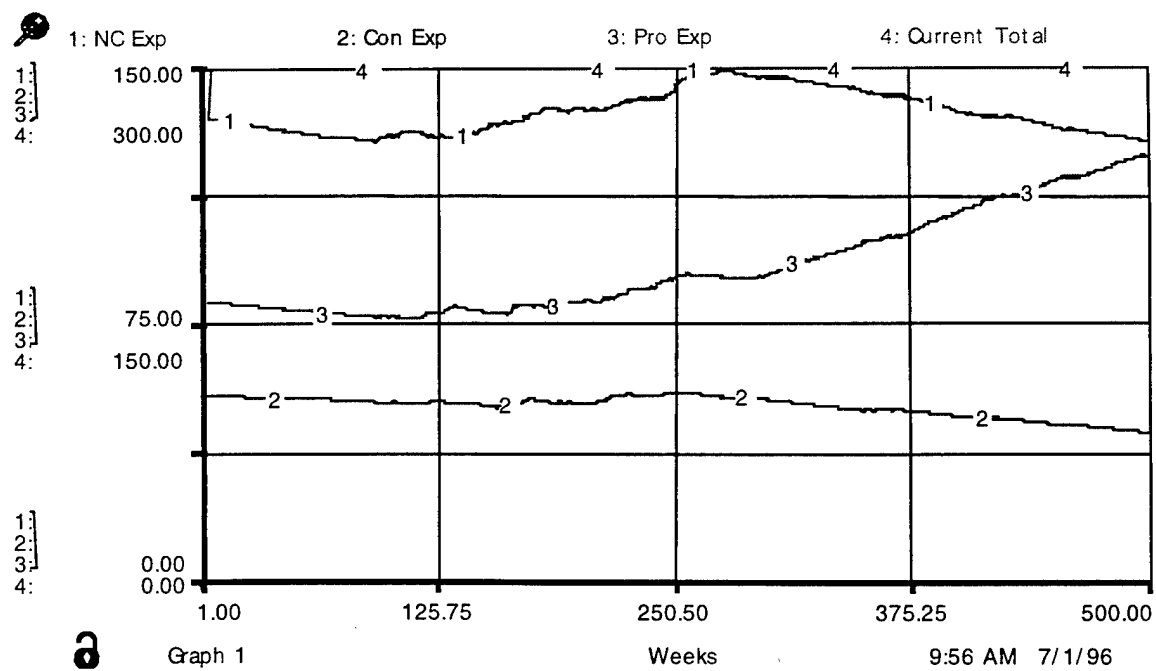
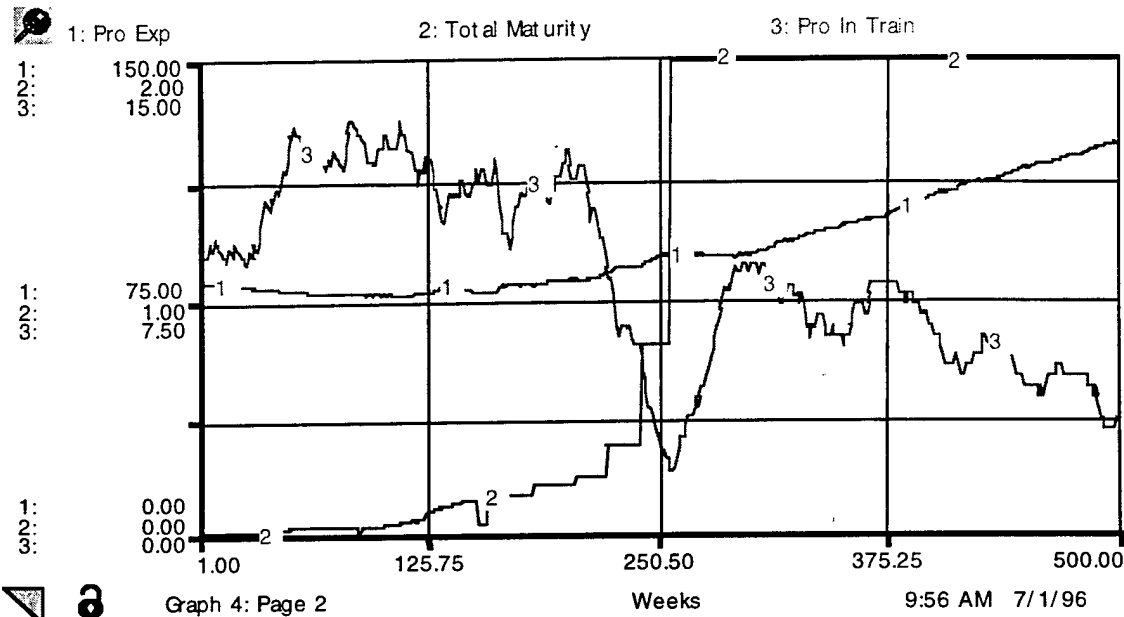


Figure 9. Flow of the Three Attitude Types Across Time for the SEL Baseline Case

Figure 10 shows the backlog of pro-SPI new hires ("Pro In Train") waiting to be promoted to the pro-SPI experienced level as process improvements decreased cycle time and turnover. Another interesting phenomena is that a new increase in the backlog of pro-SPI new hires occurs after high maturity is reached. This knowledge can allow managers to plan for this second backlog rather than be surprised.



**Figure 10. Backlog of Pro-SPI New Hires Related to Maturity
for SEL Baseline Case**

Table 5 shows the case of actively hiring pro-SPI staff to help overcome an initial high-risk pro/con ratio. In this case, a good process improvement infrastructure of high ROI and frequent suggestions is combined with actively hiring pro-SPI staff. This combination of management decisions can greatly improve potential performance if there was an initial high-risk pro/con ratio.

Figure 11 shows the dramatic difference in how pro-SPI new hires and total pro-SPI staff change when the manager actively hires pros for the SEL baseline. The number of pro-SPI staff starts to increase dramatically before high maturity is achieved.

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF	ERR	SCHD	SIZE	MAT	Comments
					End	End	End	End	End	
1	.4	30	.66	Pass	8.93	.96	58.43	72.58	>500	Bad mix
1	.4	30	.66	<u>ACT</u>	4.32	.59	11.20	41.1	305	Active Pro hiring with a good ROI and SUG can overcome bad initial mix

Table 5: Results of Actively Hiring Pro-SPI Staff

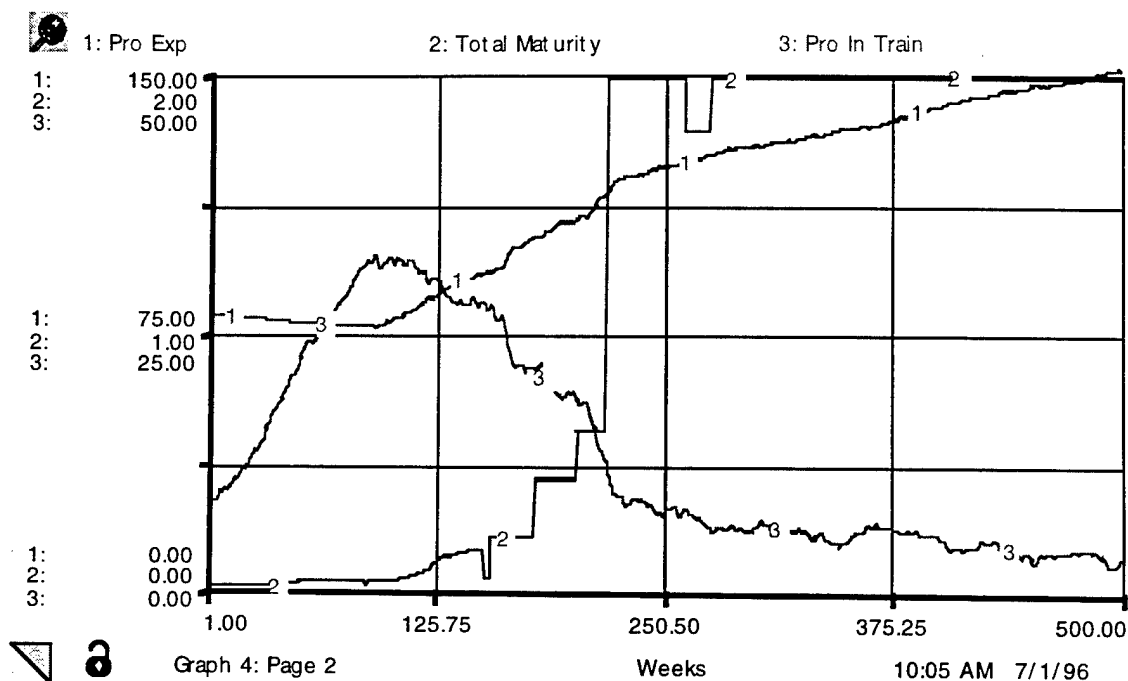


Figure 11. Pros Increase Before High Maturity Achieved for Active Hiring

Table 6 shows the impact of having low ROI (0.25%) suggestions. A high suggestion frequency and high lesson sharing (OUT) can help overcome the effects of low ROI. The table even shows that a bad pro/con mix with low ROI can be overcome with active hiring of pros and having high suggestion frequency and lesson sharing.

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
<u>.25</u>	.2	60	1.5	Pass	4.2	.81	24.12	41.6	260	Low ROI Impacts
.25	.2	60	<u>1</u>	Pass	7.49	.97	50.96	64.17	>500	Big loss due to bad mix and low ROI
.25	<u>.4</u>	60	1	<u>Act</u>	5.4	.79	29.79	53.52	224	Active with frequent sugss overcomes bad mix & low ROI

Table 6: The Impact of Low ROI and How It Can Be Overcome

Table 7 shows the value of taking the radical action of replacing con-SPI staff with pros after the first pilot. Replacing pros with cons accelerated time to high maturity by another year and a half from the active hiring case.

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE FIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	.2	60	<u>1</u>	Pass	7.19	.95	43.36	62.35	>500	Base bad mix
.5	.2	60	1	<u>Act</u>	4.56	.71	21.11	46.94	288	Active hiring improves perf.
.5	.2	60	1	<u>Act / 1st Pilot</u>	2.66	.65	10.08	29.61	194	WOW - Better than baseline that had a good mix
Cons	Re-	placed	by	Pros						

Table 7: Actively Replacing Cons Can Rapidly Overcome Bad Initial Mix

Tables 8 and 9 show the case for a large-size project—starting off with a 150 KDLOC project instead of 100 KDLOC. They also show the “cut-out” cases. The “cut-out” cases show the lost potential of having the Experience factory staff “cut out.” In these “cut-out” cases, no staff pilots and deploys major SPIs across projects. Each project does their own piloting one cycle at a time. Also, there is no lesson sharing (OUT = 0%).

Input Variables					Output Values					
ROI	SUG	OUT	PRO/CON	HIRE & SIZE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	.2	60	1.5	Pass & <u>150</u>	6.31	.78	38.74	66.02	470	Large initial size needs more time to reach maturity
					KDL	OC				

Table 8: Large KDLOC Case

Input Variables					Output Values					
ROI	SUG	OUT	PRO/CON	HIRE	EFF Final	ERR Final	SCHED Final	SIZE Final	MAT Final	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	.2	0	1.5	Pass	5.1	.94	34.46	48.08	>500	No DPP
.5	.2	0	1.5	Pass	7.17	.96	43.31	58.01	>500	Cut-out has big impact
		THIS	IS	CUT	OUT		CASE			

Input Variables					Output Values					
ROI	SUG	OUT	PRO/CON	HIRE & SIZE	EFF Final	ERR Final	SCHED Final	SIZE Final	MAT Final	Comments
1	.4	60	1.5	Pass	2.13	.48	1.48	20.14	190	Best baseline
1	.4	0	1.5	Pass	5.16	.83	27.19	48.44	461	Cut out - Even bigger percent impact on high performance
	CUT	OUT	CASE		FOR	BEST	BASE	LINE		

Table 9: "Cut-Out" Cases

For the large-size combination, the large cycle time needed to complete the project greatly slowed down the time to reach high maturity. Since the SEL tracks its major SPIs to cycle time,

it is important to choose projects with a short cycle time to show immediate results and to get the process improvement momentum rolling.

For the two cut-out cases, the lost potential between the baseline cases and the cut out cases are large—on the order of 100% to 250%. The best baseline represents the case where there are very experienced “hot shots” who can make frequent, high-benefit SPI suggestions. Their loss is greater than the baseline case. Interpretations are discussed later.

Figure 12 shows the change in size, effort, etc., for the cut-out case. Note that there are no large dramatic drops since there are no deploys across the entire organization. However, unlike the baseline case which only had three major SPIs, the cut-out case always has a project piloting a major SPI. This is reflected in the many step-drops in the graph. In this case, the final values do not come close to the final values of the non-cut out case. High maturity was never achieved.

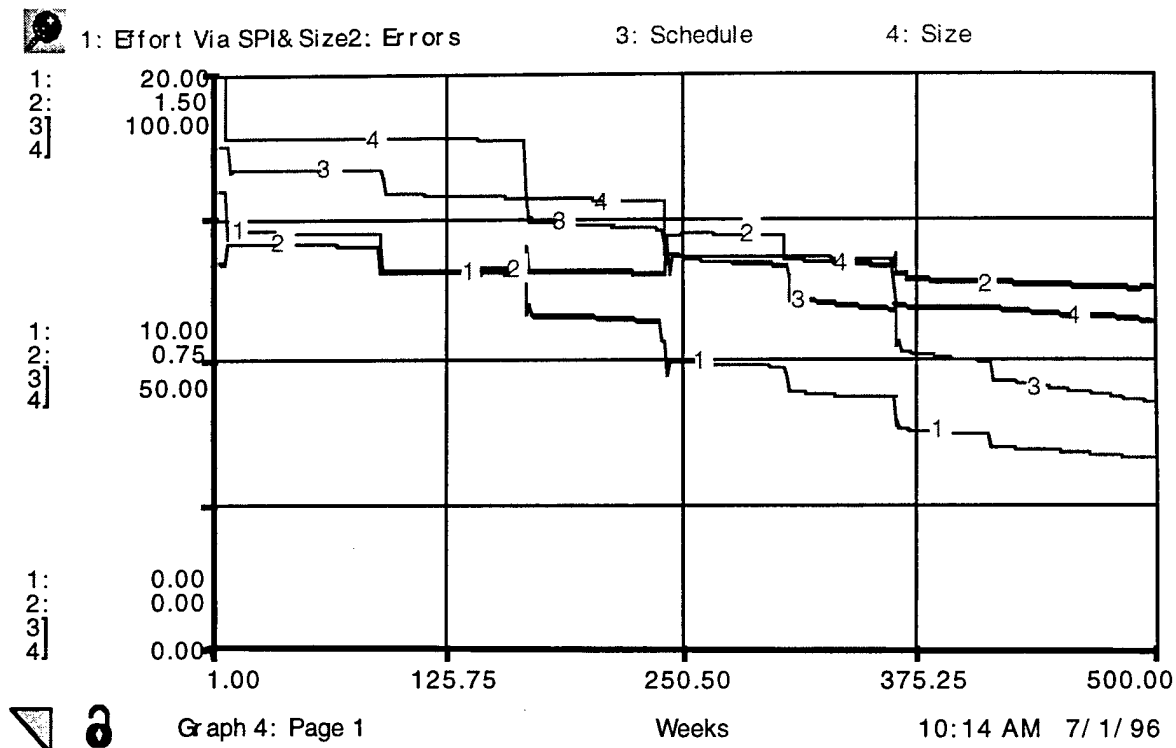


Figure 12. The Impact of the Cut-Out Case

Appendix C contains more combinations of input variables, their results, and a brief interpretation of those results.

Appendix C: Test Output Tables

The following tables show most of the combinations of input variables along with the resulting output values and a concise comment that describes an interpretation of the results.

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	.2	<u>30</u>	1.5	Pass	4.83	.91	31.71	46.86	>500	30 vs. 60 OUT
.5	.2	<u>0</u>	1.5	Pass	5.1	.94	34.46	48.08	>500	Value of DPP
.5	<u>.4</u>	<u>60</u>	1.5	Pass	3.14	.68	12.23	31.94	198	Value of Doubling SUG
.5	.4	<u>30</u>	1.5	Pass	3.19	.69	13.1	32.94	218	Big improvement from .5, .2, 30
<u>1</u>	<u>.2</u>	<u>60</u>	1.5	Pass	2.21	.52	3.27	22.54	228	Value of doubling ROI
1	<u>.4</u>	60	1.5	Pass	2.13	.48	1.48	20.14	190	Best baseline
1	.4	<u>0</u>	1.5	Pass	2.25	.54	4.06	23.6	248	High ROI and SUG offset low OUT

Table 10: Value of Varying ROI, SUG, and OUT

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	<u>1</u>	Pass	7.19	.95	43.36	62.35	>500	Big drop due to bad PRO/CON
.5	.2	60	<u>.66</u>	Pass	9.54	.96	64.54	75.35	>500	Even bigger drop
<u>1</u>	<u>.4</u>	<u>60</u>	.66	Pass	5.34	.72	21.04	47.09	295	Big drop from best baseline
.5	.2	60	1.5	<u>Act.</u>	3.06	.67	12.02	32.08	216	Active causes improvement from baseline
.5	.2	<u>30</u>	1.5	<u>Act.</u>	3.28	.71	14.73	34.78	272	Big Improvement From .5, .2, 30
.5	.2	<u>0</u>	1.5	<u>Act.</u>	4.04	.82	23.8	42.41	432	Big Improvement from .5, .2, 0

Table 11: Value of Various Attitudinal Mixes and the Value of Actively Hiring Pros

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
1	.4	30	<u>.66</u>	Pass	8.93	.96	58.43	72.58	>500	Bad mix
1	.4	30	.66	<u>ACT</u>	4.32	.59	11.20	41.1	305	Active pro hiring can overcome bad initial mix
<u>.5</u>	<u>.2</u>	<u>0</u>	<u>1</u>	<u>Pass</u>	7.84	.99	54.17	65.27	>500	Bad mix and bad ROI, SUG & OUT
.5	.2	0	1	<u>ACT</u>	6.3	.88	41.88	62.76	>500	ACTIVE alone not much help without ROI, SUG & OUT
.5	.2	<u>60</u>	<u>.66</u>	<u>Pass</u>	9.41	.96	63.68	74.94	>500	Baseline with bad mix
.5	.2	60	.66	<u>ACT</u>	6.04	.78	30.75	58.8	>500	ACTIVE w/OUT helps some

Table 12: More Cases of the Value of Various Attitudinal Mixes and the Value of Actively Hiring Pros

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
<u>.25</u>	.2	60	1.5	Pass	4.2	.81	24.12	41.6	260	Low ROI impacts
.25	.2	<u>0</u>	1.5	Pass	5.37	.96	37.02	49.08	>500	Low ROI and no DPP bad combo
.25	<u>.4</u>	<u>60</u>	1.5	Pass	4.03	.78	22.07	40.08	203	Frequent suggs alone don't make that much difference
.25	.2	60	<u>1</u>	Pass	7.49	.97	50.96	64.17	>500	Big loss due to bad mix and low ROI
.25	.2	60	1	<u>Act</u>	5.54	.8	31.94	55.44	273	Active w/ frequent suggs overcomes bad mix & low ROI

Table 13: Impact of Low ROI and How It Can Be Overcome

Table 14 shows the value of taking the radical action of firing cons after either first pilot or the first deploy, and hiring either no-cares or pros.

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE FIRE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	.2	60	<u>1</u>	Pass	7.19	.95	43.36	62.35	>500	Base bad mix
.5	.2	60	1	<u>Act / 1st Dep.</u>	3.77	.66	16.19	40.97	252	WOW - Almost as good as baseline SEL
NO	Care	Hire								
.5	.2	60	1	<u>Act / 1st Pilot</u>	2.66	.65	10.08	29.61	194	Better than baseline that had a good mix
PRO S	Hire									
.5	.2	60	.66	Pass	9.41	.96	63.68	74.94	>500	Base very bad mix
.5	.2	60	.66	<u>Act / 1st Pilot</u>	2.98	.66	12.38	33.09	236	Hiring no-care better than keeping cons!
NO	Care	Hire								

Table 14: Active Firing of Cons Can Rapidly Overcome Bad Initial Mix

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE & SIZE	EFF End	ERR End	SCHD End	SIZE End	MAT End	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	.2	60	1.5	Pass & <u>150</u>	6.31	.78	38.74	66.02	470	Large initial size needs more time to reach maturity
					<u>KDL</u>	<u>OC</u>				

Table 15: The Large Project Size Case

Input Variables					Output Values					
ROI	SUG	OUT	PRO/ CON	HIRE	EFF Final	ERR Final	SCHED Final	SIZE Final	MAT Final	Comments
.5	.2	60	1.5	Pass	3.33	.71	14.8	34.51	255	Baseline SEL
.5	.2	<u>0</u>	1.5	Pass	5.1	.94	34.46	48.08	>500	No DPP
.5	.2	0	1.5	Pass	7.17	.96	43.31	58.01	>500	Cut out has big impact
		THIS	IS	CUT	OUT		CASE			

INPUT VARIABLES					OUTPUT VALUES					
ROI	SUG	OUT	PRO/ CON	HIRE & SIZE	EFF Final	ERR Final	SCHED Final	SIZE Final	MAT Final	Comments
1	.4	60	1.5	Pass	2.13	.48	1.48	20.14	190	Best baseline
1	.4	0	1.5	Pass	5.16	.83	27.19	48.44	461	Cut out - Even bigger % impact on high performance
	CUT	OUT	CASE		FOR	BEST	BASE	LINE		

Table 16: The "Cut-Out" Cases

Appendix D: Definitions of Terms and Acronyms

The following terms and acronyms are used in this report:

SPI	Software process improvement effort. Can be major if managed by the QA/SEPG staff, or minor if suggested and implemented by the technical staff.
ROI	Return on investment. The quantitative <i>benefit</i> of implementing a SPI is the ROI. A ROI can benefit life-cycle size, effort, error rate, or schedule. Note: This definition of ROI is different from the classic use of ROI in business contexts. In our context, ROI stands for benefit in terms of percent improvement from the current value of size, effort, etc. Size is a means to an effort, quality, and/or schedule benefit.
KPA Processes	Activities in CMM Maturity Levels 4 and 5 key process areas that are done by QA, SEPG, or technical staff. Distinguished from direct software development work. The SEL used the term "Experience Factory staff" to describe the people involved in major SPI activities.
DPP(OUT)	Defect prevention process. The amount of minor SPIs a project receives from other outside projects. The benefit of not repeating other's mistakes.
Pro-SPI (people)	People who support and want to participate in SPI activities, spread the word, and make minor SPI suggestions.
Con-SPI (people)	People who are antagonistic towards SPI and spread that word.
No-care (people)	People who do not care about SPI but can be influenced in their adoption of SPI by the pros or cons.
Attitude impact	The effect that the ratio of pros to cons has on the potential benefit of adopting a SPI. It can increase or decrease total ROI.
SEL	Software Engineering Lab run by CSC, NASA Goddard, and University of Maryland. Is composed of five separate software development departments/projects.
Maturity	The probability of successfully accomplishing your goals. This is the spirit of CMM, but does not quantitatively correspond to the five discrete maturity levels of the CMM.
Size	The amount of work performed on a particular project. It is measured in thousands of <i>developed</i> lines of code (KDLOC) to distinguish it from total lines of code (KSLOC). KSLOC may include reused code.

Effort	The cost of performing the required work in thousands of person-hours.
Error rate	The quality of the work product measured in real code or design errors, found in any development stage (reviews or testing) as reported by the SEL, per thousand lines of developed code (expressed in Errors/KDLOC).
Schedule	The elapsed calendar time used to produce the product, measured in weeks.

References

- Abdel-Hamid, T. and S. Madnick (1991). Software Project Dynamics. Englewood Cliffs, NJ, Prentice Hall.
- Glickman, S. and J. Kopcho (1995). Belcore's Experiences Using Abdel-Hamid's Systems Dynamics Model. 1995 COCOMO Conference, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University.
- Goldenson, D. and J. Herbsleb. After the Appraisal: A Systematic Survey of Process Improvement, Its Benefits, and Factors that Influence Success (CMU/SEI-95-TR-09, ADA 302225). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University.
- Hansen, G. (1996). "Simulating Software Development Processes." IEEE Computer 29(1): 73-77.
- HPS (1994). Introduction to Systems Thinking and iThink. iThink Technical Reference Manual. Hanover, NH, High Performance Systems, Inc.
- Johnson, M. (1995). IT Organization Flight Simulation. Menlo Park, CA, Bartz Associates.
- Lehman, M. (1994). Evolution, Feedback and Software Technology. 9th International Software Process Workshop, Airlie, VA, IEEE Press.
- Lehman, M. (1995). Process Improvement - the Way Forward. 7th International Conference, CAISE '95, Jyvaskyla, Finland.
- Madachy, R. (1996). Process Modeling with Systems Dynamics. 1996 SEPG Conference, Atlantic City, NJ, Software Engineering Institute, Carnegie Mellon University.
- McGarry, F., R. Pajerski, et al. Software Process Improvement in the NASA Software Engineering Laboratory (CMU/SEI-94-TR-22, ADA 289912). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University.
- Nguyen, N. and S. Ahmed (1995). Dynamic Software Process Modeling. 1995 SEPG Conference, Boston, MA, Software Engineering Institute, Carnegie Mellon University.
- Pajerski, R. (1995). What's Happening in the SEL?. Twentieth SEL Workshop, NASA GSFC, Greenbelt, MD, 1995.
- Rubin, H. (1996). Dynamic Modeling of the Personal Software Process. IT Effectiveness Workshop, Seafood, NY, IT Effectiveness, Inc.
- Senge, P. (1990). The Fifth Discipline: The Art and Practice of the Learning Organization. New York, Doubleday Press.

Tvedt, J. and J. Collofello (1995). Evaluating the Effectiveness of Process Improvements on Software Development Cycle Time via System Dynamics Modeling. International Conference on Software Engineering, IEEE.

Waligora, S., J. Baily, et al. The Impact of Ada and Object Oriented Design in the Flight Dynamics Division at GSFC (SEL 95-001). Greenbelt, MD, NASA GSFC.

Waligora, S. and R. Coon (1995). Improving the Software Test Process at NASA SEL. Twentieth SEL Workshop, NASA GSFC, Greenbelt, MD, 1995.

Yourdon, E., H. Rubin, et al. (1994). "With the SEI As My Co-Pilot." American Programmer(Sept. 1994): 50-57.

Bibliography

SEL Reports

Agresti, W., F. McGarry, et al. Manager's Handbook for Software Development (SEL 84-101). Greenbelt, MD, NASA GSFC.

Basili, V. Software Design - A Paradigm for the Future (UM IASC TR 89-57). College Park, MD, University of Maryland.

Basili, V. (1992). The Experience Factory - Can It Make You a 5? 17th Annual Software Engineering Conference 1992, Greenbelt, MD, NASA GSFC.

Basili, V. and G. Caldieri. The Experience Factory - Strategy and Practice (SEL 95-003). College Park, MD, Software Engineering Laboratory.

Basili, V., F. McGarry, et al. The SEL - An Operational Software Experience Factory (SEL 92-003). College Park, MD, University of Maryland.

Decker, W., R. Hendrick, et al. SEL Relationships, Models, and Management Rules (SEL 95-003). Greenbelt, MD, NASA GSFC.

Doerflinger, C. Monitoring Software Development Through Dynamic Variables, Rev. 1 (SEL 93-106). Greenbelt, MD, NASA GSFC.

Kistler, D., J. Bristow, et al. Annotated Bibliography of SEL Literature (SEL 82-1306). Greenbelt, MD, NASA GSFC.

McGarry, F., S. Waligora, et al. Recommended Approach to Software Development, Rev. 3.0 (SEL 81-305). Greenbelt, MD, NASA GSFC.

McGarry, F., S. Waligora, et al. An Overview of the SEL (SEL 94-005). Greenbelt, MD, NASA GSFC.

SEI Reports and Tools

Austin, R. and D. Paulish. A Survey of Commonly Applied Methods for Software Process Improvement (CMU/SEI 93-TR-27, ADA 278595). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University.

Curtis, B., W. E. Hefley, et al. People Capability Maturity Model (CMU/SEI 95-MM-02, ADA 300822). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University.

Dion, R. and T. Haley. Raytheon Systems Experience in Software Process Improvement (CMU/SEI 95-TR-17, ADA 303319). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University.

HPS (1994). Introduction to Systems Thinking and iThink, iThink Technical Reference Manual. Hanover, NH, High Performance Systems Inc.

Kellner, M. (1991). Software Process Modeling Support for Management Planning and Control. 1st International Conference on the Software Process, Redondo Beach, CA, IEEE Press.

Paulk, M., B. Curtis, et al. Capability Maturity Model for Software, V1.1 (CMU/SEI 93-TR-24, ADA 263403). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University.

Raffo, D. Assessing the Impact of Process Changes for Large Scale Software Development Using Process Modeling (CMU TR WP 1993-14). Pittsburgh, PA, Carnegie Mellon University.

Raffo, D. (1995). Modeling Software Processes Quantitatively and Assessing the Impact of Potential Process Changes for Process Performance (PhD dissertation). Pittsburgh, PA, Carnegie Mellon University.

Software Process Improvement

Bach, J. (1994). "The Immaturity of the CMM." American Programmer(Sept 1994): 13-17.

Biehl, R. (1995). "The Application of Statistical Process Control at CMM Level 4." Software Process Improvement Forum(May/June 1995): 18-21.

Brodman, J. and D. Johnson (1995). "Return on Investment from Software Process Improvement as Measured by US Industry." Software Process Improvement and Practice 1(1): 35-47.

Debou, C., M. Haux, et al. (1995). "A Measurement Framework for Improving Verification Processes." Software Quality Journal 4(3): 207-225.

DeMarco, T. (1995). Why Does Software Cost So Much? New York, NY, Dorset House.

DeMarco, T. and T. Lister (1987). Peopleware. New York, NY, Dorset House.

Glass, R. (1995). Software Creativity. Englewood Cliffs, NJ, Prentice Hall.

Goodhew, P. Ensuring Profitable Investment in Software Process Improvement London, U.K., ESPI Foundation.

Gottesdiener, E. (1996). "What is Your Development Maturity?" Application Development Trends: 60-81.

Henry, J. (1995). "Quantitative Evaluation of Software Process Improvement." Journal of Systems and Software: 169-177.

Humphrey, W. (1989). Managing the Software Process. Reading, MA, Addison-Wesley.

Johnson, A. (1994). Software Process Improvement Experience in the DP/MIS Function. Proceedings of the 16th International Conference on Software Engineering, IEEE Press.

Jones, C. (1995). Return on Investment in Software Measurement. Burlington, MA, SPR.

Jones, C. (1996). "The Economics of Software Process Improvement." Computer 29(1): 95-97.

Lehman, M. (1991). "Software Engineering, the Software Process, and Their Support." Software Engineering Journal 6(5): 243-257.

Lehman, M. (1994). Evolution, Feedback and Software Technology. 9th International Software Process Workshop, Airlie, VA, IEEE Press.

Mogilensky, J. and B. Deimel (1994). "Where Do People Fit in the CMM?" American Programmer(Sept. 1994): 36-43.

Paulk, M., C. Weber, et al. (1995). The Capability Maturity Model - Guidelines for Improving the Software Process. Reading, MA, Addison-Wesley.

Tanaka, T., K. Sakamoto, et al. (1995). Improvement of Software Process by Process Description and Benefit Estimation. ACM International Conference on Software Engineering, Seattle, WA, ACM Press.

Tausworthe, R. A General Software Reliability Process Simulation Technique (NASA JPL Publication 91-7). Pasadena, CA, NASA JPL.

Tausworthe, R. (1992). "Information Models of Software Productivity: Limits on Productivity Growth." Journal of Systems and Software: 185-201.

Tausworthe, R. and M. Lyu (1994). A General Software Reliability Process Simulation Technique and Tool. International Conference on Software Engineering, IEEE Press.

Valdes, R. (1995). "In Search of Best Practices." Dr. Dobbs' Developer Update 2(11): 1-6.

Weinberg, G. (1992). Systems Thinking. New York, NY, Dorset House.

Weller, E. (1993). "Lessons From Three Years of Inspection Data." IEEE Software 10(5): 38-45.

Wohlwend, H. and S. Rosenbaum (1994). "Schlumberger's Software Improvement Program." IEEE Transactions on Software Engineering 20(11): 833-889.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (LEAVE BLANK)		2. REPORT DATE June 1997	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization			5. FUNDING NUMBERS C — F19628-95-C-0003
6. AUTHOR(S) Steven Burke			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-96-TR-024
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/AXS 5 Eglin Street Hanscom AFB, MA 01731-2116			10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-96-024
11. SUPPLEMENTARY NOTES			
12.A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS			12.B DISTRIBUTION CODE
13. ABSTRACT (MAXIMUM 200 WORDS) This report describes the methodology used to create a simulation of a high-maturity software organization and the results of this simulation. The goal of this research was to find the quantitative value of improving from Capability Maturity Model (sm) [(CMMsm)] Level 3 to Level 5. The method was to simulate a high-maturity organization using its actual empirical data and then "cut out" the high-maturity elements of the simulation. The resulting change in software size, effort, schedule, and quality would be a more accurate measure of the value of high maturity than working forward with a low- or medium-maturity organization and merely hypothesizing the activities and values of high maturity. The author used computer simulations based on systems thinking and systems dynamics, which reasonably modeled the "soft variables" of the people aspects of an organization (personnel attitudes, learning curve, participation in software process improvement, etc.). The simulation also related the soft variables to the hard variables of a software organization's life-cycle process (software size, effort, schedule, quality, etc.).			
14. SUBJECT TERMS capability maturity model, computer simulation, software process improvement, systems dynamics, systems thinking			15. NUMBER OF PAGES 70
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL