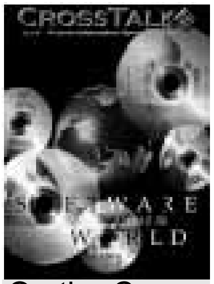
CrossTalk

August 2001 The Journal of Defense Software Engineering Vol 14 No. 8



On the Cover: Kent Bingham, Digital Illustration and Design, is a self-taught graphic artist/designer who freelances print and Web design projects.

CROSSTALK

SPONSOR	*Lt. Col. Glenn A. Palmer*
PUBLISHER	*Tracy Stauder*
ASSOCIATE PUBLISHER	*Elizabeth Starrett*
MANAGING EDITOR	*Pam Bowers*
ASSOCIATE EDITOR	*Benjamin Facer*
ARTICLE COORDINATOR	*Nicole Kentta*
CREATIVE SERVICES COORDINATOR	*Janna Kay Jensen*
PHONE	(801) 586-0095
FAX	(801) 777-5633
E-MAIL	crosstalk.staff@hill.af.mil
CROSSTALK ONLINE	www.stsc.hill.af.mil/crosstalk/crostalk.html
CRSIP ONLINE	www.crsip.hill.af.mil

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail or use the form on p. 11.

Ogden ALC/TISE
7278 Fourth St.
Hill AFB, Utah 84056-5205

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CROSSTALK editorial board prior to publication. Please follow the Author Guidelines, available at www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf. CROSSTALK does not pay for submissions. Articles published in CROSSTALK remain the property of the authors and may be submitted to other publications.

Reprints and Permissions: Requests for reprints must be requested from the author or the copyright holder. Please coordinate your request with CROSSTALK.

Trademarks and Endorsements: This DoD journal is an authorized publication for members of the Department of Defense. Contents of CROSSTALK are not necessarily the official views of, or endorsed by, the government, the Department of Defense, or the Software Technology Support Center. All product names referenced in this issue are trademarks of their companies.

Coming Events: We often list conferences, seminars, symposiums, etc. that are of interest to our readers. There is no fee for this service, but we must receive the information at least 90 days before registration. Send an announcement to the CROSSTALK Editorial Department.

STSC Online Services: www.stsc.hill.af.mil
Call (801) 777-7026, E-mail: randy.schreifels@hill.af.mil

Back Issues Available: The STSC sometimes has extra copies of back issues of CROSSTALK available free of charge.

The Software Technology Support Center was established at Ogden Air Logistics Center (AFMC) by Headquarters U.S. Air Force to help Air Force software organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery.

Software: An International Tie to be Grateful For

I have been extremely fortunate in my lifetime to travel abroad. Europe, Australia, and New Zealand were my destinations. I can remember these trips as if they happened yesterday. From getting my passport to making sure my hair dryer worked overseas, what a thrill it was to prepare for these trips. And from the food, to the people, to the cultures, my experiences on land and sea (i.e., the Great Barrier Reef) were unforgettable.

I also realize that traveling abroad isn't as convenient or even possible without software. Software was an aid to the travel agent that booked my airline, hotel, and tour reservations, and it was an aid to the 747 pilot that flew me halfway around the world. We all know that software is everywhere; it is global. Reflecting on my travels makes me very thankful for this important international tie among countries.

From a defense software community standpoint, software around the world is a critical offensive and defensive element to a country's wartime and peacetime stature. As Americans involved in defense software projects, we work hard to ensure that our armed forces' software intensive systems work worldwide in an effort to outsmart and beat the enemy. But what about our allies, and how they "do" software? All the better if we are interoperable. In this month's issue, Dr. Frederick I. Moxley, Lucien Simon, and Elbert J. Wells present an architectural approach for laying a structural foundation for information interoperability between diverse military information systems, and they discuss why this has been proposed for use throughout NATO.

Just as important as a country's strong defense is their industrial power. Take a look at Australia's information industry in Alastair James' article *Can Australia Improve Its Software Processes?* This country understands the benefits of employing software engineering best practices and processes, but just like any other entity it is hard to put change in place. We know how hard it is for one project or for one organization to embrace change. Is it even possible for a nation to embrace software process improvement all at once?

Also included in this month's issue is a look at India's software industry status. This country has a very strong commitment to software quality as well as to the education and training of software professionals. As Chellam Embar writes in *The State of Software Development in India*, information technology professionals in this country are highly desirable and have more earning power than physicians. Unfortunately, I doubt that this will ever be the case in the United States.

It is refreshing to see that the software best practices that have been created, tried, and proven here in the states are in use throughout the world. It is also very rewarding to be a part of CrossTalk and to disseminate information on software technologies and best practices worldwide. I encourage our international, on-line readers to send in a letter to the editor or to submit an article. Help us share your software lessons learned and best practices worldwide.

I hope you enjoy this month's issue and bon voyage to wherever your travels may take you.

Tracy L. Stauder

Tracy L. Stauder
Publisher

A Foundation for Coalition Interoperability Using NATO's C3 Technical Architecture

Dr. Frederick I. Moxley
Defense Information Systems Agency

Lucien Simon
NATO C3 Agency

Elbert J. Wells
U.S. Mission to NATO

Current projections indicate that in the future, the ability to share information between military systems will ultimately determine whether or not a mission will be a successful. Based on the probability that conflicts will continue to occur involving allied command structures that utilize diverse information systems, information interoperability will be the crucial factor for success when conducting future combined and joint military operations. This paper describes an architectural approach that lays the structural foundation necessary to attain interoperability between diverse C3 systems and provides the rationale on why this approach has been proposed for use throughout NATO.

The North Atlantic Treaty Organization (NATO) has recognized that future military information systems will need to interoperate with one another more effectively than ever before[1]. The number of unforeseen contingencies and international conflicts have elevated the need to provide accurate information to the warfighter upon demand, i.e., wherever and whenever it is needed.

However in order to make this a reality, it is obvious that future coalition information system services will need to be fused together, having the ability to retain their own national identities and operational independence, as well as interoperate with one another in a more effective and seamless manner.

Unfortunately, achieving and sustaining interoperability among diverse systems is not, nor has it ever been an easily attainable objective. As indicated in [1], historically speaking, interoperability has been one of the most difficult areas with which to deal. Interoperability is a broad and complex area of endeavor that cuts across many functional domain areas and applications. Often deemed elusive due to the level of complexity entailed when integrating diverse system components together, the real challenge lies in the overall scope and extent of the system, as well as the level of interoperability and integration desired [2].

Nevertheless, integrating diverse military system components together cohesively within a coalition environment can add significantly to the level of complexity entailed. For instance, when different parts of a system are built separately by independent developers, the end results often vary greatly. This may be attributed to flaws in the design specification and/or how it has been interpreted during various system development stages.

The term used synonymously with design specification today is architectural design. The architectural design is concerned with determining the architectural style of the system as opposed to the detailed design of individual algorithms and data stores. Architectural design also involves the high-level decomposition of the system into components and the relationships and interactions of these components, which usually determines the specific architecture of the system [3]. If misinterpreted or designed poorly, chances are the system(s) once fielded will function improperly, or more than likely, in a limited capacity.

When put in the context of a coalition environment, the ratio for failure increases significantly due to the sheer number of diverse factors that must be taken into account and reckoned with accordingly (e.g., language differences, level of training, number of system developers and integrators involved, type of experience, etc.).

Architectural Views and Interoperability

In 1996, the U.S. Department of Defense (DoD) first introduced the concept of architectural views under the guise of a C4ISR Architecture Framework[2]. Known independently as the Operational, System, and Technical Architectural Views, all three views, when logically combined together, expanded on the de facto definition pertaining to architecture within the realm of information technology[3]. Until that time, there had been no common approach for architectural development throughout the DoD.

As a combined effort, NATO in turn refined each one of these architectural views and incorporated them into what is now known as the NATO Policy for C3[4] Interoperability. All three views as defined below, are considered critical elements of the NATO C3 Interoperability Environment (NIE):

- Operational View: This view describes the tasks and activities, organizational and operational elements, and information flows required to accomplish or to support military or consultation function.
- System View: This view is generated from the Operational View by the responsible host nation or design authority. It describes and identifies the system(s), both internal and external, and interconnections required to accomplish or to support the military or consultation function. This view maps information flows, hardware, and applications to user locations and specifies the connectivity, performance, and other constraints.
- Technical View: This view, generated by the host nation or equivalent authority, describes the arrangement, interaction, and interdependence of the elements of the system and takes into account the technical constraints imposed by the Systems View. It provides the minimal set of rules governing the selection of the appropriate standards and products from the implementation domain.

The NIE encompasses the standards, products, and agreements adopted by the Alliance to ensure C3 interoperability. It serves as the basis for the development and evolution of C3 Systems.

Organizational Structure

NATO has defined interoperability organizationally as the ability of systems, units, or forces to provide services to, and accept services from other systems, units, or forces, and to use the services so exchanged to enable them to operate effectively [4].

The primary organization within NATO that addresses interoperability policy and procedures is the NATO Consultation, Command, and Control Board (NC3B). Structurally, the NC3B consists of eight sub-committees, two of which play an important role in the context of this paper. The first, the Interoperability Sub-Committee is responsible for establishing C3 systems interoperability policy and implementing C3 standardization objectives deemed necessary for improving interoperability. Underneath the Interoperability Sub-Committee are four working groups. Each in their own right helps to perpetuate interoperability policy and standardization initiatives throughout the alliance.

The second, known as the Information Systems Sub-Committee (ISSC) is, at the moment, comprised of eight working groups that primarily address and support information system implementation throughout all of NATO.

When examining NATO's overall interoperability structure collectively, we see that NATO has an interoperability framework (NIF) that can be divided into three distinct categories (see Figure 1):

1. Policy: The NATO Policy for C3 interoperability represents the policy layer. It is a policy that addresses all overarching and essential C3 interoperability issues, identifies each of the respective authorities and associated responsibilities, links existing interoperability documents, defines the relationship with the NATO Standardization Organization, and other relevant organizations.
2. Execution: The NATO Interoperability Management Plan and the five year Rolling Interoperability Program comprise this layer.
3. Products: The NIE comprises this layer [5].

In 1997, the NC3B identified several goals and objectives that were considered necessary to attain interoperability between NATO common funded C3 systems. In response to these goals and objectives, the NC3B ISSC formed the NATO Open Systems Working Group (NOSWG), tasking them to develop a technical architecture on behalf of NATO. The technical architecture would become known as the NATO C3 Technical Architecture (NC3TA) [6].

Upon completion, the NC3TA would provide the structural foundation necessary

"Unfortunately, achieving and sustaining interoperability among diverse systems is not, nor has it ever been an easily attainable objective."

to attain information interoperability between NATO C3 systems and national systems, as well as address interoperability concerns for all NATO common funded systems. Furthermore, the NC3TA would perpetuate the development of a common core for the Bi-SC[5] Automated Information System (AIS).

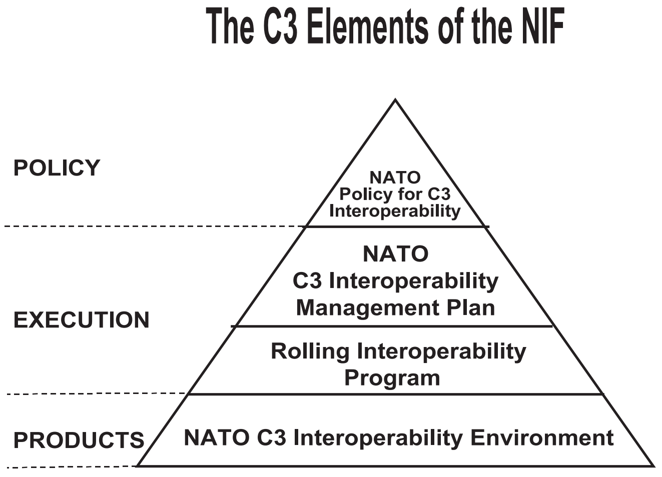
NATO C3 Technical Architecture

To facilitate the creation of the NC3TA, the NOSWG first assessed the merits of each national architectural effort early on, gleaning from each as much as practically possible. Each had technical merit but differed in overall content and composition. As a result, the NOSWG decided to develop the NC3TA in accordance with the definition for a technical architectural view[6] as much as feasibly possible. By definition, this meant that it would provide the minimal set of rules governing the selection of appropriate standards and products from the implementation domain. Moreover, the NC3TA would also extrapolate, as well as improve upon existing approaches from each one of the contributing national technical architectural efforts.

A look at the overall structure and content shows that in contrast to national technical architectural efforts, the NC3TA is unique in that it is comprised of a five-volume set that consists of the following[7]:

- Volume 1–Management: This volume provides the management framework for the development, as well as the configuration control of the NC3TA. It includes the general management procedures for the application of the NC3TA in NATO C3 systems development.
- Volume 2–Architectural Models and Description: This volume principally supports a NATO technical framework to provide a common basis for the establishment of the architecture for NATO information system projects. It also offers a vision on the use of emerging off-the-shelf technologies.
- Volume 3–Base Standards and Profiles: This volume contains all of the current open system and communication standards applicable to NATO information systems, as well as guidance for their use.
- Volume 4–NATO C3 Common Standards Profile (NCSP): This volume mandates the subset of standards that are critical to interoperability. It provides the link between degrees of interoperability as described in the NATO policy for interoperability of C3 systems, and standards selection.
- Volume 5–NATO C3 Common Operating Environment (NCOE): This

Figure 1: *NATO's Interoperability Framework*


The C3 Elements of the NIF
POLICY
EXECUTION
PRODUCTS
NATO Policy for C3 Interoperability
NATO C3 Interoperability Management Plan
Rolling Interoperability Program
NATO C3 Interoperability Environment

volume is the NCSP standards-based computing and communication infrastructure.

The chairman of the NOSWG meets regularly with other NC3B working groups to ensure that all areas of technical concern (e.g., security, data, communications, etc.) are taken into account by the appropriate working group bodies [7]. This simple cross evaluation and coordination procedure serves as only one of the preliminary fail-safe steps that is required as a part of the overall NC3TA management process described in Volume 1.

Consistently updated, Volume 2 reflects various architectural models such as the Technical Reference Model, the NATO Component Model, as well as definitive descriptions or reference pointers to new and emerging technologies such as JAVA and the eXtensible Markup Language. The descriptions provided are primarily derived from the NATO Open Systems Environment and NATO Open Systems Interconnectivity Profile that essentially serve as reference material to the system developer, implementor, and end-user. Editorial updates are made primarily through the NC3 Agency.

The encyclopedic nature of Volume 3 serves as another reference document. It too is derived from the NATO Open Systems Environment and NATO Open Systems Interconnectivity Profile and contains all of the current references on communication and information standards. This volume will also be maintained in an HTML version on the web8.

Due to their impact on the systems design, development, and implementation for all NATO common funded systems, the two remaining Volumes 4 and 5 of the NC3TA are considered extremely important (see Figure 2).

Volume 4, although considered to be quite mature, will undergo periodic updates in order to ensure that the evolution in standards are incorporated to benefit the developer/end-user community on a regular basis. The definitive process for submitting and incorporating candidate standards for consideration into the NCSP is outlined through the "change proposal" section of Volume 1. Volume 4 also has focused on attaining degrees of interoperability through an interoperability profiling procedure that is being worked in coordination with other affiliated sub-committee working groups.

In conjunction with Volume 4, Volume 5 is probably the single most important document within the NC3TA. To note its relevance, all NATO authorities are required, and the nations are encouraged to implement C3 Systems using the mandatory standards and products as specified in the NCSP and NCOE, in accordance with the NATO Policy for C3 Interoperability [8].

Once the NC3B approves future versions of the NCOE, those products that are identified for incorporation will be mandated for all NATO Common Funded Systems.

NCOE Significant Features

Volume 5 of the NC3TA is considered evolutionary and therefore a living document. While it will eventually specify particular products for incorporation into the NCOE, at the present time it does not. However once selected, these products will be primarily chosen from an off-the-shelf -based basket of products. These products will eventually populate the various service layers of the NATO Component Model, which capitalizes on the top-down layered approach provided by the Technical Reference Model as described in Volume 2 of the NC3TA.

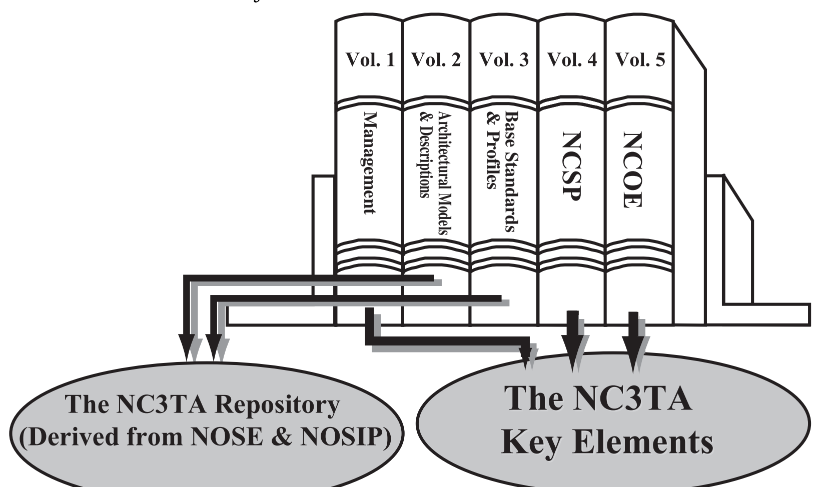
Following are the principle components of the NATO Component Model:

- Network Services: These constitute the basic transparent interfaces between the platform and the underlying networking infrastructure, including the IP layer services.
- Kernel Services: These are that subset of the NCOE component segments that are required for all workstations and servers (see Figure 3). At a minimum, this sub-set would consist of the operating system, windowing software, security services, segment installation software, and an executive manager.
- Infrastructure Services: These services directly support the flow of information across NATO systems. Infrastructure services provide a set of integrated capabilities that the applications will access to evoke NCOE services.
- Common Support Application Services: These services are necessary to view data in a common way (share data) across the network. They essentially promote inter-

Coalition Interoperability Acronym Guide

C3	Consultation, Command and Control.
C4ISR	Command, Control, Computers, Intelligence, Surveillance, and Reconnaissance.
ISSC	International Social Sciences Council
NATO	North Atlantic Treaty Organization.
NIE	NATO C3 Interoperability Environment.
NC3B	NATO Consultation, Command and Control Board.
NIF	NATO Interoperability Framework.
NOSWG	NATO Open Systems Working Group.
NC3TA	NATO C3 Technical Architecture.
AIS	Automated Information System.
NCSP	NATO C3 Common Standards Profile.
NCOE	NATO C3 Common Operation Environment.

Figure 2: *Relative Structure of the NC3TA*


Vol. 1
Vol. 2
Vol. 3
Vol. 4
Vol. 5
Management
Architectural Models & Descriptions
Base Standards & Profiles
NCSP
NCOE
The NC3TA Repository (Derived from NOSE & NOSIP)
The NC3TA Key Elements

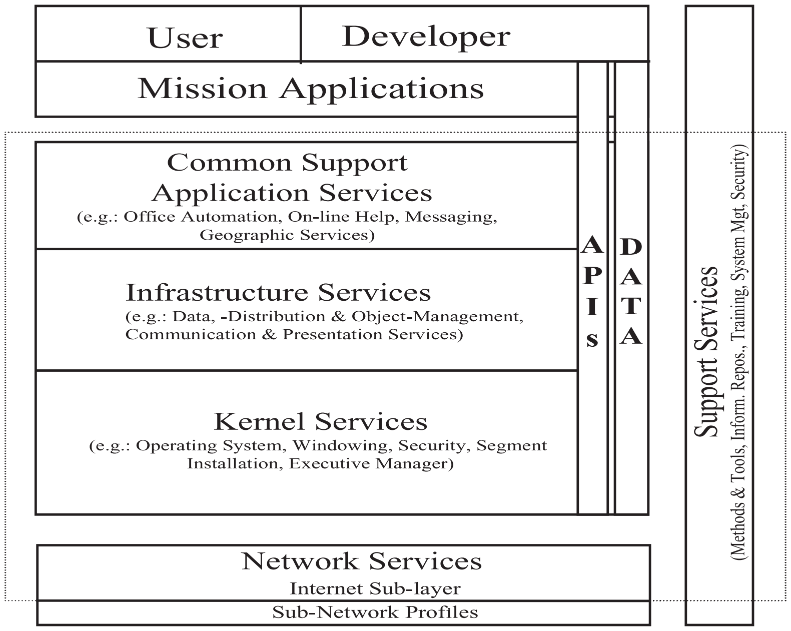

User
Developer
Mission Applications
Common Support Application Services
(e.g.: Office Automation, On-line Help, Messaging, Geographic Services)
Infrastructure Services
(e.g.: Data, -Distribution & Object-Management, Communication & Presentation Services)
Kernel Services
(e.g.: Operating System, Windowing, Security, Segment Installation, Executive Manager)
APIs
DATA
Support Services
(Methods & Tools, Inform. Repos., Training, System Mgt, Security)
Network Services
Internet Sub-layer
Sub-Network Profiles

Figure 3: *NCOE Component Model*

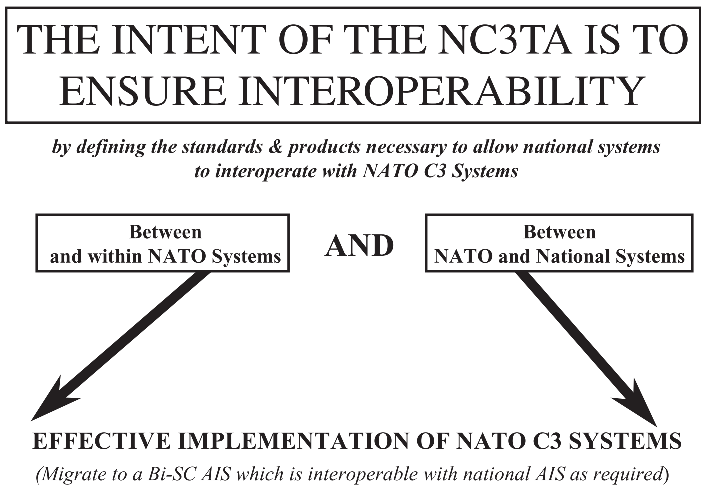
operability among various mission applications.

- Application Programming Interfaces: These are integrated into the NCOE through a common set of application programming interfaces, which are invoked by the applications and services as required.
- Data Component Definition: This refers to the way in which data is taken into account in the NCOE and is related to the main components of the NCOE (common support application services, infrastructure services, kernel service) and even, out of NCOE components *stricto sensu*, to mission applications.
- Support Services: These include methods and tools, information repository, training services, system management, and security.

Segmentation is one of the most debated and often discussed features of the NCOE. Segmentation can be defined in terms of the functionality that is seen from the end-user's perspective. It allows the user(s) to easily add only those required modules that are deemed necessary by the end-user community. This way, the end user may view the NCOE as a set of building blocks in which a system is built. Since the NCOE is not a system in and by itself, it can be more easily understood as the foundation for building open systems through such inherent features as segmentation. The overall concept for segmentation is predicated on national[9] as well as commercially viable efforts.

As noted previously, one of the goals and objectives of the NC3TA is the development of a common core. In direct response to this need, the Bi-SC AIS core will eventually be implemented utilizing those standards and products stipulated by the NCSP and NCOE. However, to do so will require that the basket of products be populated in the NCOE. The initial version of the NCOE was released in July of 1999 as Volume 5 of the NC3TA. The latest NC3TA version 2.0 was approved in May 2001 by the NC3 board. Version 2.0 provides an outline of the basket of products, as well as the set of interoperability standards profiles to be used by the Bi-SCs.

Figure 4: *Interoperability and the NC3TA*


THE INTENT OF THE NC3TA IS TO ENSURE INTEROPERABILITY
by defining the standards & products necessary to allow national systems to interoperate with NATO C3 Systems
Between and within NATO Systems
AND
Between NATO and National Systems
EFFECTIVE IMPLEMENTATION OF NATO C3 SYSTEMS
(Migrate to a Bi-SC AIS which is interoperable with national AIS as required)

Conclusion

Interoperability has long been an elusive and sought after goal. Especially, within the realm of coalition information systems. However, a well defined architectural approach can lay the structural foundation necessary to attain interoperability for diverse military information systems in the future (see Figure 4).

When all five volumes of the NC3TA are finalized, it is anticipated that the structural foundation will be in place for future coalition systems to build systems upon for years to come.◆

References

1. Wentz, Larry, *Lessons From Bosnia: The IFOR Experience*, National Defense University Press, Washington, D.C., 1997, p. 434.
2. Moxley, Frederick I., Interoperability and the DII COE, Proceedings of the International CIS Interoperability Conference, London, March 2000.
3. Moxley, Frederick I., On the Specification of Complex Software Systems, Proceedings of the Second IEEE International Conference on Engineering of Complex Computer Systems, Oct. 1996.
4. Joint Publication 1-02, Joint Chiefs of Staff, Washington, D.C., 1994.
5. Vogt, Bernd (Col., GE), An Outline of NATO C3 Standardization Interoperability Policy Issues, Proceedings of the International CIS Interoperability Conference, London, March 2000.
6. NATO C3 Technical Architecture (NC3TA), Version 1.0, July 30, 1999. NATO HQ, B-1140 Brussels, Belgium.
7. Simon, Lucien (Lt. Col., FR), NOSWG Briefing on the NATO C3 Technical

Architecture, Chairman's Report to the NC3B ISSC, NATO HQ, Brussels, Belgium, Oct. 1999.
8. Wells, Elbert J., The NCOE: Keystone to NATO Interoperability, Proceedings of the London AFCEATechnet Conference, Oct. 1999.

Notes

1. Item 4 of the Defense Capabilities Initiative issued during the Washington Summit on April 23-24, 1999.
2. C4ISR Architecture Framework, Version 1.0.
3. IEEE Std 610.12 lists complete definition.
4. Within NATO, C3 refers to "Consul - tation, Command, and Control."
5. The two Major NATO Commands, i.e., Supreme Headquarters Allied Powers Europe (SHAPE) and Supreme Allied Commander Atlantic (SACL-ANT).
6. For more details, see the NATO C3 Interoperability Environment (NIE).
7. For a complete description, see NC3TA Vol. 1.
8. The NC3TA is accessible at http://194.7.79.15
9. For more details see DII COE at www.disa.mil

About the Authors

Fred Moxley, Ph.D., is a senior technical advisor within the Defense Information Systems Agency. He has several years of experience designing, developing, implementing, and managing a variety of software systems for the Department of Defense, as well as other agencies throughout the federal government. Dr. Moxley is presently the principal U.S. representative to NATO for open systems. His research interests include distributed software system architectures, artificial intelligence, and software design methodologies. Dr. Moxley holds advanced degrees in both telecommunications and computer information systems and sciences.

Defense Information Systems Agency
5600 Columbia Pike
Falls Church, VA 22041
E-mail: moxleyf@ncr.disa.mil

Lt. Col. (Armament) Lucien Simon is chairman of the NATO Open Systems Working Group of the NATO C3 Board's Information Systems Sub-Committee. He joined the NATO C3 agency in September 1997 as a French National Expert. From 1993 to 1996 he was program manager for the French Army Command, Control Information System (CCIS) after having been responsible for various activities within the French CCIS domain. He graduated in the field of armament engineering and holds a post-graduate degree in computer science. In 1997 Simon graduated from the French Joint Defense Staff College.

NATO C3Agency
Rue de Geneve 8
B-1140 Brussels, BE
E-mail: lucien.simon@nc3a.nato.int

Elbert J. Wells has more than 20 years experience in the development and implementation of U.S. national and NATO C3 systems. He is currently at the U.S. Mission to NATO where he is responsible for information system matters. Previous NATO assignments included tours at the former STC and NACISA. Previous U.S. national assignments included the position as project manager of the U.S. Navy Research & Design Distributed C2. Wells holds master's degrees in both electrical engineering and computer science.

U.S. Mission to NATO
Autoroute de Zaventem
1110 Brussels, BE
E-mail: ewells@mitre.org

Call for Articles

If your experience or research has produced information that could be useful to others, CrossTalk will get the word out. We welcome articles on all software-related topics, but are looking for pieces in several high-interest areas. Drawing from reader survey data, we will highlight your most requested article topics as themes for future issues. We will place a special, yet nonexclusive, focus on the following tentative issues of CrossTalk:

CMMI
February 2002
Submission Deadline: Sept. 19, 2001

System Requirement Risks
March 2002
Submission Deadline: Oct. 24, 2001

Forging the Future of Defense Through Technology
May 2002
Submission Deadline: Jan. 2, 2002

Software Estimation
April 2002
Submission Deadline: Nov. 21, 2001

We accept article submissions on all software-related topics at any time; issues will not focus exclusively on the featured theme.
Please follow the Author Guidelines for CrossTalk, available on the Internet at www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf

The State of Software Development in India

Chellam Embar
ChangeShop, Inc.

Recently software development costs in this country have skyrocketed and qualified talent has been difficult to find. An increasing number of organizations have been exploring ways to get developmental help from other countries. This article is the result of the author's participation in a Strategic Technology Tour of India and will try to shed some light on the status of the software industry in that country and how it can benefit the software development needs in the United States.

Micro Electronics and Computer Technology Corporation (MCC) organized a Strategic Technology Tour (STT) of India. The reasons for the tour were twofold:

1. To get an understanding of the state of software development in India.
2. To understand opportunities for partnering with Indian organizations.

Several North American and European companies were represented on the STT team, as was the Center for Information Systems Engineering, Carnegie Mellon University. This author was one of the representatives of that university.

During this two-week tour, the STT team made site visits to Indian-owned software firms, Indian software development centers of North American firms, and joint ventures. Also, team members met with several governmental and quasi-governmental organizations related to the Indian software industry, and participated in four industry roundtables organized by the Software Technology Parks of India (STPI).

Government Involvement

In recent years, the government of India has strongly encouraged the development and export of software by Indian firms. The STPI is a quasi-public organization established by the government of India, Department of Electronics. It plays an important part in providing the infrastructure for the development of the Indian software industry.

STPI provides physical facilities by making office space available on favorable terms to software firms, which conduct a high proportion of export work. In addition, STPI has created an innovative data communications network that employs microwave communications between software development facilities and a local STPI node, and satellite links for long-haul segments (including international). Also, software firms that function under the STPI are able to import computer and telecommunications gear for use in their development centers without paying import duty.

Additionally, state governments are strongly involved in software development activities. The state of Andhra Pradesh for example, is very actively working on establishing state-of-the-art facilities and infrastructures in Hyderabad, the capital. The STT team visited with the chief minister of Hyderabad, who firmly believes in the important role information technology would play in the development of his state as well as the entire country.

Technical Capabilities and Methods

Indian software firms are leaders in software quality and managing quality processes, although they are typically followers in the software development technology domain. The firms the team visited focus on using current-generation tools to maintain and develop software for paying customers, rather than on developing the next generation of tools and techniques.

Most Indian software firms use a variety of software development models – waterfall, spiral, and rapid application development – with the choice driven by the requirements of a particular project such as maintenance or development. Several Indian firms have developed impressive in-house systems for software development support and project management. Most firms have their process documents and manuals on the Intranet or other in-house information system for easy online access.

Contrary to our initial assumptions, many Indian companies are no longer satisfied with "just writing code." Most companies the team visited expressed an interest in moving up the value chain in their relationship with their Western customers. Some companies are already exhibiting increasing capabilities in architecture and whole systems design, systems integration, and systems migration. Favored application domains include banking and finance, electronic commerce, health-care information systems, electronic commerce, and telecommunications and network management.

The STT team, overall, was very impressed with the competence of Indian companies. Most of the companies who had undergone a formal Capability Maturity Model®-based assessment were at a Level 3 or 4. The team had the general impression that even the companies that had not gone through a formal assessment were at a high maturity level. It is to be noted that unlike U.S. companies, Indian companies are reluctant to go through an assessment until they are absolutely sure that they will be assessed at least a Level 3.

Education and Training

It is estimated that India has about 150,000 information technology (IT) professionals. This number can be broken down into roughly 60,000 software engineers, 15,000 graduates in computer science and related disciplines, and 75,000 from other engineering disciplines. In contrast to the *wimp factor* sometimes attached to it in the United States, an IT career is highly desirable in India, and currently has earning power even greater than physicians.

A key piece of infrastructure for the Indian software industry is the country's

® The Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

educational system. There are some 210 universities in operation in India. These universities currently graduate more than 25,000 students in computer science and related disciplines each year, and initiatives are under way to double this number. The Indian Institutes of Technology and the Indian Institute of Science are widely acknowledged to be excellent. The Indian Institutes of Technology have strict admissions criteria, and attract the very best of students. As a result, their graduates are much coveted by the top software companies such as Hughes Software Systems, Citicorp Information Technology Industries, Motorola India Electronics, Infosys, Wipro Systems, and HCL Consulting.

Despite many new entrants to the software engineering ranks, India is now faced with a labor shortage with regard to the highly talented professionals who leave to seek employment in the United States and other Western countries.

Training after hiring is also a high priority for Indian software organizations. Across the sites the team visited, each technical employee receives between 1.5 and 3.5 weeks of training each year. One company has three months of classroom training and an additional three months of on-the-job training for each new employee. Areas emphasized for training included information system technologies, application domains, design/development processes, quality assurance methods, and tools. The most pressing skill needed in the industry is for project managers, and most companies are developing programs to train (some to certify as well) their promising technical personnel for those positions. Several companies have their own training institutes to develop and manage the delivery of the training material.

Managing Human Resources

Companies used various recruitment methods, including newspaper advertisements, working with placement consultants, and recruiting on university and college campuses (quoted as the main source). Some companies claimed to have very stringent recruitment criteria. Only top of the class were invited for the initial interviews and the companies also administered several internal tests before the hiring decisions were made. Most companies preferred graduates with computer science degrees. One company hired graduates from any engineering discipline, but con ducted a mandatory 12-week initial training before commencing regular work.

"It is estimated that India has about 150,000 information technology professionals."

Even though highly qualified talent is more readily available in India than in the United States, retention seems to be a large issue. The attrition rate quoted varied between 12 percent to 30 percent. Main reasons cited were the IT professionals' desire to emigrate to the United States, or their being hired away by other large companies. Indian software engineers can now frequently find positions in North America or Europe with the employer willing to make arrangements for a work visa.

Indian software companies are seeking to counter this problem by creating opportunities for young software engineers to work abroad, and by wooing middle-aged technical managers who may want to return to India to raise their children. Nevertheless, this brain drain is so severe that several of the firms we met with avoid recruiting from the six Indian Institutes of Technology because as many as 95 percent of their graduates go abroad for jobs or further education.

The most successful companies seem to be those that have been the most innovative in managing this issue. These companies are stepping up their attention to employee quality-of-life issues such as choice of assignments and locations, working hours, and training. Several tactics used to retain employees include free or subsidized housing, stock options, opportunities to work abroad, free or subsidized meals, assistance in buying cars, and free transportation. Providing job satisfaction was also mentioned as a key retention factor.

The STT team was consistently impressed with the focused attention paid to retention issues by the human resources departments of most Indian companies. Several attribute this focus to their use of formal models. One uses the Software Engineering Institute's People Capability Maturity Model (P-CMM), while another uses its own organization maturity model. Like the Software Capability Maturity Model (SW-CMM), the P-CMM helps an organization self-assess its capability in key process areas associated with each of five levels of maturity, and guides an organization in improving its level of maturity.

Project Management

The Indian software industry's approach to managing IT projects has evolved significantly during the past decade. It is rooted in the industry's experience with customers in North America and reflects a number of standard U.S. approaches to project management. Under the influence of ISO 9000 and the SW-CMM, their approach has evolved to be more focused on quantitative process management and in-process quality. Today, people, process, and quality considerations primarily drive it.

The project manager is involved with the project planning exercise right from the beginning. The projects are controlled tightly with the help of separate execution plans, quality plans, and configuration management plans. They assess status by using quantitative data (metrics) collected with in-house developed tools. Many companies use a process guide containing the entry criteria, task identification, verification methods, and exit criteria for each process element to plan and manage their projects. Typically, the guides contain about 20 to 25 process elements (six to eight management, 10 to 12 development, and four to six support) and are available on-line to all their employees.

Global Competitiveness

Some companies such as Digital and Sun Microsystems have formed partnerships with North American and European companies in joint software development.

Other companies have dedicated marketing organizations overseas. Even though the software development companies currently enjoy a pricing advantage (developmental costs in India are about one-third of those in the United States), they expect to lose that advantage in the next three to five years and at that time, they expect to compete solely on the basis of quality.

Conclusion

Even though the STT team was aware of increased software development activity in India, some members had assumed that most of it consisted of writing code for systems designed by North American and European companies. It was also assumed that developmental costs in India were considerably cheaper and therefore desirable to North American and European companies. As mentioned earlier though, this price advantage is likely to dwindle as the premier software companies, which take pride in their products and processes, hope to compete strictly on the basis of quality.

This writer's general observation was that the work ethic in the organizations the team visited in India was very high. Even though some of the organizations we visited had not gone through a CMM-based assessment as yet, they were stringently following all of the key processes in anticipation of reaching Level 3 or 4. Unlike the individualistic culture in many U.S. organizations, which resists following processes, Indian companies expect processes to be followed and there are no questions asked. Also, a high value is placed on collecting and tracking metrics and project management.

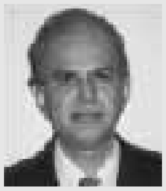
At the conclusion of the tour, the STT team certainly had a better understanding and appreciation for software development capabilities in India. Some of the companies on the tour that already had operations in India have greatly expanded their investment. Other companies such as American Express and Oracle have opened up new operations. According to the Software Technology Parks of India, which has 19 centers throughout the country, software export has shown an increase of 95 percent during the past five years.

Indian software companies place great value in understanding and meeting their customers' requirements and providing the highest quality products and services. Language is not a barrier since English is the language of business in India. The United States has much to gain in associating with Indian companies in terms of working with a highly disciplined workforce with a view toward obtaining timely, cost-effective software of the highest quality.◆

Acknowledgement

The author wants to acknowledge and thank Micro Electronics and Computer Technology Corporation and Global Technology Services for organizing the Strategic Technology Tour of India.

About the Author



Chellam Embar is the principal of ChangeShop, Inc., a consulting firm that offers services in Technology Introduction and Transition. He is also a visiting scientist with the Software Engineering Institute, of Carnegie Mellon University. Embar has more than 20 years of change management experience with technology intensive organizations in private industry, military, government, and non-profit sectors. He has helped organizations in the areas of process improvement, transition and change management, strategic visioning, culture change and values identification, productivity, profitability, and quality improvement efforts, employee involvement programs, communication skills training, team building, conflict resolution, performance management, management development, and career development. Embar recently participated in a Strategic Technology Tour of India, and keeps current with the state software development in that country. Embar received an MA degree in Human Development from Governors State University, University Park, Illinois. He is the author of several books and publications.

ChangeShop, Inc.
10819 Piping Rock Circle
Orlando, FL 32817
E-mail ce@changeshop.com
Phone: (407)-384-4939
FAX: (603)-843-0064

Can Australia Improve Its Software Processes?

Alastair James[1]
STM Consulting

It is critical to the competitiveness of Australia's information industry that the software industry maintains world-class performance by employing best practices in all its business aspects, but most particularly in software development and acquisition. Software Engineering Australia (National) commissioned research on the key quality and process improvement issues being faced by Australia's software industry[2]. This article examines the current quality and processes improvement status of Australia's software industry.

What is the current status in quality and process improvement of Australia's software industry? The consensus is that Australia needs to move quickly to adapt and deploy the tools available in order for its software industry to have the best opportunity for future global competitiveness. This view is based on discussions with key players in the field – acquirers, research institutes, and consultants – and on additional desk research.

Almost all information industries' products involve software, whether it is explicit software systems or software embedded in electronic products or systems. A recent report published by the Information Industries and Online Taskforce, *The Stocktake of Australia's Information Industries*[3] had this conclusion:

> … Australia could become a significant global player in a range of information industries segments and, arguably, has a strategic imperative to do so. Australia's performance to date has shown substantial promise of our ability to achieve this goal … Australia has world class strengths in many aspects of the information industries. We are advanced users of IT&T, with most measures putting us amongst the highest per capita in the world … Our information industries have advanced and demanding clients in sectors such as finance, retail, air transport, government, energy, agriculture and mining, for which the industry has designed world-class solutions.

It is clear, then, that for the global potential of Australia's information industry to be achieved, Australia's software industry must be among the world's leaders in employing appropriate technologies. That includes employing best practices in all its business aspects.

Quality and Process Improvement Benefits

The British Standards Institution (BSI)[4], which has been active in the development of software quality standards, admits that it is generally accepted that the costs of poor software quality are much easier to quantify than some of the benefits of implementing quality systems and process improvement. The costs to the developer of rework and after-sales support, and to customers associated with system non-availability and maintenance charges, are more clearly visible.

"Despite the benefits of software quality and process improvement being demonstrated, why have smaller developers been slow to adopt technologies which will have clear benefits on their bottom line …"

The benefits of using a quality system lie in improvement in quality and repeatability reflected in increased customer satisfaction, higher process efficiencies, and a reduction in failure costs. Failure costs typically comprise costs of correcting defects, cost of overruns against time and budget, unnecessarily high maintenance costs, and loss of business due to poor reputation.

The BSI quotes typical failure costs of up to 20 percent of turnover for developers without a quality system, and that up to half these costs could be saved by implementation of a quality system.

Case studies of Australian and overseas software projects also highlight the benefits to both developers and acquirers of adoption of software quality and process improvement technologies. Benefits for developers include higher quality and productivity, faster delivery and time to market, lower costs, and higher profitability; for acquirers, greater predictability of scheduling, quality and cost, and thus lower risk and happier customers/users.

Several initiatives in software process improvement are underway or being developed in Europe. It appears to be a common characteristic of software industries that awareness of the benefits of quality and process improvement are greatest in larger organizations, and therefore many programs focus on addressing the needs of smaller developers.

In a paper to the Software Engineering Australia 2000 Conference in Canberra, Australia, Fran O'Hara, principal consultant at Insight Consulting Ltd, Dublin, reported that uptake of software process improvement (SPI) "has not yet become ingrained into the culture of the software industry in the manner that it now is in the U.S." Other commentators suggest that, while awareness of SPI is greater in the United States than elsewhere, adoption has been principally among the larger firms.

The Software Process Improvement in Regions of Europe (SPIRE) program is an example of an initiative aimed at the small and medium enterprise market. SPIRE is a European Systems and Software Initiative project financially supported by the European Commission. The Centre for Software Engineering in Dublin undertook the project in Ireland and coordinated activities with partners in Austria, Italy, and Sweden.

SPIRE targets the needs of software development groups with up to 50 staff (either in small software companies or small software units in larger organizations). It has been piloted in more than 70 organizations. A key feature of the work has been the analysis and dissemination of the results through a Web database so that others can learn from the pilot's experience.

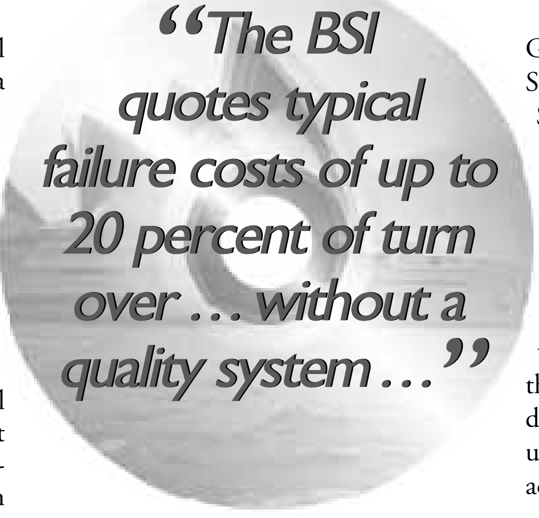
The results showed measurable improvements across participants in all the countries where the project was run. The project was undertaken at no charge to participants, who record high levels of satisfaction with the results. Clear evidence of success is demonstrated by the fact that more than three-quarters of participants felt that they would definitely continue improvement projects without any form of external funding.

Although a range of SPI approaches have been deployed in overseas markets, the majority are based on Software Process Improvement and Capability Determination (SPICE) or the Capability Maturity Model® (CMM®) approaches, in some cases locally adapted to suit the needs of smaller companies. SPICE is an international initiative to develop a widely accepted standard for software process assessment involving groups in many countries around the world. Sponsored by the U.S. Department of Defense (DoD), CMM has been developed at the Software Engineering Institute at Carnegie Mellon University.

Despite the demonstrated benefits of software quality and process improvement, why have smaller software developers overseas and in Australia been slow to adopt technologies that will have clear short-term benefits on their bottom line, and in the longer term may be crucial to the industry's international competitive survival?

Activities in Australia

In Australia, there is no definitive current information on the levels of adoption of quality software assurance and process improvement methods among developers. Nevertheless, anecdotal evidence suggests



"The BSI quotes typical failure costs of up to 20 percent of turn over . . . without a quality system . . ."

awareness and adoption among small-to-medium enterprises is very low.

According to Geoff Bowker, a software engineer and former executive director of SEA in New South Wales, Australia, there are several potential factors affecting take-up of software quality and process improvement technologies. While awareness of such technologies may be high among larger acquirers and developers, the level of general awareness of both the availability of solutions and indeed of the problem is an issue.

Many acquirers, including systems integrators and consultants advising acquirers are not aware that approaches are available to improve their confidence in on-time, on-budget, on-quality delivery, and thus create happy customers. Equally, while some developers have adopted quality standards (for example ISO 9000 series), particularly among small-to-medium enterprises, there is no widespread recognition that technologies are available to address the specific software development and acquisition processes.

Among those with some awareness of process improvement technologies, there is often a perception that they are suitable only for large organizations, that they are costly and time consuming to implement, that their benefits are not proven, and that they require a major commitment of resources. This suggests that many organizations have not been able to link these technologies with the business objectives of reduced cost and improved productivity and customer satisfaction.

A chicken and egg situation in the market for SPI may partly explain low levels of awareness and adoption. Although ongoing work has been proceeding for several years within institutions such as the Software Quality Institute (SQI) at Griffith University and the Australian Software Engineering Institute (ASEI) in South Australia among others, commercial products and services have only recently begun to emerge in the Australian market. Those that have been available have mainly been U.S. developed, and have proven unsuited in their original form to the needs of the typical Australian software company. On the other hand, had there been greater demand, local organizations would undoubtedly have responded earlier with adaptations more suited to local needs.

The DoD, as a major acquirer of software intensive systems, has been in the forefront of investigation of approaches aimed to reduce risk in software acquisition. Within the Defense Materiel Organization, a Software Acquisition Reform Program has been established to review approaches, conduct trials, and develop strategies to achieve this aim. These will need to embrace people, processes, and technologies, on the part of both acquirer and developer, to reduce risk and improve software quality.

So far, one benefit from the U.S. industry's experience is an emphasis on SEI's CMM programs at Carnegie Mellon University (a body sponsored by the DoD). Other initiatives include sponsorship of work with SQI on Capability Maturity Model Integration^{SM} and SPICE methodologies, with the results being fed back to SEI, and liaison with the UK Ministry of Defense.

Some might dismiss such programs as being aimed at large defense contractors with little relevance for Australian small-to-medium enterprises. However, as many larger defense contractors in Australia subcontract work to small-to-medium enterprises, there are obvious benefits if both prime- and sub-contractors are working within the same quality framework. Also, many of the quality improvement elements of these programs are generic, with potential impact throughout the software industry. CMM programs have been adapted to the needs of smaller organizations, too.

In collaboration with SEA[2], SQI has recognized the need for approaches that can generate incremental improvements to demonstrate business benefits, particularly for smaller developers. According to Professor Geoff Dromey, director of the institute, a new Rapid Assessment Program (based on SPICE technology) has been developed. About 30 Queensland software developers have successfully completed trials. Companies such as these typically do not appreciate the strategic benefits of SPI programs. They are too busy running hard to complete projects to be able to justify dedicating the required resources, so quick results are required.

The program provides a relatively low-cost combination of assessment, mentoring, and training during about six months to deliver a structured framework for the developer to implement ongoing improvements. SQI was founded in 1991, and has worked extensively with overseas institutes, including SEI, on SPICE and CMM programs among others.

The ASEI in Adelaide has also responded to a similar need for affordable, bite-sized, easily digestible programs suited to small-to-medium enterprises. Founded in 1995, ASEI is a cooperative enterprise between the software industry and academic and research institutions in South Australia and is supported by the South Australian Government.

Apart from a few larger branches of defense contractors, all software developers in South Australia are small-to-medium enterprises. ASEI is developing a suite of services called Sound Software Engineering Practices for small-to-medium enterprises tailored to their needs as identified in research. In phase one of these services[2], emphasis is on configuration management, which the research highlighted as a major issue in more than 60 percent of small-to-medium enterprises. ASEI also has plans to roll out other modules addressing further SPI areas.

The phase-one service includes a high proportion of customized assessment and mentoring time that provides considerable flexibility and adaptability. Therefore it depends on the availability of trained and experienced staff for implementation. Trials have been successfully completed with 15 small-to-medium enterprises (mostly South Australian, but one each from Victoria and Northern Territory).

Conclusion

There appears to be much good work in software quality and process improvement taking place in Australia. Increased activity to publicize that there is a better way to build and acquire software will result in higher awareness and create demand for the programs becoming available. This then raises the issue of how to ensure adequate resources with appropriate experience and training to support a national rollout.

If deployed more widely, programs tailored to the needs of small-to-medium enterprises, can assist in promoting the concept that worthwhile returns can be achieved on modest investments of time, resources, and money. ◆

Notes

1. Alastair James, director of STM Consulting, undertook research for this report on behalf of SEA.
2. These research programs were funded through the Department of Communications, Information Technology and the Arts via Software Engineering Australia's (National) (SEA) project funding.
3. Stocktake of Australia's Information Industries, A report by STM Consulting Pty Ltd for the Information Industries and Online Taskforce, DIST, Canberra, 1998
4. www.tickit.org/quality.htm

SEA Profile

This article is reprinted with editing from the Software Engineering Australia (National) (SEA) Software Journal, November 2000. SEA is an industry-led body with the charter of improving the quality and reliability of software in Australia. SEA is supported by the Commonwealth through the Department of Communications, Information Techno- logy, and the Arts.

To achieve this Charter SEA focuses on these three core mission statements:

- Provision of information and services to assist entrepreneurs and managers to build evermore robust software businesses focused on exceeding customer expectations (business development).
- Provision of services for both developers and acquirers of software and software systems support for the continuous but rapid improvement of productivity, timeliness, and quality levels.
- Smoothing the path to new technology awareness, understanding, and adoption.

SEA works collaboratively with domestic and global alliances, with government departments, organizations, and professional bodies to drive its mission. SEA serves as an information network to all those in the software industry involved in the research, development, production, acquisition, and use of software in Australia.

E-mail: info@seanational.com.au
www.seanational.com.au

Letter to the Editor

Dear CrossTalk:

Theron Leishman's June 2001 article, *Extreme Methodologies for an Extreme World*, is a nice introduction to the agile-methods world. However, Figure 4, "Evolutionary/Spiral Model," is not the version of this model being endorsed by the DoD 5000 series of regulations. It is instead an Incremental Waterfall process, an example of the "Hazardous Spiral Look-Alikes" that Fred Hansen and I discussed in our May 2001 CrossTalk article, *The Spiral Model as a Tool for Evolutionary Acquisition*. Assuming that a point-solution design for the requirements in increment 1 can be scaled up to the requirements of future increments may work well for small projects done by refactoring experts, but will generally be a disaster for larger-scale and embedded systems.

One way to fix Leishman's Figure 4 is to replace the "Requirements Analysis, Preliminary Design, ..." segments of the spiral by "Inception, Elaboration, Construction and Transition." These phases, used by the Rational Unified Process and MBASE, use risk considerations to determine under what conditions an extreme method or a more heavyweight method will best fit the system's needs.

Sincerely,
Barry Boehm
University of Southern California

Software Maintainability Metrics Model: An Improvement in the Coleman-Oman Model

Aldo Liso
AIPA

This article analyzes the contribution of the number-of-comments parameter used in the Coleman-Oman regression model, a model aimed at determining the maintainability index of a software system. Some characteristics of this parameter can prove to be unsuitable for application to many software systems. A change to the number-of-comments parameter of the Coleman-Oman model will therefore be proposed, as well as a change of the way to obtain the evaluation of the parameter. These changes do not touch the structure of the parameter and are likely to give better results, which can be widely used. This article does not include research that results in a new parameter, but is intended to stimulate thought within the software community that results in the required research and new parameter.

Among the various metrics used to measure software system maintainability, the class of regression models defined by Coleman–Oman and others [1] have raised particular interest in the academic, industrial, and public administration circles. These models are based on a combination of variables (independent metrics) in a polynomial expression that allow calculation of the maintainability index. The studies documented by several software laboratories, the University of Idaho, and Hewlett Packard have shown the high significance, in the software system maintainability evaluation, of the regression models based on Halstead metrics, McCabe's Cyclomatic Complexity, Lines of Code, and Number of Comments.

It is known that the software system maintainability is of the utmost importance, since through it the following is possible:

- Monitor changes to the software system and its quality characteristic.
- Make decisions about the most appropriate maintenance strategy for software procedures by locating the source code that could cause the greatest risks.
- Compare the quality of various software systems and supply information for the best choice among these systems.

The Coleman–Oman Model

The most used model for determining the maintainability index (MI) of a software system is the Coleman-Oman model described in [2]:

Model 1:

1) $$MI = 171 - 5.2\ln(aveV) - 0.23aveV(g') - 16.2\ln(aveLOC) + 50\sin\sqrt{2.46\,perCM}$$

where:

- *aveV* is the average Halstead Volume per module.
- *aveV(g')* is the average extended cyclomatic complexity per module.
- *aveLOC* is the average lines of code per module.
- *perCM* is the average percent of lines of comment per module.

It is known that the first version of Model 1 contained the *aveCM* variable (average lines of comment per module) rather than the *perCM* variable (average percent of lines of comment per module) [2]. However, this first version was not satisfactory because it was too sensitive to the presence of a large number of comments. This means that the presence of large comment blocks, especially inside small modules, results in an increase of the maintainability values. In order to correct such a behavior, the *aveCM* variable was replaced with the *perCM* variable and a threshold value of 50 for this variable was determined and applied [2].

Welker and Oman suggested choosing between Model 1 and the following Model 2:

2) $$MI = 171 - 5.2\ln(aveV) - 0.23aveV(g') - 16.2\ln(aveLOC)$$

which is derived from Model 1 by removing the *comments* variable due to the considerations on the comments arising from code reading. They suggested using Model 2 when the following hypotheses are verified:

- The comments are not closely related to the code. In such situations the comments can become out-of-sync with the code and therefore make the code less maintainable.
- At the beginning of each module there are large blocks of comments (company-standard comments, for example) that bring a negligible benefit to the maintainability of the software. In such situations biased estimates of the maintainability are highly probable.
- There are large sections of code that have been commented out, which creates maintenance difficulties.

In short, the choice between the two regression models depends on the evaluation of the contribution of the comment variable to the software maintainability. Only when this contribution has been demonstrated is it advisable to introduce the number-of-comments variable in the model. In a major research effort due to Hewlett Packard [3], the following thresholds for the evaluation of the maintainability index, calculated by means of the previous models, have been determined:

$MI < 65$	poor maintainability
$65 \leq MI < 85$	fair maintainability
$85 \leq MI$	excellent maintainability

Comments Component Analysis

If we analyze the contribution of the

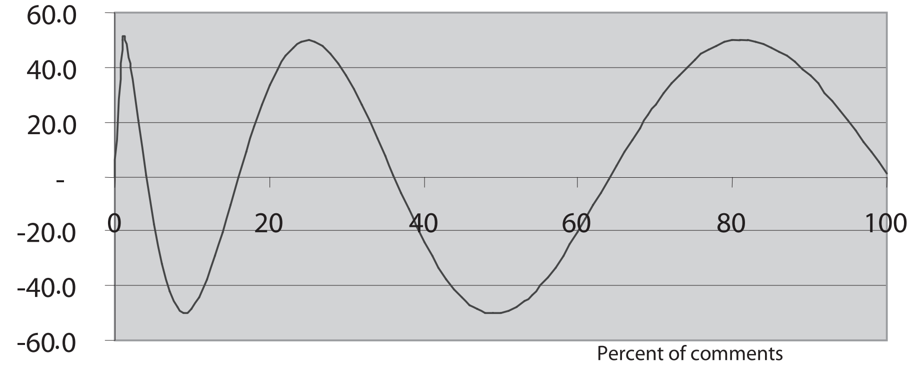


Figure 1: *Values of the Function:* $50\, sin \sqrt{2.46\, perCM}$

perCM factor as indicated in Model 1

$$50\, sin \sqrt{2.46\, perCM}$$

we can notice a sinusoidal path with maximum height equal to 50. The function must be considered in the [0, 100] interval, closed on the left and opened to right, therefore excluding "100" value. In fact, while it is possible to consider a module without comments, it is senseless to consider a module made up by comments only.

By analyzing the path of this function in relation to the maintainability metric, we can notice that there are some drawbacks to this function:

- The function has three peaks at 1 percent, 25 percent, and 81 percent. Now, while it is commonly accepted that a normal value for the comment variable is around 25 percent, the other two values are not justified.
- The decreasing path of the function in the interval from the 1 percent to 9 percent is inexplicable, since the trend would have to be positive in this interval.
- It is difficult to explain the function path in the interval to the right of the 49 percent *perCM* value.

It is also important to make the following remarks. If the comments are significant, as required by the Oman model, the function must be increasing between 0 percent and a fixed maximum value (for example 20 percent); then it must decrease gradually for the successive values. Also we believe that a situation in which the comment value is around 80 percent is to be preferred to the total absence of comments. Therefore the value of the function, there, must be positive, even if its contribution is insignificant. In conclusion it is important to search a function with a new path similar to the following one in Figure 2.

The Proposed Model Modification

There are various solutions to this problem. In this article, a modification to the model of Coleman-Oman is proposed that leaves the structure of the original metric unchanged.

$$A\ sin \sqrt{K \cdot perCM}$$

The change is made by means of the following procedure. First of all it is necessary to determine the multiplicative coefficient K that better represents the contribution to the maintainability of the comment variable in the specific software system to be evaluated. Then, once the set of the values of the variable

$$sin \sqrt{K \cdot perCM}$$

has been calculated, it is possible to determine the parameter A = β_5 by Equation 3:

$$3)\quad MI = \beta_1 + \beta_2 \ln(aveV) + \beta_3\ aveV(g') + \beta_4 \ln(aveLOC) + \beta_5\ sin \sqrt{K \cdot perCM}$$

The problem of the parameters estimation consists in obtaining numerical values for the coefficients, i.e., values for the parameters multiplying each of the variables of the Equation 3, and for the intercept.

The estimate of β_5 (A) parameter must be obtained through the same calculation process used to determine the estimates of the other parameters of the model: the intercept, the parameters of ln(*aveV*), *aveV(g')*, and ln(*aveLOC*).

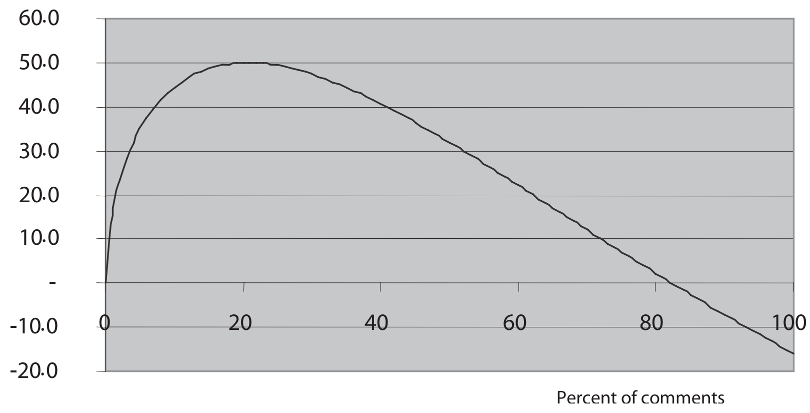
As for the determination of the coefficient *K* in the hypothesis, we neglect the *open reengineering* criterion[1]; various remarks can be made, and a great number of different factors can influence its calculation (language, programming support environment, etc.). We believe that the parameter *K* is mainly dependent on the used language. High-level languages generally require a greater percentage of comments with an equal number of developed lines of code. The diagram in Figure 3 shows the value of the component

$$\beta_5\ sin \sqrt{K \cdot perCM}$$

for values of *K* = 2.46, *K* = 0.12, *K* = 0.16, *K* = 0.20, in the case that the value of the parameter β_5 (amplitude) is equal to 50, which is the value set by Coleman-Oman in their model. Every language has its own characteristic curve; the curves shown in the diagram are obviously given by way of example, because investigation in this direction has not been carried out yet.

The main characteristics of these curves are that they differ slightly in the optimal value of the percentage of the comments; however, they differ greatly regarding the different values of comment percentage for which it is assumed that to exceed such threshold would have a negative influence on the maintainability.

Figure 2: *Ideal Path of Comments Contribution*



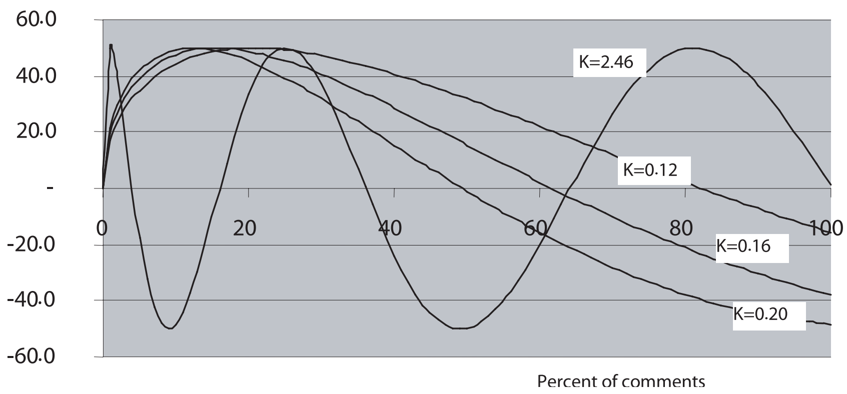


Figure 3: *Values of the Function:* $50 \sin \sqrt{K \cdot perCM}$

Conclusion

A method has been proposed to improve the Coleman-Oman model by modifying the contribution given by the comment component. The contribution to the improvement of the model of the β_5 parameter calculation is particularly significant since this parameter, with the new method proposed, will no longer be set from the outside. Instead it will be determined simultaneously by means of the same process of estimation adopted to determine the estimates of the other parameters of the model.

Future developments will therefore have to regard the estimation of the coefficient K, the estimation of the parameters of the modified Coleman-Oman model, and the experimentation of such model in industrial and public administration software systems.

References

1. Welker, K.D. and Oman, P., Software Maintainability Metrics Models in Practice, CROSSTALK, Vol. 8, pp. 19-23 Nov./Dec. 1995, www.stsc.hill.af.mil/crosstalk/1995/nov/maintain.asp
2. Coleman, D.; Lowther, B.; and Oman, P.; Using Metrics to Evaluate Software System Maintainability, *IEEE Computer*, Vol. 27(8), pp. 44-49, Aug. 1994.
3. Coleman, D.; Lowther, B.; and Oman, P.; The Application of Software Maintainability Models on Industrial Software Systems, University of Idaho, Software Engineering Test Lab, Report No. 93-03 TR, Nov. 1993.
4. Watson, A. and McCabe, T., Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, NIST, Sept. 1996, www.itl.nist.gov/div897/sqg/pubs/publications.htm

Note

1. The open reengineering concept is similar in that the abstract models used to represent software systems should be as independent as possible of implementation characteristics such as source code formatting and programming languages. The objective is to be able to set model standards and interpret the resultant numbers uniformly across software systems [4].

Additional Reading

1. Coleman, D., Assessing Maintainability, 1992 Software Engineering Productivity Conference Proceedings, Hewlett-Packard, pp. 525-532, 1992.
2. Oman, P. and Hagemeister, J., Constructing and Testing of Polynomials Predicting Software Maintainability, *Journal of Systems and Software*, Vol. 24(3), March 1994.

About the Author

Aldo Liso received the Laurea degree in Physics from the University of Bari. He is a senior member of the technical staff of Autorità per l'informatica nella Publica Amministrazione (AIPA) the Authority for I.T. in the Italy Public Administration. Liso has 20 years of experience working on software development projects and he is Professor of information systems at the DUSIT - University of Rome "La Sapienza."

AIPA
Via Isonzo, 21
00198 Roma, Italy
Phone: +39 6 85264 305
Fax: +39 6 85264 255
E-mail: liso@aipa.it

The Software Maintainability Index Revisited

Kurt D. Welker
Idaho National Engineering and Environmental Laboratory

In 1991 Oman and Hagemeister introduced a composite metric for quantifying software maintainability. This Maintainability Index (MI) has evolved into numerous variants and has been successfully applied to a number of industrial strength software systems. After nearly a decade of use, MI continues to provide valuable insight into software maintainability issues. This article presents some of the author's observations about the practical use of MI in determining software maintainability.

For many years now, software practitioners have been collecting metrics from source code in an effort to better understand the software they are developing or changing. Maintainability Index (MI) is a composite metric that incorporates a number of traditional source code metrics into a single number that indicates relative maintainability. As originally proposed by Oman and Hagemeister, the MI is comprised of weighted Halstead metrics (effort or volume), McCabe's Cyclomatic Complexity, lines of code (LOC), and number of comments [1, 2]. Two equations were presented: one that considered comments and one that did not.

The original polynimial equations defining MI are as follows:

3-Metric: $MI=171-3.42\ln(aveE)-0.23aveV(g')-16.2\ln(aveLOC)$

where *aveE* is the average Halstead Effort per module, *aveV(g')* is the average extended cyclomatic complexity per module, and *aveLOC* is the average lines of code per module.

4-Metric: $MI=171-3.42\ln(aveE)-0.23aveV(g')-16.2\ln(aveLOC)+0.99aveCM$

where *aveE* is the average Halstead Effort per module, *aveV(g')* is the average extended cyclomatic complexity per module, *aveLOC* is the average lines of code per module, and *aveCM* is the average number of lines of comments per module.

The rationale behind this selection of metrics was to construct a rough order, composite metric that incorporated quantifiable measurements for the following:

- Density of operators and operands (how many variables and how they are used).
- Logic complexity (how many execution paths are in the code).
- Size (how much code is there).
- Human insight (comments in the code).

Other variants of the MI have evolved using slightly different metrics, metric combinations, and weights [3, 4, 5]. Each has the general flavor of the basic MI equation and underlying rationale.

> *"Subjective measures applied via human code reviews still play an extremely important role in assessing software maintainability."*

Reasonable success has been achieved in using MI to quantify and improve software maintainability both during development and maintenance activities [4, 6, 7, 8, 9, 10, 11,].

Practically speaking though, the MI is only one piece in understanding the maintainability puzzle. Furthermore, it should not be interpreted in a vacuum. Rather it should be used as an indicator to direct human investigation and review. The following sections discuss some practical insight in applying MI to typical software systems.

Discussion of MI Equations

Several variants of the MI equations have evolved over time. Some academic opinion places more confidence in Halstead's Volume Metric than his Effort Metric so the MI equations were adjusted to incorporate the use of Halstead's Volume. Additionally, studies have shown that the MI model was often overly sensitive to the comment metric in the 4-Metric equation and thus that portion of the equation was modified to limit the contribution of comments in MI [11, 12].

The typical modified MI equations look similar to the following:

3-Metric: $MI = 171 - 5.2\ln(aveV) - 0.23aveV(g') - 16.2\ln(aveLOC)$

where *aveV* is the average Halstead Volume per module, *aveV(g')* is the average extended cyclomatic complexity per module, and *aveLOC* is the average lines of code per module.

4-Metric: $MI = 171 - 5.2\ln(aveV) - 0.23aveV(g') - 16.2\ln(aveLOC) + 50.0\sin\sqrt{2.46\,perCM}$

where *aveV* is the average Halstead Volume per module, *aveV(g')* is the average extended cyclomatic complexity per module, *aveLOC* is the average lines of code per module, and *perCM* is the average percent of lines of comments per module.

Examination of these equations indicates that picking the appropriate MI equation is still a subject for discussion[1]. The consideration of comments in the MI is a big discussion point. First, if a human assessment of the software concludes that the majority of the comments in the software are correct and appropriate, then the 4-Metric MI is potentially appropriate. Otherwise, the 3-Metric equation is probably a better fit. Second, if the 4-Metric equation is selected, it is still possible that the comments may inappropriately skew the MI. New research has been performed and additional modifications have been proposed to further refine the MI [13]. These refinements appear to add stability to the behavior of the MI for assessing specific types of software systems.

Observations

As mentioned earlier, numerous papers have been written describing the successful application of MI as part of the software development process or within a software maintenance assessment. What is often not discussed are some simple, common sense guidelines that should be considered when using an objective metric such as MI. Presented in the next few paragraphs are some general observations the author has gathered from applying MI to a variety of software systems.

Comments in the Code: Comments in the source code are a two-edged sword when it comes to considering their role in software maintenance. Accurate, up-to-date comments that provide additional insight not already obvious from the source code are generally quite helpful when it comes to making changes later on. However, comments that have not continued to evolve with their associated software can actually be a maintenance hindrance.

Comments, just like source code, will degrade over time as maintenance activities are performed unless specific actions are taken to keep them from becoming inaccurate. Comments are not executed at run-time; they are usually for people. Only people can tell if the comments in the code are helpful or not. Just because there are comments in the code does not mean that the code is more maintainable.

Furthermore, not all real comments are detectable by automated tools. For instance, when writing self-documenting code, some software developers put engineering units in the variable names (distanceFt or effectivePowerW). These types of comments are quite helpful to the human developer, ignored by the compiler and also most metrics extraction tools, yet certainly make the software more maintainable.

When determining how an automated tool will credit maintainability for comments in the code, a human must first determine the quality and usefulness of the comments. A variant MI equation could be developed that penalizes maintainability based on the poor quality of the comments. Current 4-Metric MI equations that include a metric for comments, must be applied with some human insight. Comparisons between the 3-Metric MI and the 4-Metric MI are also helpful in flagging source code with inappropriate comments. One guideline is that when there is more than a 15-point delta between the 3-Metric MI and the 4-Metric MI, comments in the associated source code should be manually examined for appropriateness.

Here is the bottom line on MI and comments: A man in the maintainability assessment loop is essential both in deciding how to measure comments in the source code (which MI equations) and then in determining the meaning of the results (do the comments make the software easier to maintain).

“Comments in the source code are a two-edged sword ... ”

Interpretation of Results: Source code metrics provide only an objective measure. MI works the same. Subjective measures applied via human code reviews still play an extremely important role in assessing software maintainability. After all people maintain the software. Automated maintenance is not a reality yet. Therefore, it only makes sense that there are some characteristics of software construction that take a person to quantify for attributes such as maintainability. Determining maintainability purely by objective measures can be deceiving. Take for instance the following simplistic example. Consider two versions of a program that print the words to “The Twelve Days of Christmas” (see Figure 1, page 20). Sorry but I don’t remember whom to credit for writing Example 1. A few metrics for the two versions are as follows in Figure 2, page 20.

Possible interpretations might include: Based on the 3-Metric MI alone, Example 1 is slightly more maintainable; based on cyclomatic complexity alone, Example 1 is more maintainable; based on lines of code, Example 1 is more maintainable, i.e. less code to maintain? Based on the 4-Metric MI, Example 2 is more maintainable, but did the comments really make the difference? No, the comments are okay but they are not the driving factor for judging maintainability. Based on effort alone, Example 2 is more maintainable than Example 1.

What made your decision regarding which source was more maintainable? It was probably the human examination of the source code. Which version of the source would you want to maintain? Especially when you learn that Example 1 contains a bug as the word eighth is misspelled in the output. This example may be extreme, but it illustrates the point. The metrics for real-world software can present similar difficulties.

Object Oriented Decomposition and Fracturing: Software languages, architectures, and decomposition techniques have evolved since the original development and validation of the MI equations. Object-oriented analysis and design have influenced the structure of today’s software systems. Typical object-oriented software tends to be decomposed into smaller modules than software systems that were decomposed using other techniques. Consider all the get and set methods in a typical object class. What impact does having a large number of smaller modules play in the MI?

Object-oriented software is fundamentally composed of operators and operands and has a number of executable paths through the code. The lines of code may still be counted and commented. From this perspective, MI still provides a good fit [11]. But additionally, there are constructs such as classes and inheritance that could be considered in tailoring MI for an object-oriented system.

Discussion of these types of MI enhancements will be postponed for another time. It appears, though, that object-oriented systems by nature have a fairly high MI due to the typical smaller module size. Naturally, smaller modules contain less operators and operands, less executable paths, and less lines of comments and code; therefore, the MI tends to be higher. It is the author’s opinion that even so, the MI is still applicable for object-oriented systems, but that maybe the maintainability classification thresholds should be raised when interpreting MI’s from object-oriented systems. MI is still great for identifying overly complex (and therefore difficult to maintain) modules in object-oriented systems.

Is it possible to decompose a software system into modules that are too small? Yes, software fracturing can occur and when that happens, modules loose cohesion and the coupling between modules

Example 1

```
#include <stdio.h>
main(t,_,a)
char *a;
{
return!0<t?t<3?main(-79,-13,a+main(-87,1-_,main(-86,0,a+1)+a)):
1,t<_?main(t+1,_,a):3,main(-94,-27+t,a)&&t==2?_<13?
main(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?main(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){)#}w'){){nl]'/+#n';d}rw' i;# \
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/")
 :t<-50?_==*a?putchar(31[a]):main(-65,_,a+1):main((*a=='/')+t,_,a+1)
  :0<t?main(2,2,"%s"):*a=='/'||main(0,main(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);
}
```

Example 2

```
/****************************
**
** Print the "Twelve Days of Christmnas"
**
*****************************/
#include <stdio.h>
main()
{
int i;
char *int_to_word();

  for (i=1; i<=12; i++) {

    printf ("On the %s day of Christmas my true love gave to me\n",
          int_to_word(i));

    switch (i) {
      case 12: printf("twelve drummers drumming, ");
      case 11: printf("eleven pipers piping, ");
      case 10: printf("ten lords a-leaping,\n");
      case 9: printf("nine ladies dancing, ");
      case 8: printf("eight maids a-milking, ");
      case 7: printf("seven swans a-swimming,\n");
      case 6: printf("six geese a-laying, ");
      case 5: printf("five gold rings;\n");
      case 4: printf("four calling birds, ");
      case 3: printf("three french hens, ");
      case 2: printf("two turtle doves\nand ");
      case 1: printf("a partridge in a pear tree.\n\n");
        }
    }
}

/****************************
**
** Convert the specified integer to English text
**
*****************************/
char *int_to_word (i)
int i;
{
  switch (i) {
        case 1: return ("first");
        case 2: return ("second");
        case 3: return ("third");
        case 4: return ("fourth");
        case 5: return ("fifth");
        case 6: return ("sixth");
        case 7: return ("seventh");
        case 8: return ("eighth");
        case 9: return ("ninth");
        case 10: return ("tenth");
        case 11: return ("eleventh");
        case 12: return ("twelfth");
  }
}
```

Figure 1: *The Twelve Days of Christmas - Sample Code*

increases. When fracturing occurs, the maintainability of the system (from a human perspective) actually decreases, and metrics such as MI are not necessarily a realistic measure of the actual maintainability. Controlled software development processes, good software engineering practices, and code reviews again become key in assuring and assessing maintainability. Thus it is evident that high MI does not guarantee the code is maintainable. A man in the loop is still essential.

Figure 2: *The Twelve Days of Christmas - Metrics*

	EXAMPLE 1	EXAMPLE 2
3 metric MI	87.4	86.7
4 metric MI	87.4	117.9
Effort	25668	6840
V (g')	13	27
LOC	17	54
Comments	0	10

Conclusion

Using the MI to assess source code and thereby identify and quantify maintainability is an effective approach. The MI provides one small perspective into the highly complex issues of software maintenance. The MI provides an excellent guide to direct human investigation. Hopefully, this paper provides some insight to the practical use of MI. To recap, continue to comment source code but do not put too much faith in comments to improve maintainability. Continue to measure maintainability using MI but do not interpret the results in a vacuum. Be aware of the limitations of objective metrics such as MI. Changing technologies will require changing metrics.◆

References

1. Oman, P.W.; Hagemeister, J.; and Ash, D., A Definition and Taxonomy for Software Maintainability, Technical Report #91-08-TR, Software Engineering Test Laboratory, University of Idaho, Moscow, ID, 1991.
2. Oman, P.W. and Hagemeister, J., (1992) Metrics for Assessing a Software System's Maintainability, Proceedings of the Conference on Software Maintenance, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 337-344.
3. Coleman, D., Assessing Maintainability, Proceedings of the Software Engineering Productivity Conference 1992, Hewlett-Packard, Palo Alto, CA, 1992, pp. 525-532.
4. Coleman, D.; Ash, D.; Lowther, B.; and Oman, P.W., Using Metrics to Evaluate Software System Maintainability, *IEEE Computer*, 1994, 27(8), pp. 44-49.
5. Oman, P.W. and Hagemeister, J., Constructing and Testing of Polynomials Predicting Software Maintainability, *Journal of Systems and Software*, 1994, 24(3), pp. 251-266.
6. Ash, D.; Alderete, J.; Yao, L.; Oman, P.W.; and Lowther, B., Using Software Maintainability Models to Track Code Health, Proceedings of the International Conference on Software Maintenance, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 154-160.
7. Coleman, D.; Lowther, B.; and Oman, P.W., The Application of Software Maintainability Models on Industrial Software Systems, *Journal of Systems and Software*, 1995, 29(1), pp. 3-16.
8. Oman, P.W., Applications of an Automated Source Code Maintainability Index, Technical Report #95-08-SL, Software Engineering Test Laboratory, University of Idaho, Moscow, ID, presented at the 1995 Software Technology Conference, Salt Lake City, UT.
9. Pearse, T. and Oman, P.W., Maintain-

ability Measurements on Industrial Source Code Maintenance Activities, Proceedings of the International Conference on Software Maintenance, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 295-303.

10. Welker, K. and Oman, P.W., Software Maintainability Metrics Models in Practice, CrossTalk, Nov./Dec. 1995, pp. 19-23 and 32.
11. Welker, K.; Oman, P.W.; and Atkinson, G., Development and Application of an Automated Source Code Maintainability Index, *Journal of Software Maintenance*, 1997, May/June, pp. 127-159.
12. Lowther, B., The Application of Software Maintainability Metric Models on Industrial Software Systems, master's thesis, Department of Computer Science, University of Idaho, Moscow, ID, 1993.
13. Liso, A., Software Maintainability Metrics Model: An Improvement in the Coleman-Oman Model," CrossTalk, Aug. 2001, pp. 15-17.

Note

1. The discussions in this article can apply to either set of MI definitions. The majority of people use the latter set of MI definitions. I still use the original MI equations for some applications. If used to track software over its life, it is important not to change equations mid-stream. There are other variants of the MI equations that organizations have tailored for specific interests (both the 3- and 4-metric versions). The discussion in the paper generally applies to most of these as well.

About the Author

Kurt D. Welker is an advisory engineer at the Idaho National Engineering and Environmental Laboratory with 14 years experience in software development, systems integration, and software measurement. He is a technical lead on the Electronic Combat System Integration Project performing reengineering, integration, and software maintenance on several electronic combat analysis models for the Air Force Information Warfare Center that simulate radar detection, weapon lethality envelopes, electronic counter-measures, reconnaissance, passive detection, and communications jamming. He functioned as the principle investigator for the development of a general-purpose lexical scanner/parser tool called the Data Stream Analyzer that provides data format integration. He also functioned as the principle investigator on a software measurement/process-improvement research initiative. He has been using MI to assess and track software maintainability for about eight years. Welker has a bachelor's of science degree in computer science from Brigham Young University and a master's of science degree in computer science from the University of Idaho.

**Idaho National Engineering
and Environmental Laboratory
Idaho Falls, Idaho
E-mail: wdk@inel.gov**

Letter to the Editor

Dear CrossTalk,

I was reading the new June 2001 issue Vol. 14 No. 6 yesterday and was nonplussed to read in three different places (From the Publisher, the abstract to the first article *Extending UML to Enable the Definition and Design of Real-Time Embedded Systems*, and the text of *The Quality of Requirements in Extreme Programming*), references to Universal Markup Language (UML).

All three of the contexts refer to the Unified Modeling Language created by Booch, Rumbaugh, and Jacobson of Rational Software Corporation. There is no real-time software design methodology called Universal Markup Language to my knowledge.

Thanks for an excellent publication.

Regards,
Karl Woelfer
Seattle, WA

Coming Events

August 27-30
Software Test Automation Conference
www.sqe.com/testautomation

August 27-31
5th IEEE International Symposium on Requirements Engineering
www.re01.org

Sept. 10-14
Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering
www.esec.ocg.at

Oct. 15-18
16th Annual SEI Symposium
www.asq.org/ed/conferences

Oct. 15-19
21st International Conference on Software Testing and EXPO 2001
www.qaiusa.com/conferences

Oct. 22-24
11th International Conference On Software Quality
www.asq.org/ed/conferences

Oct. 29-Nov. 2
Software Testing Analysis and Review
www.sqe.com/starwest

Nov. 4-7
Amplifying Your Effectiveness (AYE)
www.ayeconference.com

Feb. 4-6, 2002
International Conference on COTS-Based Software Systems (ICCBSS) At the Heart of the Revolution
www.iccbss.org

April 28 - May 3, 2002
*STC 2002
"Forging the Future of Defense Through Technology"*
www.stc-online.org

Proposal on Library-Centered Software Process Assessment

Toshihiro Komiyama, Toshihiko Sunazuka, and Shinji Koyama
NEC Corporation

This paper proposes a framework for Software Process Assessment and Improvement (SPAIM) and its performance measurement. The main purpose of this framework is to make SPAIM-related technologies adaptable to the features of an assessed organization such as organizational goals, future products, etc. The key to enact this framework is construction of SPAIM libraries containing various technologies for assessing, improving, and measuring software processes. Then, we can compose a specific SPAIM method adapted to the assessed organization by selecting and customizing technologies included in the libraries. This concept has been developed through more than 10 years of software process improvement experience in our company. In this paper, we first introduce our software process-related activities. Next we describe requirements to SPAIM that we perceived through the experience. Lastly, the proposed framework is explained.

Software Process Assessment (SPA) is an effective method used to understand software organizations' process quality and to identify issues to be resolved to achieve higher maturity. During the past 10 years, various SPA methods have been developed, proposed, and adopted. The first one was proposed by W. Humphrey, Software Engineering Institute at Carnegie Mellon University in 1987, called Process Maturity Model (PMM) [1, 2]. Then came the Capability Maturity Model® (CMM®) Ver.1.1, the Software Process Improvement and Capability determination (SPICE), Trillium, and others [3, 4, 5]. Recently, a series of international standards for SPA have been developed by International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) JTC1 SC7/WG10 and published as technical reports in 1998 [6].

Since 1990 various SPA-related activities have been conducted at NEC. In 1990 the SPA Working Group was organized to study the PMM. Core members studied *Managing the Software Process* [2] translated into Japanese. Then in 1992, the working group developed a SPA guidebook that contains questions, criteria, and guidelines for a PMM-based assessment [7]. In 1997 the base method was changed from the PMM to the CMM 1.1, and an overall revision was completed. In parallel, the working group members promoted SPA and acted as assessors to gain experience.

Our research group members have joined and led the SPA-working group activities. Additionally, to seek a more effective and efficient SPA method suited to NEC's organizational properties, we customized and applied several methods apart from the working group's activities [8, 9]. We also developed and applied original SPA methods, which are called SPICE9000 and Software Lifecycle Processes (SLCP)-based SPA. These incorporate the technical trend of SPA, or continuous model-based assessment, and an in-house opinion, i.e., synchronization of SPA and ISO 9000 internal audit. Also several tools have been developed for analyzing assessment data and enabling remote assessment using the network.

To date more than 30 in-house organizations have been assessed using various methods. More than 50 percent have taken multiple assessments periodically, and SPA has been getting diffused. Table 1 shows the history of SPA application that our research group members joined as lead assessors.

Original Methods and Tools

Here we introduce two methods and one tool that we originally developed.

SPICE9000: The SPICE9000 makes it possible to conduct a SPA and an ISO 9000 internal audit simultaneously. SPICE9000 is an integration of the first version of SPICE and ISO 9001 [10]. The assessment framework, the PMM, and the CMM are succeeded by SPICE9000. Relationships between elementary provisions of ISO9001 and practices of SPICE are clarified in Table 2. By using the relationship table both process assessment results such as capability level and ISO 9001 conformance can be obtained from assessment data.

SLCP-Based SPA: There was a discussion whether international standard software process (i.e., ISO/IEC 12207: SLCP) is or is not applicable to SPA in ISO/IEC JTC1 SC7/WG10, which is in charge of standardization of SPA. SLCP-based SPA answers this question and determines its possibilities. The standardized process model is also expected to suit broader types of organizations. SLCP-based SPA has a process model in compliance with the SLCP. To compose an assessment

Table 1: *History of SPA Application*

Org. ID	Date	SPA Method Applied	# of Question	Assessment Style	# of Project	Effort (hours/project)
Org. A	Sep.'95	PMM	85	Only Self	7	3
Org. B	Oct.'95	PMM	85	Only Self	7	3
	March'97	SLCP-Based (+ PMM)	196	Only Self	8	4
Org. C	Nov.'95	SPICE	1111	Only Self	2	2.5
	May '96	SPICE9000	1117	Only Self	1	2.5
	March'97	SLCP-Based (Detailed)	363	Self + Interview	3	2
Org. D	Feb.'96	PMM	85	Self + Interview	4	Self: 3 Int: 4
	March'98	PMM	85	Self + Interview	4	Self: 3 Int: 4
Org. E	Feb.'96	SPICE	1111	Only self	2	3
Org. F	April'96	CMM	124	Self + Interview	5	Self: 3 Int: 4
	April'98	CMM*	131	Self + Interview	2	Self: 3 Int: 3.5
Org. G	Sep.'96	CMM	124	Self + Interview	1	Self: 2 Int: 5
Org. H	May '97	SLCP-Based	163	Self + Interview	7	Self: 3 Int: 3.5
Org. I	Sep.'97	SLCP-Based	163	Self + Interview	7	Self: 3 Int: 6
Org. J	Nov.'97	CMM	124	Self + Interview	1	Self: 3 Int: 2.5
Org. K	March'98	SLCP-Based	163	Self + Interview	9	Self: 3 Int: 4
Org. L	April'98	CMM*	131	Self + Interview	5	Self: 2 Int: 3
Org. M	Oct.'98	CMM*	147	Self + Interview	3	Self: 2 Int: 4

*: Questionnaire is customized to suit the organization.

SPA Method	# of levels	Type of model	# of rating levels*	# of questions
PMM	5	Staged	2	85
CMM	5	Staged	2	124
SPICE	6	Continuous	2 or 4	1111
SPICE9000	6	Continuous	2 or 4	1117
SLCP-based	6	Intermediate	4	363

* 2: Yes or NO
4: Fully, Largely, Partially, or Not Adequate

Table 2: *Features of SPA Methods*

model from SLCP, the definition of tasks in SLCP is decomposed into sub-tasks and linked to capability levels.

Web-Based SPA Support Tool: A support tool for self-assessment was developed to conduct SPA efficiently [11]. This tool works on the World Wide Web and is mainly used for self-assessment, which is to be done prior to on-site interview. Participants can obtain their self-assessment results interactively in the form of a table or graph.

SPA Requirements

Various methods we applied are characterized in Table 2. Generally SPA methods are categorized into three types by the model shown in Figure 1. We gathered and analyzed assessments then drafted a report of the results for the organizations using different types of methods. Through these experiences we realized the benefits of each SPA method. However, we are still unable to decide which is the best method. One conclusion is that which SPA method is used is not as important as knowing how to use the selected method. We also concluded that the same requirements for conducting successful assessments are common to all SPA methods. These common requirements of assessment procedure, method, and tool are listed below.

Procedure:

Self – Assessment

- Specific questions should be developed prior to the interview to obtain consistently interpreted process features. If not, assessment data will not be reliable.
- In order to reduce the assessment load, there should not be too many questions. It is helpful to categorize questions by process domain, and ask each process domain's owners to answer the questions related to their domain.

On-Site Assessment

- Reserve a minimum of two hours for the interview to obtain a detailed process status, but do not go beyond one full day of questioning.
- If there are too many questions, they should be prioritized in order of importance, highest ranking questions listed first. Otherwise, they should be categorized by process domain, and all persons in charge of a process domain should be interviewed.
- A surveillance and interview of generic project's status during on-site assessment is helpful to obtain the information to prioritize questions and to draft process improvement proposals.
- Before interviewing, review documentation such as the development plan, progress report, and specifications to flag any interview questions and allow quick retrieval of any documents in question during the interview.

Reporting

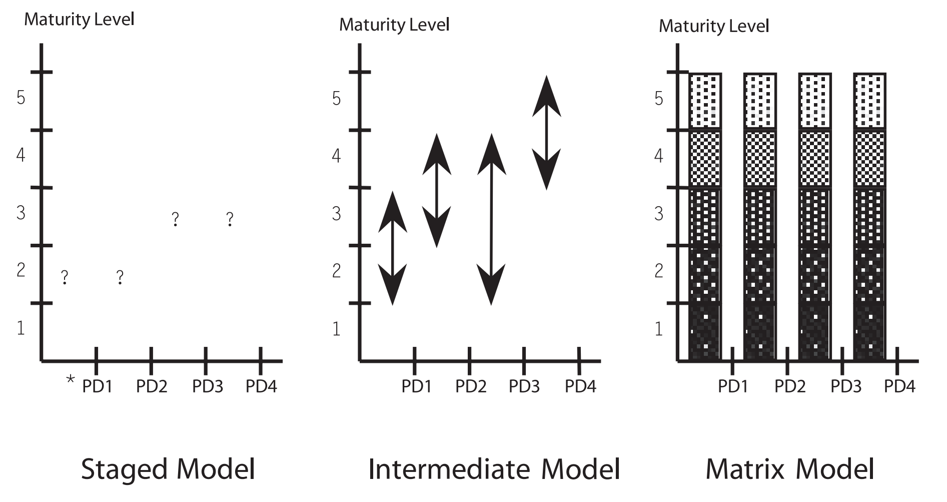
- An assessment report is composed of two parts. One part can be drafted systematically, e.g., assessment data analysis. The other part is a descrip-

  improvement proposals.
- Support tools to analyze and visualize assessment data are useful for the former, and libraries to store and share assessment know-how aid the latter.

Method

- There is no best SPA method. Methods should be easy to customize for each organizations' goals, needs, or properties.
- SPA methods should be usable for both self-assessment and on-site assessment. Also, it is desirable that the collected data are compatible between methods.
- The number of questions for one interview should be no more than 150, ideally less than 100. The wider coverage and the finer granularity, the better, but it makes a greater number of questions. They should be balanced.
- A well-structured questionnaire makes it easier to find correlations and get answers with fewer questions. It takes effect on saving interview time.
- The role or position of the interviewee should be clarified to get reliable and correct answers.
- A roadmap along with milestones for process improvement should be provided that prioritizes the established issues.
- A means to indicate the effects of process assessment and improvement quantitatively and objectively should be provided.
- The relationship between product quality or project results and process quality should be clarified based on analysis of project data and assessment results. It is helpful to prioritize the process improvement actions to be taken.

Figure 1: *Types of Assessment Frameworks* (*PD= Process Domain)



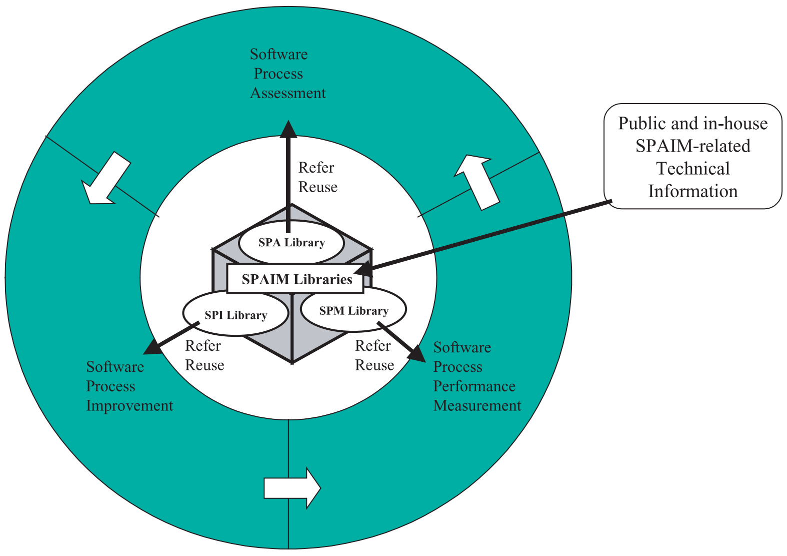

Software Process Assessment
Refer Reuse
Public and in-house SPAIM-related Technical Information
SPA Library
SPAIM Libraries
SPI Library
SPM Library
Refer Reuse
Refer Reuse
Software Process Improvement
Software Process Performance Measurement

Figure 1: *SPAIM Framework*

Support Tool

To conduct SPA effectively and efficiently, supporting tools are necessary. Required functions are as follow:

- Assessment data collection and analysis.
- On-line assessment support.
- Analysis and visualization of assessment data.
- Database of historical SPA data.
- (Semi-) Automatic assessment-report generation.
- Library of knowledge and experience on process assessment and improvement.

Reviewing the requirements above and looking back to the actual assessment situation, we concluded that the urgent and important requirements are summarized into the following:

- Adaptability: The possibility to adapt SPA methods to organizational needs, goals, and properties.
- Concreteness: The possibility to reach effective and concrete solutions based on assessment results.
- Validity: The possibility to validate the effects of process assessment and improvement activities.

We propose a framework that satisfies the above requirements hereafter.

The Framework

The SPA objectives differ by organizations, so one assessment method cannot suit all types of organizations. Therefore, it should offer selection and customization. Also, effective and concrete proposed solutions must be based on the assessment results in order to make them meaningful. Furthermore, the effects of process assessment and improvement must be indicated quantitatively and validated objectively. Here, we propose a software process assessment and improvement (SPAIM) framework to resolve these issues. A key strategy to implement the above requirements is to construct three libraries, i.e., one each for SPA, software process improvement (SPI), and software process performance measurement (SPM).

Overview of SPAIM Framework

An image of SPAIM framework shown in Figure 1 shows that technical information for SPA, SPI, and SPM are stored in SPAIM libraries. They are researched inside and outside of the organization and acquired. Then, they are reorganized into the structure shown in Figure 2 and stored in the libraries as record. When a new record is stored, consistency of terms and wording between records are coordinated to reach required records successfully.

Before starting SPA, organizational requirements for it are clarified. Then, process assessment methods and tools suitable for the organization are selected from the SPA library and appropriately customized. Any customized methods are added into the library for preparing for the next assessment, or for reusing it to the same type of organizations.

When an assessment has been completed, the data obtained is analyzed, and some primary process issues are identified. Practices, methods, and tools useful in resolving the issues are then extracted from the SPI library, and their concrete solutions are settled. If new improvement methods are developed, they are added into the library.

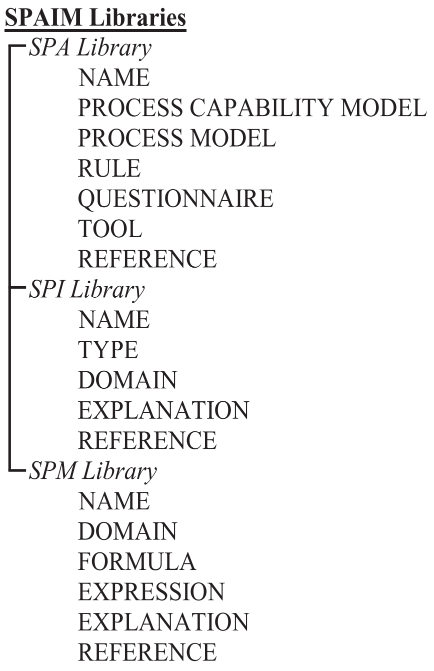
On the other hand, metrics used to measure the effects of process improvement are either developed new or selected from the SPM library accordingly considering the organizational goals for SPA and SPI. If new metrics are developed, they are added into the library for reuse. Selection of metrics and data collection should begin as early as possible.

In order to validate the effects of process improvement, it is necessary to measure not only the performance of improved processes but also that of current processes. Data required for calculating metrics should be continuously collected in parallel with process assessment and improvement. The collected data is aggregated into measures. Measures on current and improved process are compared, and the effects of SPI are validated objectively.

Architecture of SPAIM Libraries

Figure 2 shows the architecture of the SPAIM libraries, which contain information useful for SPA, SPI, and SPM. These libraries can be implemented as well-structured electronic files according to good naming convention or hyper-linked files. Structures of the SPAIM libraries follow.

Figure 2: *Architecture of SPAIM Libraries*



SPAIM Libraries

- *SPA Library*
  - NAME
  - PROCESS CAPABILITY MODEL
  - PROCESS MODEL
  - RULE
  - QUESTIONNAIRE
  - TOOL
  - REFERENCE
- *SPI Library*
  - NAME
  - TYPE
  - DOMAIN
  - EXPLANATION
  - REFERENCE
- *SPM Library*
  - NAME
  - DOMAIN
  - FORMULA
  - EXPRESSION
  - EXPLANATION
  - REFERENCE

SPA Library

Generally, the SPA method is composed of a SPA model and rules. A SPA model is composed of a process capability model and a process model. The former is a model that breaks down process quality into ordered capability levels, and further divides each level into features. For example, the process capability model of CMM Ver.1.1 is composed of five capability levels and five common features. The latter is a model breaking down whole software processes into several classes of process domains and work elements, e.g., the process model of CMM Ver.1.1 is composed of 18 key process areas and 316 key practices. The SPA method also includes some assessment rules.

There is never one best SPA method suited to all the assessment cases. This library contains various SPA methods, such as CMM Ver.1.1, SPICE and SLCP-based SPA, and also a method customized for an organization. This library makes it possible to select a SPA method from the variations. Elements of each method are structured and stored in this library. The following is the template to store this type of library information:

Name: The title of the SPA method such as CMM Ver.1.1 and SPICE are described. In organizationally customized methods, the organization name should be attached with the method name such as CMM Ver.1.1 for ABC Division.

Process Capability Model: Sets of identifier, title, and definition of process capability model components are described. Identifiers are assigned as showing classes of components. In the case of SPICE Ver.1.0, there are three classes of components, i.e., capability level, common feature, and generic practice.

Process Model: Sets of identifier, title, and definition of process model components are described. Identifiers are assigned as showing classes of components. In the case of SPICE Ver.1.0, there are three classes of components, i.e., process category, process, and base practice.

Rule: Assessment rules are described such as the rating for each question and a decision of process capability. In the case of CMM Ver.1.1, two levels of rating, i.e., Yes, No, Does not apply, or Don't know is adopted.

Questionnaire: A series of questions to be used for assessment are described.

Tool: Information of tools to be used for assessment is described. Assessment data collection, analysis, and visualization tools are necessary for each method. Data structure and algorithm of the tools depend on the model and rules of the method.

Reference: References related to the method are listed. They may be books, papers, reports, and Web addresses, which detail the method itself, its application results, and so on. Also, assessment reports using the method should be included.

"...the effects of process assessment and improvement must be indicated quantitatively and validated objectively."

SPI Library

A SPI library contains technical information to be used for the process improvement proposal, plan, and action. They are categorized into practice, method, and tool then stored with the same format in the library. Here, practice is a series of tasks to do some software-related job well. It can be expressed by workflow in conjunction with some additional information such as examples of standards, plans, forms, and checklists. Method and tool may be a research result, commercial product, or originally developed product. Process domain, on which a practice, method, or tool will have an effect, is attached with each record. This makes it possible to extract useful formats for resolving an identified process issue. By using this library, a more concrete process improvement proposal is possible. The following is the template to store information in this library:

Name: Title of practice, method, and tool for SPI such as Project Planning Procedure, COCOMO, and MS-Project are described. They should be uniquely identifiable. To do so, it may be necessary to attach an organization name or developer name such as Project Planning Procedure of XYZ Corporation.

Type: Type of the information is clarified, i.e., practice, method, or tool. To make it more specific, additional information like the technology area should be attached, for example Method for Cost Estimation.

Domain: Process domain, which is improved by the practice method, and tool, is described as Software Project Planning and Software Configuration Management.

Explanation: Description, figure, and formula that explain the practice, method, or tool are described. For example, workflow may explain practices well. A literal explanation with figures and formulas may suit some methods. Basic functions and operational environment should be used to explain least used tools.

Reference: References related to the practice, method, and tool are listed. They may be books, papers, reports, and Web sites, which address the practice, method, or tool itself, its application results, and so on. In case of tools, contact points of developers may be useful. Also, assessment reports using the practice, method, and tool should be included.

SPM Library

A SPM library contains metrics to be used for measuring process performance. Typically, it is measured from the view of quality, cost, delivery, and customer satisfaction, but not limited to these. By comparing process performances before and after process improvement, effects of the improvement can be validated. Also, metrics can be used to set goals for a series of SPA and SPI activities in conjunction with the target values. Furthermore, action items for process improvement can be prioritized by the strength of relevance to the achievement of goals. The following is the template to store information in this library:

Name: Title of metric such as Mean Time Between Failure and Average Cost Overrun, is described. As for metrics, there may be different metrics with the same name, e.g., fault densities with different size counts. This is not a problem because users will select a suitable one for his or her organization from the alternatives.

Domain: Process domain and/or type of process performance, which is evaluated by the metric, is described. There may be multiple domains to be evaluated with one metric. For example, in case of "Detected Fault Density of Design Review," both "Quality of Designing Work" and "Efficiency of Design Review" will be listed.

Formula: Formula of metric is described.

Explanation: This includes but is not limited to definition of data elements used in the formula, a means to collect the data, and interpretation of calculated values. It should be described precisely so as to be able to obtain an identical value independent of evaluators.

Expression: Visual expressions of a set of measured values using the metric are illustrated. Typical expressions may be bar chart, reader chart, and line chart.

Reference: References related to the metric are listed. They may be books, papers, reports, and Web sites, which address the metric itself, its application results, and so on. Also, assessment reports using the metric should be included.

SPAIM Library Procedure

The SPAIM procedure based on the framework is shown in Figure 3. SPAIM must be conducted continuously and recursively to pursue upper levels of process and catch up with software technology evolution or changing user needs. Each step of the procedure is explained below.

(1) Requirements Specifications for SPA and SPI

Requirements for SPA and SPI are specified and SPA method and SPM methods are settled. Requirements are identified from the following viewpoints.

Purpose: Purpose of SPA and SPI is specified. Typical purposes are to grasp current capability level, to identify and resolve issues on processes, and to improve process capability performance. There may be additional purposes such as to evaluate the effect of ISO 9001 certification.

Restriction: Limitations of resources for assessment are specified. Resources involve personnel, funds, and time.

Scope: Scope of SPA and SPI is specified. Organizational domain, process domain, and capability level can limit it. Even though a whole organization is assessed, not all the divisions and projects may be assessed. Also, some process domains may not be as important, not applicable to the organization, or out of scope. For example, if the organization develops software by itself, no acquisition process can exist. Furthermore, the capability range may be limited considering the current level of process capability, for example no higher than Level 3 for Level 1 organization.

Process Capability Goal: Goals on process capability are specified. Before setting them, it should be decided which SPA method to adopt for the assessment, considering both the requirements of the organization and the features of candidate methods. Goals are set using the selected SPA model. There are a variety of goal settings such as single goal on capability level for an entire software process or multiple goals on satisfactory level for each process domain. For example CMM Ver.1.1, the former can be expressed as "Level 3" and the latter as "satisfaction of goals on requirement management key process areas (KPA) and software project planning KPA."

Process Performance Goal: Goals to achieve process performance are specified. Goals may relate to process effectiveness, product quality, project results, customer satisfaction, and so on. They are broken into factors such as reliability for product quality. They can be further divided into lower factors and lastly into metrics. Finally, goals are shown quantitatively by the metrics attached with the target values.

(2) Software Process Measurement

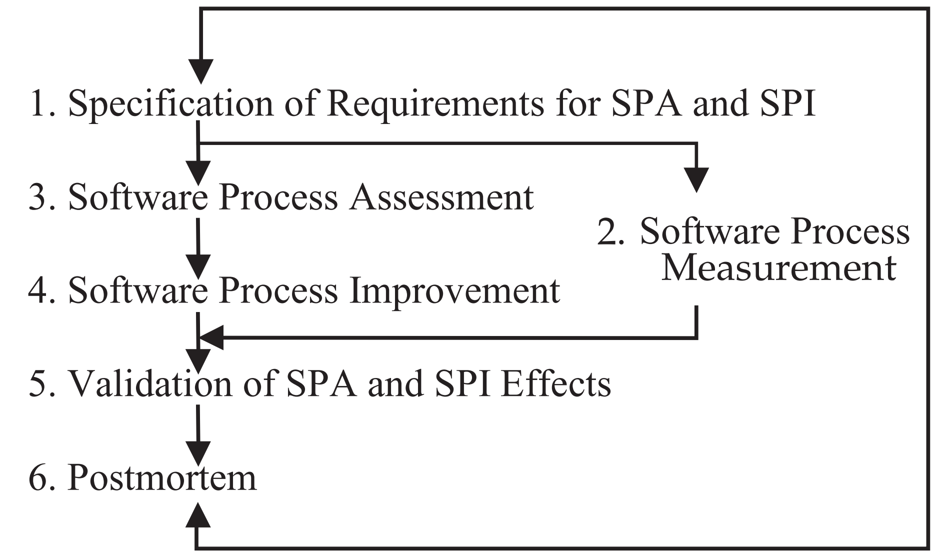
Just after the previous step, data collection for the selected metrics begins for evaluating current process performance. As for selected metrics, some data may have been collected. This data is gathered and made usable. All the necessary data is collected thereafter. Process performance is quantified by calculating the metrics using the above data. It will be done at a minimum when the assessment has been completed and analyzed in conjunction with assessment data. Measurement is conducted constantly in parallel with process assessment and improvement. All the measurement results are included in the submitted assessment report.

(3) Software Process Assessment

Assessment is conducted by the selected SPA method. At first, self-assessment is conducted. Personnel within the organization to be assessed answer the questionnaire themselves. The answer is a rated value such as Yes or No. Rating rules vary by method. After that, on-site assessment is conducted by interviews, and all the self-assessment results are confirmed and corrected if necessary. Both assessments refer to a SPA model. They indicate software process and illustrate the desirable practices level by level.

Confirmed rated values are aggregated into values for process domains, process features, and capability level. Those values are visualized in the form of graphs. Issues on processes are identified with these graphs and values, and also interview results. Issues are raised from the views of process domains and process features. Basic patterns to perceive issues are as follows:

Figure 3: *SPAIM Procedure*


1. Specification of Requirements for SPA and SPI
3. Software Process Assessment
2. Software Process Measurement
4. Software Process Improvement
5. Validation of SPA and SPI Effects
6. Postmortem

- Compare capability goals with the actual assessment results.
- Find process domains and features interfering to achieve the next level of capability.
- Compare values of the process domains and features with each other and find relatively lower ones.

All the assessment results are included in the submitted assessment report.

(4) Software Process Improvement

If too many issues are raised as a result of the assessment, they need to be prioritized. Criteria for the prioritization are as follow:

- Criticality to the current and subsequent projects' success.
- Harmfulness to process performance.
- An appropriate level of capability.

When primary issues are selected, effective practices, methods, and tools for resolving the issues are searched from the SPI library. The name of the process domain and feature are used as key search words for required information. Concrete proposals for process improvement are drafted using the disclosed information. All the proposals are included in the submitted assessment report.

All the findings and proposals written in the assessment report are explained to the personnel and management of the assessed organization in a meeting. They are discussed and consensus is reached. Once the consensus is made, an action plan is developed, and process improvement is carried out accordingly.

(5) Validation of SPA and SPI Effects

After planned improvement has been done and improved processes may have settled in the organization, the effects of improvement are measured and validated by the process performance metrics. A comparison is made between measured values on process before and after improvement. If the values of the metrics have been moving in a preferable direction, then the effects of SPA and SPI are validated. If not, the cause is analyzed, and a new action plan is drafted. Findings here are included in the assessment report at the appropriate time, or drafted separately in the improvement report.

(6) Postmortem

Looking back through a series of SPAIM activities, modified, acquired or newly developed practices, methods and tools for SPAIM are identified. Information about them is structured according to the architecture of SPAIM libraries and stored in the libraries. As for practices, methods, and tools used in a series of SPAIM activities, reference to the assessment or improvement reports are added.

Benefits of SPAIM Framework

By using the SPAIM framework, the following benefits are expected:

- The SPA method is suited for specific organizational goals and needs, and features can be adapted. This improves the accuracy of SPA results.
- More concrete action can be planned for process improvement by using stored knowledge and experience on practices, methods, and tools for SPAIM. This makes it sure to initiate process improvement.
- The goals and effects of SPA and SPI can be shown quantitatively and objectively using process performance metrics. This makes SPA and SPI goal-oriented activities, and makes it possible to check the adequacy of these activities and adjust them.

Current Status and Future Works

A prototype SPAIM support tool that partially implements a SPA library was already developed and presented [12]. It is a PC-based tool named Software Process Assessment supporT System (SPATS). It involves the components of SPICE and makes them customizable. The first version was developed on EXCEL and the second version on ACCESS. In order to make the SPAIM framework concept more effective and practical, enrichment of records in the SPAIM libraries is the most important issue. The following is the status of construction of SPAIM libraries.

SPA Library: The following information has been gathered, but structuring is underway.

- SPA methods: PMM, CMM Ver.1.1, Trillium, SPICE, SPICE9000, and SLCP-based SPA.
- Standards: ISO 9001, ISO/IEC 12207, and ISO/IEC TR 15504 series.
- The others: P-CMM, SE-CMM, and Malcolm Baldrige National Quality Award Criteria [15].

SPI Library: The following information has been gathered; structuring is underway.

- Process assessment reports (18 cases in Table 1).
- QC activity reports (more then 2,000 cases).

SPM Library: Investigation of process performance metrics is completed. Gathered metrics were categorized and tabled. Structuring is underway.

The first future direction this research will take is that the process domain of SPAIM libraries shall be expanded to system, people, and so on [13, 14, 15]. This makes it possible to treat a greater variety of process issues with the libraries. Secondly, the architecture of SPAIM libraries will be more precisely specified. This formalization will make it possible to store SPAIM-related information into a database and make better use of stored information. Hopefully, this architecture will be standardized, and the international public SPAIM database will be constructed. This will enhance sharing and exchanging knowledge and experience on SPAIM across organizations. Then the assets of software engineering will become more beneficial to the software industry. Finally, a relationship between process quality, product quality, and project results shall be clarified based on analysis of rich experimental data. This makes it possible to prioritize process domains to achieve SPAIM goals and to predict the effects of process improvement.

Summary

This paper introduced software process assessment and improvement activities in NEC. First, research products and an overview of assessment activities are introduced. Second, findings on SPAIM are listed. They are then summarized into primary features required for SPAIM. Third, the SPAIM framework having those features is proposed. The architectural and operational procedures of SPAIM are explained. Finally, future works on SPAIM are described.◆

Acknowledgments

We gratefully appreciate in-house colleagues who cooperated with our process assessment and improvement activities.

References

1. Humphrey W. S. and Sweet W. L., A Method for Assessing the Software Engineering Capability of Contractors, CMU/SEI-87-TR-23, 1987.
2. Humphrey W. S., *Managing the Software Process*, Addison Wesley, 1989.
3. Northern Telecom, Trillium Release3 - Model For Telecom Product Develop - ment and Support Process Capability, 1995.
4. Paulk M. et al., The Capability Maturity Model – Guidelines for Improving the Software Process, Addison Wesley, 1995.
5. ISO/IEC, ISO/IEC: Information Technology – Software Process Assessment – Part2: A Model for Process Management, Working Draft V1.00, 1995.
6. ISO/IEC, ISO/IEC TR 15504 Series: Information Technology – Software Process Assessment, 1998.13.
7. Nishimura T. et al., A Software Process Assessment Method Based on the Capability Maturity Model in NEC, Proceedings of IPSJ 52nd National Conference (in Japanese), 1996.
8. Honda, K. et al., Software Process Innovation Methodology – Multiple Approach Including ISO9001, Maturity Model and QC Techniques, NEC R&D, 1997, Vol.38, No.1, pp.96-104
9. Koyama, S., et al., Construction of Software Process Diagnosis Method – Software Process Assessment and Improvement Considering the Viewpoints of Management Quality, Proceedings of JUSE 18th Software Quality Management Symposium (in Japanese), 1998.
10. Koyama S. et al., Evaluation of ISO9001 Conformance Based on Software Process Maturity Model, IPSJ SIG Notes, 1996-SE-110, pp.17-24 (in Japanese)
11. Komiyama T. et al., WWW-Based Software Process Assessment Support Tools, 17th Software Reliability Symposium (in Japanese) 1997.
12. Omoto N. et al., Software Process Assessment Support System, IPSJ SIG Notes, 1995-SE-102, pp.159-164 (in Japanese).
13. Curtis, B. et al., People Capability Maturity Model, CMU/SEI-95-MM-02, 1995.
14. Kuhn D. A. et al., A Description of the Systems Engineering Capability Maturity Model Appraisal Method Version 1.1, CMU/SEI-96-HB-004, 1996.
15. United States Department of Commerce, Malcolm Baldrige National Quality Award – 1997 Criteria for Performance Excellence.

W e b S i t e s

India Software Network

www.e-isn.com

The India Software Network is a network of leading IT companies and software technology parks in India that has been in operation since 1998. It offers the IT services of companies in its network, which is a selective list of Indian software companies that have highly skilled manpower and excellent infrastructure. India Software Network maintains a database of each of these companies with regard to their specialization, manpower, infrastructure, finance, past projects, management, etc. Some of the companies in the network have SEI CMM Level 5 certification.

New Zealand Software Association

www.nzsa.org.nz

Members get support in management and marketing, export and education, finance and funding. Through conferences, courses, seminars, and visiting overseas experts, members learn more about these and other important subjects related specifically to the software industry. There are regular opportunities to meet informally, share knowledge, pool resources and simply get to know each other. The site features news, events lists, educational opportunities, a resource center, and more.

About the Authors

Toshihiro Komiyama is a manager of E-Learning Division at NEC Corporation He is a specialist of software product and process evaluation, and currently engaged in consultation of software process improvement as a SEI authorized Lead Assessor. He is also a secretariat of ISO/IEC JTC1 SC7/WG6. He has a bachelor's degree in mathematical science from Keio University and a master's defree in information systems from the University of Electro-Communications. He is a member of the IEEE and the IPSJ.

E-Learning Division
NEC Corporation
7-17, Shiba 2-chome, Minato-ku
Tokyo 105-0014, Japan
Phone: (+81-3)-5232-3080
FAX: (+81-3)-5232-3089
E-mail: komiyama@mve.biglobe.ne.jp

Toshihiko Sunazuka is a manager of E-Learning Division at NEC Corporation. He is currently engaged in consultation of software process assessment and improvement, and training of relative courses and personal software process. He has bachelor's and master's degrees in industrial engineering from Waseda University. He is a member of the IPSJ and the JSQC.

E-Learning Division
NEC Corporation
7-17, Shiba 2-chome, Minato-ku
Tokyo 105-0014, Japan
Phone: (+81-3)-5232-3080
FAX: (+81-3)-5232-3089
E-mail: sunazuka@mvi.biglobe.ne.jp

Shinji Koyama is a senior engineer of E-Learning Division at NEC Corporation with 10 years experience in research and development of software engineering. He is currently engaged in consultation of software process assessment and improvement to software organizations. He has a bachelor's degree in industrial engineering from Waseda University.

E-Learning Division
NEC Corporation
7-17, Shiba 2-chome, Minato-ku
Tokyo 105-0014, Japan
Phone: (+81-3)-5232-3080
FAX: (+81-3)-5232-3089
E-mail: s-koyama@mui.biglobe.ne.jp

Lessons Learned in Attempting to Acheive Software CMM Level 4

Al Florence
The MITRE Corp[1].

Getting to the Software Engineering Institute's Software Capability Maturity Model® Level 3 may be quite different than getting to Level 4. The forces, dynamics, commitments, and resources may be quite different for Level 3 than for Level 4. This article focuses on those differences and provides valuable lessons learned gathered on an organization that had achieved Level 3 but failed to achieve Level 4.

This article is based on an organization[2] that had achieved Software Capability Maturity Model (CMM) Level 3 and was working toward Level 4 [1]. Table 1 shows the Software Engineering Institute's CMM levels while Table 2 shows the key process areas (KPAs) within each level [2]. In order to be compliant with any level, an organization must be compliant with all KPAs at that level and all lower levels [2].

This author was the software manager and software engineering process group (SEPG) lead on project X when Level 3 was achieved[3]. Later, as SEPG lead at the next higher organizational level, he developed and executed Level 4 and Level 5 processes for project X. The projects in the organizations were geographically dispersed between both coasts and involved in diversified applications.

An organizational standard process (OSP) existed at the corporate level that only had processes for Level 2 and Level 3. The OSP was adapted and tailored to the projects as projects' defined process (PDP). SEPGs existed at various levels, and the corporation had a software-process training program that supported Level 2 and Level 3. All employees engaged in software development were required to take process training appropriate to their software tasks.

Level	Name	Characteristics
5	Optimizing	Continuously improving.
4	Managed	Quantitative control of products and process.
3	Defined	Management and engineering practices defined at the organizational level.
2	Repeatable	Basic project management established.
1	Initial	Ad hoc and often chaotic.

Table 1: *SEI SW CMM*

Getting to Level 3

While pursuing Level 3 all projects within the organization were committed and cooperated. The corporate SEPG had membership from the organizations' SEPGs and met monthly. The organization's SEPG met weekly and had membership from the projects. The projects' SEPGs meet weekly. The SEPGs coordinated on the OSP and the PDP and ensured that they were applied in a consistent and repeatable fashion across the organization.

Project X was required to follow Department of Defense (DoD)-STD-2165A, Standard for Software Development, along with supporting DoD standards, which provide for all processes and artifacts required for Level 2 and many for Level 3.

Processes for all Level 2 and Level 3 KPAs were installed and executed on the projects. Individuals received process training for both levels. Extensive Level 2 and Level 3 artifacts were collected. Several dry run assessments were conducted and supported with various government CMM Software Capability Evaluations for procurements.

The organization achieved Level 3 in 27 months after being awarded the contract for project X, which was never assessed at Level 2. The assessment was a Software Engineering Institute (SEI) CMM-based appraisal for internal process improvement (CBA-IPI). The lead assessor was from an external vendor while the rest of the assessment team was internal.

Not Getting to Level 4

Executive management mandated that the organization achieve Level 4. Unfortunately, while senior management was somewhat committed and cooperative, project X management was not. They stated that Level 3 was good enough, and that they did not sign up for Level 4. Project X personnel were also neither committed, cooperative, or involved except for the project SEPG lead and the project Software Quality Assurance manager. Project X's customer may not have even been aware of the Level 4 efforts.

Funding from the corporation, the organization, and the projects remained the same as for Level 3, which was insufficient for Level 4. Process staff did not increase more than what was provided for Level 3. Both funding and staff should have increased since new processes and training had to be developed and installed on the projects, and standards did not provide for process or artifacts like they did for Level 2 and Level 3.

The corporate SEPG was not involved with Level 4 activities at the time. The corporation did not have processes or training

Table 2: *KPA's of the CMM*

Level	Name	KPAs
5	Optimizing	Defect Prevention, Technology Change Management, Process Change Management.
4	Managed	Quantitative Process Management, Software Quality Management.
3	Defined	Organization Process Focus, Organization Process Definition, Training Program, Integrated Software Management, Software Product Engineering, Inter-group Coordination, Peer Reviews.
2	Repeatable	Requirements Management, Software Project Planning, Software Project Tracking and Oversight, Software Subcontract Management, Software Quality Assurance, Software Configuration Management.

for Level 4 or Level 5. The organization SEPG had membership from the projects' SEPGs, and they coordinated weekly. Level 4 coordination was very difficult between physically separate locations due to new processes being developed and difficulties encountered in their consistent application. Level 4 was not applied in a consistent or repeatable fashion across the organization.

The projects conducted insufficient Level 4 training, which lacked Level 4 corporate training material. The Level 4 and Level 5 training for project X was developed and provided by this author with little cooperation from project personnel. Project personnel were reluctant to attend training sessions. There were other hindrances:

- Software development standards on contract provided for all Level 2 processes and artifacts and many for Level 3, but not for Level 4 or Level 5. Software development standards do not address such things as quantitative analysis and continuous improvement.
- There was limited industry literature on Level 4 and Level 5 and few examples from which to draw.
- The organization conducted only one dry run for the level 4 assessment that surfaced some problems.

Although project X executed all KPA processes and collected extensive Level 2, Level 3 and Level 4 artifacts, the organization failed to achieve Level 4. The assessment performed was a SEI CMM CBA-IPI, with an external lead assessor and the rest of the assessment team internal to the corporation. There were few assessors that had conducted Level 4 and Level 5 assessments at that time. When assessors do not have appropriate experience with specific KPAs, it becomes difficult to arrive at consistent conclusions.

Reasons Level 3 Achieved but not Level 4

Here are some reasons the organization did not achieve Level 4:

- Commitment, funding, and cooperation existed at Level 3, but were not adequate for Level 4.
- Standards on contract provided for processes and artifacts at Level 3 but not for Level 4.
- All were involved with process improvement at Level 3; only SEPG members on project X were involved at Level 4.
- Level 3 was based on business goals, but Level 4 was done for process sake.
- There were many published examples for Level 3 but few for Level 4.
- There were many experienced assessors for Level 3 but not for Level 4.

Additionally Level 4 is a drastic paradigm shift from Level 3, however, this paradigm shift is not always recognized:

- Level 2 and Level 3 activities are common sense "things to do" in order to develop good software. Level 4 goes beyond this and is for organizations that really want to go the extra mile along the road to process improvement [3].
- Level 3 relies on existing software engineering and project management skills; new quantitative and statistical skills must be acquired for Level 4 [4].
- Level 3's main focus is on the organization, while Level 4's main focus is on the projects [4].
- At Level 3 measurements are used to status activities and correct problems. Level 4 requires measurements be quantitatively analyzed and that immediate actions be taken to remedy issues [2].
- Level 3 requires that process capability be institutionalized, while Level 4 requires that it be understood and controlled quantitatively [2].
- Level 3 requires that quality assurance be institutionalized. Level 4 requires that plans for quality goals are established and that progress towards achieving those goals be quantitatively managed [2].

Conclusions

Getting to CMM level 3 can be quite different than achieving Level 4. The forces, commitments, dynamics, and resources can be quite different, meaning possible success at Level 3 and perhaps failure at Level 4. Process improvement only works if everyone is committed, cooperative, and involved; and if proper resources are available, and improvement is based on business goals. Level 4 is a drastic paradigm shift from Level 3. New and additional skills are required at Level 4 (quantitative and statistical). Process improvement is not the sole responsibility of the SEPG. As with all CMM levels, the entire organization needs to be involved. It cannot be accomplished from outside the organization and the projects; it needs to be everyone's responsibility.

References

1. Florence, Al CMM Level 4 and Level 5 Approaches, 1999 SEPG Proceedings, Atlanta, Ga., Mar 1999.
2. Paulk, Mark C.; Curtis, Bill; Chrissis, Mary Beth; and Weber, Charles V., *Capability Maturity Model for Software*, V1.1, Software Engineering Institute, Feb.1993.
3. Florence, Al, Success at SW CMM Level 3 But Not at Level 4, Lessons Learned, 2001 Software technology Conference Proceedings, Salt Lake City, May 2001.
4. Perdue, Jeff, Why is Level 4 So Hard? Washington D.C., Software process Improvement Network, Nov. 2000.

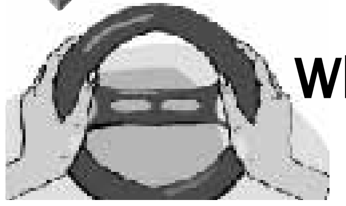
Notes

1. This article is not based on work done at or by MITRE. Any implications in this article should not be associated with MITRE.
2. When used, organizations and projects refer to the process at the organizational level.
3. When project X is used, the reference is only to that one project.

About the Author

Al Florence has worked at many high technology and aerospace companies and is currently at the MITRE Corporation. He has been involved in all phases of the life cycle as a developer and as a manager. He has developed processes for all CMM key process areas at all CMM levels and is a trained evaluator and assessor. He has a bachelor's degree in mathematics and physics from the University of New Mexico and did graduate work in computer science at the University of California in Los Angeles and at the University of Southern California.

The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102-3481
Phone: 703-883-7476
Fax: 703-883-1889
E-mail: florence@mitre.org



Who's Driving the Software World Anyway?

Let's face it folks, these days software is a lot like oxygen, we don't think about it a whole lot, but it pretty much keeps everything running. Perhaps a better analogy might be the electric power grid – with apologies to our California readers – of something that runs just about everything, but no one thinks about it. Until, of course, it breaks down; then not only do we think about it, we realize how much we depend on it. Then we inevitably get angry and a bit irrational. We do odd things like buy mail order windmills and photocells and listen to engineers on the radio. This is a frightening prospect for any community; but what if it affected all of us, and instead of the lights, it was the software that went south?

Do I hear a chorus of, "been there, done that"? True enough, the software-initiated have learned to live with the equivalent of rolling blackouts, but it can still get under your skin. Take my father-in-law's car for example. When purchased, it was a nice sort of upscale sporty model that had the requisite number of gizmos. Everything was electronic. It featured one of those on-board computers that could tell you everything about your drive to the grocery store. I thrilled my wife with the news that the stop sign at 49th Street and my lead foot had combined to produce an instantaneous gas mileage of .01 miles per gallon!

Of course the automakers have yet to take a lesson from the F-16 and add a heads-up display to this little number. So every time I felt the deep-seated urge to check my instantaneous vs. trip mileage, or see what the DTE was (I never did figure out what that meant), it was time to pray for a straight road. Fortunately, my wife's early warning system always seemed to distract me from the computer in time to avoid a total system crash. But even she could not save the operator from a feature called service monitoring.

I never got to see this service feature in action. That's because my father-in-law Jim did. It went like this: On a drive to work one day the service monitor announced to a surprised but grateful Jim that it was time to service the car. It did this with the usual visual warnings and a beep. This was good for Jim, who didn't know his car needed attention. It was soon bad for everyone else on the road because the beep never quit, and Jim became a heads down button-pusher – without the benefit of my wife.

One could understand this if the wheels were about to fall off, or if the engine was red-zoned. But this computer was pulling out all the stops for a check-the-fluids drill. Jim was reminded of his delinquent checkup all the way home, and therefore dutifully complied with his car's software demands at the earliest opportunity. After that, all was right with the software – and Jim's driving – for a few days. Then the demands started again – time for another service call. I believe it was on the third service visit that Jim took back control of his life from his car's software. Taking a cue from the classic *2001, A Space Odyssey*, he asked the mechanic, "Can't you just disconnect the @#%$# thing?" Jim has been driving happily with his neutered software ever since.

Another Detour

Irritating beeps in the car are one thing, but the really important software is much better than what is under the hood. Take the code that runs your bank; when it comes to money, the stuff has got to be bullet proof. That's what I thought until one afternoon when my wife's laughter caught my attention. She was opening the mail and had just happened upon an innocuous envelope from the bank with our new credit cards. These were those nifty new cards that include your picture on the front to help make your transactions more secure.

Thinking this a great idea, I had dutifully visited my local branch and had my picture taken. The lady at the bank was very nice and agreed that a second picture was warranted in my case, as the first exposure was sure to embarrass the kids at checkout time. Now I knew the second photo was nothing to post on the Internet, but neither did I think it merited the obvious entertainment my wife was enjoying. It was a gem of a photo, although not quite as I remembered it at the bank. I was now a woman in her thirties with nicely set hair.

Sure enough it was my credit card number and my digitized signature on the back, but a photo that would forever separate me from my children at the checkout counter. Embarrassing? Hey, this could drive the kids into therapy. I imagined my clergyman dropping by to see what he could do. I agonized over my next "may I see TWO forms of ID," experience.

Faced with such horrors, I called the bank hotline to see if I could get my face back. They explained that their system had now assigned this, well, rather attractive face to my account, and there was no way they could find my old digitized face. So the only thing to do was visit the nice lady in the bank again and hope for a more true-to-life exposure.

Now I realize that I may be denigrating my bank's software without due cause, after all, this may be how that nice lady gets through a long afternoon. But digitized pictures and databases gave cause for guilt, and software got the blame.

Now I could go on as there are more tales to tell, including my favorite of when one of my service providers upgraded their computer system then started giving me months of free service – despite my protestations. But really, despite my occasional crashes with software, things are well. Jim's car is still on the road, and the kids willingly let me use my credit card (with the old face). And I must admit that life is a bit nicer because of all that software doing such a great job. But then, once in a while I wonder, if my picture didn't get on my credit card, where did it go?

–Tony Henderson
Software Technology Support Center

CAPABILITY APPRAISAL

know your CAPABILITIES *before it* counts

You feel as though your process is out of control, but you cannot identify the problem. You need to fix something. You are over budget and behind schedule and your employees are over worked. What do you do? Where do you begin?

Call us. The *Software Technology Support Center* can provide capability appraisals at any level. We will bring the tools to tailor an appraisal to your exact needs, whether it's the Capability Maturity Model® (CBA-IPI, snapshot/quick look, or Independent Expert Program Review), the Capability Maturity Model Integrated℠ (SCAMPI®), program evaluation, or desk audit.

We get to the root of your problem and provide a solution that best fits your organization. You will receive an accurate baseline of how your organization measures up to the desired reference model, and what you need to do to improve your processes. Don't attempt a capability appraisal without a team of experts to support it and provide follow-up aid.

The STSC is a transition partner with the Software Engineering Institute and part of the product development team for CMMI℠. When problems begin, call us first. Whether your organization is big or small, just starting a project or embattled in difficulties, we can help.

OO-ALC/TISE 7278 4th Street Hill AFB, UT 84056 801 775 5555 FAX 801 777 8069 www.stsc.hill.af.mil

MISSION CRITICAL SOFTWARE CRSIP

Sponsored by the Computer Resources Support Improvement Program (CRSIP)

CROSSTALK / TISE
5851 F Ave.
Bldg. 849, Rm B04
Hill AFB, UT 84056-5713

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737