

Has Your Software Structure **DETERIORATED?**

FEATURED IN THIS EDITION:

Geriatric Issues of Aging Software * Measuring the "Health" of Software Structure
* The Millennial Tidalwave: Five Elements That Will Change the Workplace of Tomorrow *
Commentary on Rising Elephant * IT Management Briefing - Software Maintenance Challenges





Table of Contents

APRIL 2008
VOLUME 22, NUMBER 2

EDITORIAL BOARD

BOARD CHAIRMAN

Peter Wilson, Ph.D., CSQA, CSTE
Mosaic, Inc.

BOARD MEMBERS

Robert H. Goerss
Consultant

Kirby Fortenberry, CSQA, CSTE
Shell Services International

Charles Hollocker, CSQA
Process Technology Growth

Eldon T. Li, Ph.D., CPIM, CDE
California Polytechnic
State University

Peggy Myles, CSQA

**Rebecca Staton-Reinstein,
Ph.D., CSQA**
Advantage Leadership

Linda T. Taylor, CSQA
Taylor Tech Management

EDITORIAL STAFF

William E. Perry, CSQA, CSTE
QAI, Managing Editor and Publisher

Ngoc Huynh
QAI, Editor

*The Journal of the
QUALITY ASSURANCE INSTITUTE*

Copyright © 2008, is published quarterly
by Quality Assurance Institute (QAI)
with editorial and executive offices at
2101 Park Center Drive, Suite 200
Orlando, FL 32835-7614
407-363-1111
Fax: 407-363-1112
www.QAIworldwide.org
Individual copies

Features

Letter from the Editor

***Who Maintains Your Software Structure/Architecture?* 3**
by William E. Perry

***Measuring the Health of Your Software Structure.* 5**
by William E. Perry

***The Millennial Tidalwave:
Five Elements That Will Change The Workplace of Tomorrow* 11**
by Alicia Blain

***IT Management Briefing on the Challenges of
Effectively Managing Software Maintenance* 15**

***Software Assurance, Attacking Security Threats Head On* 18**
by Tom Ticknor

***Geriatric Issues of Aging Software.* 21**
by Capers Jones

***A Commentary on the Book Rising Elephant* 29**
by Denise Copening

Departments

Software Certification Update 4

Technology Brief 19

QAI e-Campus 20



Letter from the Editor

Who Maintains Your Software Structure/Architecture?

BY WILLIAM E. PERRY



Dear Colleague:

There are two components of software: structure and functionality. Users own functionality, that is, they specify functionality to ensure it meets their business needs. Users do not let functionality deteriorate.

The question that remains is “who owns the structure of the software?” Whoever it is, they are responsible for ensuring the ongoing effectiveness and efficiency of software structure. If no one owns the structure, no one maintains the structure. Unfortunately, the answer often is that no one owns structure.

Another often asked question is “if users pay to maintain software functionality, who pays to maintain its structure?” Using a housing analogy, if you rent an apartment or office suite, the landlord pays to maintain the building by incorporating into your rent sufficient funds to maintain the building. If this analogy makes sense, why doesn't IT incorporate into its costs to change functionality the monies to keep the software structure current?

Software maintenance offers quality professionals an opportunity to return extensive value to their IT organization. For example, if you could show IT management how to extend the life of software, two or more years what would that be worth to your organization? If you could reduce the cost of software maintenance by 10% with the introduction of new work practices, what would that be worth to your IT organization? This issue of the Journal focuses on ways to provide more value by improving the software maintenance program.

Sincerely yours,



William E. Perry
CEO, Quality Assurance Institute



SOFTWARE CERTIFICATIONS UPDATES

Software Certifications Board Announces Changes to the CSTE and CSQA Eligibility Requirements

At the annual meeting, the Software Certification Board recommended modifications to the eligibility requirements for the Certified Software Tester (CSTE) and Certified Software Quality Analyst (CSQA). The recommendation requires candidates to have a minimum of two years experience in the IT profession. The Software Certification Board voted to change the eligibility requirements effective June 1st, 2007. Candidates who apply on or after June 1, 2007 will be required to meet one of the following prerequisites:

1. 4 year college degree plus a minimum of 2 years experience working in the IT profession, or
2. 2 year college degree plus a minimum of 4 years experience working in the IT profession, or
3. 6 years or more experience working in the IT profession.

Candidates who attend a college or university that offers a 3 year degree program would be required to earn the 3 year degree plus have a minimum of 3 years experience in the IT profession.

Any applications submitted prior to June 1, 2007 will fall under the previous eligibility requirements, regardless of the exam date selected.

Software Certifications Board Announces Associate Level Certification Programs

The Software Certifications Board also voted to establish foundation level certification programs. The Certified Associate in Software Testing (CAST) and the Certified Associate in Software Quality (CASQ) have been designed specifically for individuals who have acquired a foundation level of knowledge in the areas of software testing and/or software quality assurance but who have either no experience or limited experience in applying that knowledge.

Software Certifications began accepting applications for the CAST and CASQ on June 1, 2007. To qualify for acceptance in the CAST or CASQ program a candidate must meet one of the following requirements:

1. 4 year college degree, or
2. 2 year college degree plus a minimum of 2 years experience working in the IT profession, or
3. 4 years or more experience working in the IT profession.

Candidates who attend a college or university that offers a 3 year degree program will be required to earn the 3 year degree plus have a minimum of 1 year experience in the IT profession.

The first examination for the Associate Level certification was given during the public exams in September 2007.

For more information on the Associate Level certifications, please visit the Software Certifications web site at www.softwarecertifications.org.

If you're already certified, don't forget to continue earning your CPE credits. Details of what is required to maintain your certification can be found on the Software Certifications website under "Recertification". A great way to earn 40 easy credits and to learn about the latest trends in the industry is to attend one of QAI's IT quality and testing conferences.

Visit www.QAIworldwide.org for conference details.





Measuring the Health of Your Software Structure

BY William E. Perry
CSQA, CSTE

If not addressed, software will start to deteriorate from the day the software is placed into production, and sometimes during the period leading up to implementation. The reasons for the deterioration include:

- Functions changed, structure not changed
- Programs grow in size and complexity
- Documentation not updated as software changes
- Turnover of maintenance staff
- Operations and/or technology/hardware changes

The dilemma that management faces is evaluating the cost differences between maintaining the old system and rewriting it. For example, system X may cost \$25,000 to maintain during the current year, but it would cost \$100,000 to rewrite it. Management views this as spending \$25,000 versus spending \$100,000. However, this thinking may be short-sighted. For example, the following factors also need to be considered:

- Is the annual cost of maintenance steady or is it increasing, and at what rate?
- Would the cost of operation drop significantly if the system was redesigned?
- Would the cost of maintenance drop significantly if the system was redesigned?
- Would additional benefits be accrued if the system were redesigned?
- Would hardware and software costs drop if systems were redesigned using newer technology?
- Could obsolete hardware and/or software be eliminated if systems were redesigned?

Some of these questions could be answered if management analyzed the usefulness of the current system structure; other questions require an analysis of the operating environment. For example, to determine whether a specific piece of hardware or software could be eliminated, management has to analyze the operating environment. There are features and characteristics unique to each operating environment, and they are difficult to generalize.

This article presents criteria to use in determining whether an existing application system should be continued in production or whether it should be rewritten or redesigned. The criteria are designed to provide information helpful in making this analysis. The criteria should be considered as input to the decision process, and not the decision process itself.

The criteria are designed to help IT management measure the health and well-being of an application system. As a system begins to deteriorate, the criteria help diagnose that deterioration. However, if an application system malfunctions, it does not mean that rewriting or redesigning is justified.

Management has two options when a system deteriorates in performance. One option is to rewrite or redesign the application system. The other option is to deal specifically with the characteristics causing the deterioration.

The criteria provided in this article to diagnose application system deterioration are provided as a representative, but not exhaustive, set of measurement criteria. Organizations should be able to add to this list of general criteria, as well as being able to add some specific criteria applicable only to their organization.

The criteria are described in Figure 1. Each criterion in Figure 1 is described individually on a **Criteria to Measure Need to Rewrite** form which contains the following information about each criteria:

- **Criterion name** - The specific criterion that can be used to measure the deterioration of performance of an application system.
- **Description of criterion** - A narrative explanation of the characteristics of the criteria that need to be collected to measure how well or how poorly systems are performing.
- **Data collection techniques** - The methods, tools, and techniques that can be used to collect the data used for measurement purposes.
- **Criterion strength** - The value of the criterion in measuring application system deterioration,
- **Criterion weaknesses** - The concerns or attributes associated with that criterion which may cause it to falsely represent the deterioration of the application system.

Using the Criteria

The system being evaluated to determine its health should be assessed using the "rewrite" criteria provided in the **Criteria to Measure the Health of a Software System** section of this article. This criteria may be supplemented by an organization's internally developed criteria. A self assessment checklist of a system's health is provided for this purpose (see Figure 1).

To use the checklist, the evaluator rates the application being assessed by each criterion for its applicability to that application. For example, Criterion #1 asked if the user has stopped making requests for changes. If that is true or nearly true, the criterion should be rated

“applicable,” and if the number of changes being requested is consistent with prior periods, the criterion should be rated “not applicable.” Each of the twelve criteria should be rated.

After all twelve criteria are rated, they should be scored as follows:

- Very applicable scores = 3
- Applicable = 2
- Not applicable scores = 1

Add up the score. The following provides an indication of the rewrite potential of the assessed application.

Score	Action
12-16	Rewrite unnecessary
17-22	Some reworking required
23-27	Major reworking required
28-36	Scrap the system

Criteria to Measure the Health of a Software System

Criterion #1 Users Stop Making Requests for Changes

Description of Change

When systems become too unresponsive to user needs, users find alternative means to satisfy their needs. Some develop informal sys-

tems which maintain the needed information. When these conditions occur, users stop using the application system and rely upon their informal systems. Thus, they stop requesting changes to the system.

Data Collection Techniques

If an organization has a formal system change request form, count the number of requests made. If the number of requests is abnormally low, or dropping, user personnel should be interviewed regarding the usefulness of the system in their area.

Criterion Strength

Business conditions continually change, and application systems must stay in synchronization with business conditions. Therefore, needed systems will undergo a series of continual changes. If these changes stop or significantly decrease, it may indicate that the system is out of synchronization with the user function.

Criterion Weakness

Having few requests for system changes can also indicate that the user is completely satisfied with the application system. Systems that have anticipated user needs may have built those capabilities into the system prior to the need. However, an interview with user personnel is a means of quickly pinpointing the cause of minimal (or no) system change requests.

McCabe CycloPath Coverage

Exclusive to McCabe. Essential to Your Success.

Think that the Statement, Branch, or even Boolean coverage you're using tells you all you need to know about your software testing? You could be risking more than your organization's next release.

Don't get blindsided. Maybe these techniques were enough in the past, but your increasingly complex code demands that you utilize better, more stringent, and logical coverage technology.

McCabe IQ provides the solution with "**CycloPath Coverage**" technology. CycloPath Coverage, based on the McCabe- authored Cyclomatic Complexity metric, shows you the exact number of tests required. Since complexity is highly correlated with errors, this means that your testing effort will be concentrated on your most error-prone code.

Only McCabe has CycloPath Coverage capability, but McCabe IQ also provides multiple levels of code coverage including Line/Statement, Unit Level, Integration Level, Branch, and Boolean, to meet any testing requirement you have.

Visit www.mccabe.com/quest or call 800-638-6316 for details.

Criterion #2

Number of Errors Detected by the Application System

Description of Criterion

During computer processing, data is continually checked for errors. Errors can result from improper input, inconsistent data relationships, failure to meet system criteria, data source unlocatable (e.g., if an employee cannot be located in the employee master file), and improprieties in output preparations.

Data Collection Techniques

For each detected error in an application system one or more error messages is produced. Each error message should represent an error for the purposes of this criteria. The error messages should be counted to produce a total number of errors detected. This could be a routine added to the application system; or if errors are placed on a tape or disk file and then printed, it could be counted either automatically during printing or manually after printing.

Criterion Strength

Errors can be indicative of complexity or unresponsiveness of an application system. Errors can be made because the system does not readily fit into the day-to-day needs of the organization. The higher the number of detected errors, the greater the need to rewrite the system to simplify it and to better integrate it into user processing.

Criterion Weaknesses

Errors can also represent data preparation by unskilled people, or people who have sloppy or unsupervised work habits. These types of errors can be reduced with additional training and supervision.

Criterion #3

High Number of Requests for System Change

Description of Criterion

If an organization has a formal system change request, the number of changes can be obtained by counting the requests. Without a formal system, the number of changes requested must be obtained from the application project team. The magnitude of the number of changes can be obtained by comparing two previous periods in the same system or other systems of similar size and complexity.

Criterion Strength

A change request indicates that the system is not satisfactorily meeting the needs of the user. The more changes, the less satisfactory the system is in meeting user needs. The high frequency of change may represent a serious problem in systems structure in meeting user needs.

Criterion Weakness

A great number of changes may also indicate changes in the conduct of business in the user area. In this instance the magnitude of change would not reflect upon the viability of the application system but, rather, indicates the volatility of procedures in the user area.

Criterion #4

High Maintenance Dollars

Description of Criterion

The magnitude of changes to an application system can be measured two ways. One way is the number of changes and the second way is to

present the dollar cost of those changes. Dollar cost is perhaps more representative of the size and complexity of the changes than is the number of changes.

One change in system X may cause considerably more than 25 changes in system B.

Data Collection Techniques

If project personnel charge their time and the computer time they consume to a maintenance budget, the cost of maintenance can be obtained from the budgetary process. The magnitude of the costs can be determined by comparing it to maintenance dollars for previous accounting periods on the same project, or measuring it against maintenance costs for other projects of similar size and complexity.

Criterion Strength

Maintenance costs are more indicative of the magnitude of change than is the number of changes. Thus, maintenance dollars should be a good predictor of the magnitude of change made to an application system.

Criterion Weakness

Maintenance dollars may be expended for error correction, system enhancements, and new requirements. The allocation of maintenance dollars among the three categories may indicate that new requirements and enhancements are consuming maintenance dollars to achieve new needs as opposed to correcting problems because the system is out of synchronization with the user area.

Criterion #5

Difficulty in Extracting Data from the Application System

Description of Criterion

Many user needs are one-time needs. Special types of extracts or reports are prepared to meet a specific need, which may or may not be repeated at a future time. The difficulty in providing users with this special information is indicative of the responsiveness of the application system to meeting user needs.

Data Collection Techniques

One-time user requests for information should be identified. If the program change request system provides this information, it can be collected from the system change request procedure. However, if the system does not provide this information, then either interview the application project team or the user to determine which changes are of a one-time nature. For each of these changes, determine the days required for satisfaction. Compare this with the length of time required to make similar changes in previous periods, or against other application systems of the same size and complexity.

Criterion Strength

As the data processing systems become more integrated into the day-to-day processing in the user area, the user must be able to satisfy special needs quickly. The more responsive the system is in satisfying those needs, the more valuable it becomes to the user.

Criterion Weakness

The number of days required to satisfy a user need may be more dependent upon the complexity of the extract than on the ability of the application system to meet user needs. Likewise, requests quickly satisfied may be easily satisfied, even though the system is deteriorating.



“ Quality is never an accident, it is always the result of high intention, sincere effort, intelligent direction and skillful execution; It represents the wise choice of many alternatives ”

Lets begin
the journey

 **Scalar USA**
Where quality is the quest...

www.scalarusa.com

Criterion #6 Application System is Difficult to Change

Description of Criterion

The difficulty of changing a system can be measured in the number of days required to make a change. The number of days should not include time in a queue awaiting systems and programming attention but, rather, the number of days from the time the change is started until it is ready to go into operation. The assumption is that in good systems changes can be installed quickly, while in poorly structured or poorly documented systems the change process becomes time consuming.

Data Collection Techniques

If a formal change process exists, and the project personnel record the day they start work on a change and the day that it is ready to go into operation, the information can be collected from those forms. Otherwise the information must be gathered by interviewing project personnel.

Criterion Strength

The amount of time to make a change is normally extensive if it is difficult to determine where to make the change, how to fit the change into the existing logic, and to test and document the change. The time required grows as the system deteriorates.

Criterion Weakness

The number of days to install a change can also be indicative of the magnitude of the change. Big changes can take a large number of days to install regardless of the effectiveness or ineffectiveness of the application systems structure.

Criterion #7 Cost to Process Transaction

Description of Criterion

The cost to process a transaction should be evaluated on a continuous basis. The criterion is really the cost trend to process a transaction. An increasing dollar cost is indicative of a deteriorating application system, while consistent or decreasing costs are indicative that the system is still viable and takes advantage of new technology.

Data Collection Techniques

The cost of processing a transaction requires two pieces of information: the number of transactions processed per run and the cost per run. Frequently, this type of information is available from job accounting systems. The complex part is to define what is meant by a transaction. For example, several computer records may encompass a single business transaction.

Criteria Strength

As computer technology becomes more efficient, the cost to process a transaction should decrease. However, if the systems structure is obsolete, or is not in synchronization with the hardware and software capabilities, the cost will increase. Thus cost to process a transaction should be a good predictor of systems obsolescence.

Criterion Weakness

The cost to process a transaction is also related to the complexity of the processing required for that transaction. Thus a change in the mix of transactions or an increasing complexity in processing may also result in increased cost.

Criterion #8 Obsolete Hardware/Software

Description of Criterion

The hardware and software that an organization uses changes over time. Systems developed using one generation of hardware and software may not operate effectively on a new generation of hardware and software. Each new generation makes many existing application systems obsolete.

Data Collection Techniques

To determine if changeovers in hardware and software have occurred, interview either computer operations personnel or a hardware/software acquisitions group if the organization has one. Interview project personnel to determine the type of technology required to operate their application system.

Criterion Strength

Operating systems using obsolete hardware and software may substantially increase the cost and complexity of operation. Occasionally these systems require special emulators or simulators which themselves may be costly to obtain and operate.

Criterion Weakness

Application systems which are written in high-level language, and are device-independent, can frequently be shifted to new hardware and software with minimal or no loss of performance.

Criterion #9 Application System is Complex to Operate

Description of Criterion

Difficult-to-operate application systems increase both the cost of operation and the potential for error. Complexity of operation can be due to poor systems design or change in operating methods both within the user area and the computer operations area. Thus, complexity can be a problem of computer operators and/or users of the application.

Data Collection Techniques

Statistics on complexity to operate normally have to be gathered from interviewing or observing computer operators and/or user personnel. However, occasionally reviewing the procedures required to accomplish work in the application system is sufficient to provide insight in the complexity of operation.

Criterion Strength

If a system is complex to operate, people may avoid using that system. Also, turnover in personnel will require extensive training and also subject the organization to a potentially high error rate.

Criterion Weakness

That a system is too complex to operate may be a subjective judgment. Items are only complex if there is an easier way to operate the application system.

Criterion #10 Application System Structure Out of Sequence With Data Processing Plans

Description of Criterion

Data processing organizations change methods, procedures and technology periodically. For example, an organization may move from batch to on-line data base systems. When these changes occur, application systems not using that technology can cause problems for data pro-

cessing management. For example, it is difficult keeping people interested in and adept at working with older methods and procedures.

Data Collection Techniques

The organization's long- and short-range plans should be studied and then a determination made which application systems are within the structure of those short- and long-range plans. This may require interviews with project managers and data processing management.

Criterion Strength

A change in methods, procedures or technology normally requires extensive investments of time and dollars. When there are exceptions to the normal procedures these continue to take an ever-increasing amount of effort over time.

Criterion Weakness

There may be little reason to change an effectively functioning application to a new method, procedure, or technology if that move cannot be cost justified.

Criterion #11 Cost of Testing System Changes

Description of Criterion

The cost of change may not be as indicative of obsolete systems structure as is the cost of testing. Testing provides insight into the complexity of getting changes operational. The more difficult the system is to maintain, the more difficult testing becomes.

Data Collection Techniques

The cost of testing can be measured in terms of number of tests, minutes of tests or dollar cost of tests. This information may be available from the departmental budgetary systems or computerized job accounting systems. The information collection technique should be the one simplest to perform.

Criterion Strength

The cost of testing provides a guide to the difficulty in making a successful change in an application system. The more difficult to make the change, the higher the probability that testing will become costly.

Criterion Weakness

A high cost of testing can also be attributable to sloppy programming or using programmers inexperienced in the application system or in the software systems in which the application and testing reside.

Criterion #12 Problems Encountered in Placing Changes Into a Production Status

Description of Criterion

The installation of a change in an application system has many risks associated with it. The change may not perform correctly, the changes may impact unchanged areas of the application, and users may not understand the change and therefore may make errors.

Data Collection Techniques

The amount of documentation recorded during the implementation process determines the ease of gathering data about encountered problems. Frequently this information must be gathered by interviewing users and operational and project personnel. In addition, project personnel should analyze error messages immediately following a change, additional change requests which may be related to correcting problems caused by the change, and changes taken out of production status for more testing.

Criterion Strength

The final success of the change process is a problem free conversion. Thus, if present personnel take shortcuts in making and testing changes, the change conversion process will be adversely affected. The application system may seem to be viable when in fact it is deteriorating. However, the test states of the system won't be apparent.

Criterion Weakness

Changes in the conversion process may be caused by conditions other than a deteriorating application systems structure. For example, it could be caused by unskilled operators and user personnel.

How To Use This Software Health Self Assessment

The primary purpose for conducting this self assessment is to cause IT organizations to focus on the state of their software struc-

tures. The assessment process is more valuable than the actual 'health' score produced if the result causes IT organizations to upgrade the structures of their software systems.

My recommendations, and one used by many IT organizations, involves:

- IT budgeting money for software structure improvement
- Project managers assessing the structure of the systems they manage
- Based on that assessment, project managers can more accurately bid on IT's structure improvement budget.

This process allocates money to those project managers motivated to improve the structure of the software systems they manage.

Figure 1. System Health Self Assessment

System Being Assessed _____

By: _____ Date: _____

Step	SYSTEM HEALTH CRITERION	ASSESSMENT		
		VERY APPLICABLE	APPLICABLE	NOT APPLICABLE
1.	Users stop making requests for changes			
2.	Number of errors detected by the application system			
3.	High number of requests for system change			
4.	High maintenance dollars			
5.	Difficulty in extracting data from the application system			
6.	Application system is difficult to change			
7.	Cost to process a transaction			
8.	Obsolete hardware/software			
9.	Application system is complex to operate			
10.	Application system structure out of synchronization with data processing plans			
11.	Cost of testing system changes			
12.	Problems encountered in placing changes into a production status			
	TOTAL CHECKED			



The Millennial Tidalwave: Five Elements That Will Change The Workplace of Tomorrow

BY ALICIA BLAIN

Jay was my first Millennial employee and even though I had managed many different types of employees over my 24 years in management positions in Corporate America, Jay was different in every way. What I've come to understand is that he represents the future workforce and in looking back I came very close to rejecting him and in doing so I would have closed the door to a sneak peek into the future.

I thought I had seen it all in my years of managing people and I felt there were no surprises left for me. Well, there was a surprise for me, and it came packaged in a 23 year old Millennial employee named Jay.

For those of you who are not familiar with the term "Millennials" don't feel bad because I was not aware of that term either. In fact, I was not very aware of any generational distinctions in the workplace until I was confronted by a person that I did not understand and that led me to an awareness that there are several generations that work side by side in the office and the more you understand each of them, the better you will be able to communicate and work with them and be more productive. So how many generations are there and why should you care?

For the first time in the history of Corporate America, there are four generations working together in the workplace. Here's a brief summary of the generations and what defines them.

The Veterans

This group is all about honor and respect. They were born before 1945, fought in World War II, suffered and survived the Great Depression. These were the creators of Corporate America whose values were steeped in hard work, honor, loyalty to your employer, following rules and saving a good portion of what you earned.

The Baby Boomers

This group is all about changing the status quo and the signs of success. Born from 1945 to 1962 (give or take a few years) at a time of great optimism and prosperity in the U.S and consequently think that life is full of possibilities. This was a generation that was catered to from birth and by its sheer size (over 78 million) continues to re-shape every phase of life it enters. Because there were so many of them competing for such few positions in the workplace, they became workaholics as they tackled more assignments to differentiate them from other Baby Boomers. This is a group that challenges the status quo, seeks equality and believes in personal gratification and paying one's dues. While in school they never had a computer.

Generation X

This group is all about self-reliance and skepticism. Born between 1963 and 1980 (give or take a few years), this generation saw every major

institution in the U.S. - from the government to sports to religion - face scandal. They watched corporations lay off their parents after they had worked so hard for their employer and in doing so, sacrificed going to their Xer children's ballet recitals and baseball games and spending any quality time with them. This is a small generation (about 45 million) who is understandably cynical after witnessing such gloomy events in their life. They are self-reliant, have a casual approach to authority and see work as just a job. Permanence is not something employers should expect from them. Personal computers were invented during their generation and as such, their comfort with technology distinguishes them from the two previous generations.

Then came the Millennials who are the focus of this article and who, in the near future, will be the largest group of individuals in the workforce.

Millennials

This group is all about cohesion and hope. Born between 1981 and 2000 (give or take a few years), they represent optimism for a better world. This is the second largest generation (over 76 million) in the history of the U.S. As the children of Baby Boomers, there is nothing this generation feels it can't do. They have been the most protected generation ever seen with a lot of structure and a lot of demands placed on their young lives. They are heavily team oriented due to having many after-school activities, optimistic, very civic minded and extremely confident in themselves and their abilities. They have a wonderful relationship with their parents. They feel their parents are "cool" and rely on them in all their key decisions such as school selection or employment interviews. Computers have always been around and they are the most highly multi-tasking generation in the workplace today.

The Millennials are coming to the workplace with very unique qualities that have never existed previously. There are 5 specific characteristics that I believe will not only shift the workplace but will have a profound affect on how companies do business and compete in the global market.

- 1. Technology.** What strikes me the most about working with Millennials is that the word technology has no real meaning to them. While all the generations that preceded the Millennials can understand and appreciate the great strides that have been made with technology, the Millennials can't. It is something that has been with them from birth and they have no concept of life before technology. For the Millennials, using a computer, a cell phone or any other device is not a talent to master, it is simply part of their life. In other words - it just is. They are the very first generation in the U.S. that have been completely immersed in technology and because it is second nature to them, it has and will continue to transform how they see the world and interact in it, how they gather and utilize information, how they live their lives.

This is a generation that expects to be always connected, the Internet is the vehicle of choice for any research to be done, question to be answered, information to be obtained. In my experience managing them, I have never seen a Millennial utilize another source other than the Internet to get information. It doesn't matter what you ask them to find, the Internet is where they go. They socialize via the Internet, they get their news from the Internet, they make their purchases on the Internet and research their homework on there as well. Their entire life is on their computer not on paper. Their handwriting for the most part is atrocious but who cares when no one is faster typing on a keyboard than they are!!!!

This fundamental mind shift away from viewing technology as technology but rather as just a part of life will make the workplace a very different place in the future. The structure of Corporate America was put in place decades ago and over time it has absorbed and incorporated technology into all of its processes and functions as a way of automating things and lowering costs but has the organization embraced technology as an integral part of its fabric? Maybe. Maybe not. For a Millennial, they have not had to absorb technology. They breathe it. Any work that is **not** performed and maintained strictly on a computer is strange. Keeping manual files and folders with information printed from the computer is strange. Having a group of people dedicated to helping employees with what they perceive to be very simple technical questions is strange. Holding meetings that people need to physically attend is strange to them. Seeing executives receive a printed copy of the Wall Street Journal is strange. To a Millennial, embracing technology is not a concept – it is in the very fabric of their being and as they enter the workforce in numbers they will push for it to be the same in the workplace. They will undoubtedly push for it in ways that we, those of us from previous generations, can't conceive or imagine. Yet if we are still in the workplace we will have to conform to it

2. Team Oriented. In all of my years of managing people, I have never encountered a generation that is so naturally team oriented as the Millennials. The reason is that we made them that way, right? From a very young age, they have been involved in some type of team activity whether it was in day care or in soccer practice or ballet or in class or in study groups, car pools, and on and on. They learned to appreciate and value interacting with their peers and will often rely on fellow Millennials for information. Next to the Internet, the Millennials use their network of friends as a way of finding out about something or researching something. They also help each other out without being prompted. I have witnessed many occasions when one Millennial will have a specific assignment and his Millennial teammates will automatically pitch in to help get the assignment completed. For a baby boomer this is unbelievable since we spent a good deal of our time trying to differentiate ourselves from our baby boomer peers. Unlike other generations, especially the baby boomer generation since it's the most similar in numbers, Millennials like to come together rather than compete against each other. They work extremely well in teams and are very cohesive and more importantly, they act as a team. They are natural collaborators. I believe that in the future workplace, the Millennials will come together to drive change WITHIN the organization instead of competing against each other for their own particular gain. They have a collective spirit that is refreshing and understand the concept that in working together things get done better, faster and for everyone's benefit. Rather than teach them to be better team players, Corporate America may have the challenge of getting them comfortable with individual decision making and personal responsibility.

3. Experiential Learning – this is a group that is not afraid of failure, they are not afraid to try again and get better at something. Why? Video games!!! Yes, video games!! Now the baby boomers and Generation Xers are aware of and have played video games and so they are probably wondering what's the big deal? Well there's a very big deal. The video game industry has grown leaps and bounds since Pac-Man. Today's video games are not only visually stimulating but mentally stimulating as well. I know we hear a lot of bad things about how video games are alienating our children and making them anti-social while they play these silly games alone in their room. First of all, they are not alone. Most of the video games today are connecting Millennials with other gamers in all parts of the world. Secondly, they are no longer silly games. These are very sophisticated, complex games that are teaching Millennials how to strategize, problem solve, make decisions and be accountable for those decisions. The scenarios that they convey on the screen are often real world scenarios that are making Millennials not only think about how they would handle a particular situation but it shows them the repercussions of that decision. To get to the next level, the Millennial has to show proficiency in many different areas and it's often very difficult to get to the next level. That is where the failure comes in. Instead of seeing it as failure, the Millennials see it as an opportunity to get better at something so they can get to the next level. Failure doesn't disappoint them, it gives them a reason to try again. To them, video games are the way they learn and they learn through experimentation and in a very visual, third dimensional manner. As this group begins to dominate the workplace, they will develop an environment of creativity, of experimentation, of trying new approaches to resolve problems that will shift the corporate mindset from being process oriented and procedure based to more flexible, resourceful with a emphasis on experimentation and creativity.

4. Racial and Ethnic Diversity - This is the most ethnically and racially diverse generation ever in the history of the U.S. One out of three or 30% of Millennials is ethnically or racially diverse. The expectation is that if you include the immigrant population, that number may rise as high as 50% in the future. Think about that! This is a group that will not only be happy seeing diversity in an organization but will almost demand it. Unlike the generations before them that were often not faced with much diversity growing up, in their schools, in their neighborhoods, in their families, this group has and they are completely comfortable with diversity. By being around kids of different ethnic and racial backgrounds, the Millennials have learned to appreciate, value and learn from the different cultures and customs of their friends. Instead of rejecting the differences, they welcome it and embrace the richness that this diversity provides in their daily life. This acceptance is also fostered by the friends they have made all over the world on the Internet. Also, unlike previous generations they have had the privilege of traveling quite a bit at very young ages which has also made them very aware of other customs and traditions. To them diversity has always existed and they don't know the hurdles that people and organizations had to go through to get where they are today. What this means to the future workforce is that Millennials will expect to work with diverse ethnic and racial groups and will favor corporations that have that. Diversity programs as they exist today will most likely either go away or change in scope since a large portion of that generation will already qualify for and embrace diversity.

5. Empowerment - This has been an extremely pampered, protected, and child-centered generation. Unlike Generation X who were for the most part the Latch Key Generation, whose parents worked such long hours that they had to become very independent and resourceful, the Millennials had parents that were very involved in their upbringing and structured a lot of their young lives. Millennial

parents have encouraged their children to question everything. They have involved the Millennials in all the major decisions facing the family and have asked for their input. Their parents have put a lot of effort in grooming them and preparing them for their future in the workplace. As a result, this generation has grown up feeling very empowered and ready to question and challenge the status quo with the support and backing of their parents. Given that they are the most highly educated generation and one that has had a great deal of pressure to achieve, they have a very different attitude when it relates to their place in the corporation. Unlike other generations, Millennials do not believe it is necessary for them to “pay their dues” to receive a promotion or raise. They should not have to work on menial assignments normally given to young employees to “break them in”. They are very confident in their abilities and want to make an impact in the organization as soon as they start. They don’t have a problem working hard but it has to be with a goal in mind. They want to use their creativity, ideas and education to contribute to the success of the organization – to be an integral part of the organization reaching a major milestone. They want to be of value and they need to feel their input is meaningful. They will look to their manager for this just as they did their parents. Managers of the future have to be very comfortable coaching and mentoring these young employees because often they are what keep the young employee motivated, engaged and wanting to stay with the organization. Millennials will look to their managers for guidance and leadership and also for the challenging work assignments that they feel they can do. This generation more than any other will have expectations of their manager and will tap into that relationship heavily and often. Organizations in the future will need to be ready for this empowered mindset and have managers that are trained as coaches or mentors that have the ability to interact well with their

direct reports not only to inspire and lead them but also to find ways to keep them challenged early on and motivated to stay.

Any one of these five elements alone or even the combination of a few together would not make the future workforce significantly different than it is today but when you bring all five together I believe they will be the driving force for change in the future. These elements do not exist in tandem in any other generation today so it is very difficult to know precisely how they will change the workforce in years to come. One thing is certain: The workplace will be very different because the generation that will populate it in great numbers IS very different and there is no reason to believe that this difference will not carry over into the workplace. I think the more open we are that change will take place and the more we try to prepare for it, the better we will all be and the more effective and positive the change will be. Change often happens slowly and imperceptibly and that is why it is good to be aware of it so that it doesn’t catch us by surprise. Now that you have this information, try to be more aware of the Millennials around you. See how they interact, how they think, how they absorb information, what is important to them. See if you can spot the 5 elements in their behavior and way of doing things. Being more aware of their unique qualities will give you what Jay gave to me – a sneak peek into the future. A future that promises to be what the present is not – something that we know or can take for granted. The future always brings uncertainty but the Millennials, with the five elements that have come together uniquely to shape them, will be certain to make it different – especially in Corporate America.

For further information on generational diversity in the workplace, please contact Alicia Blain at: ablain04@bellsouth.net

For 20 years, Parasoft has been empowering organizations to deliver better business applications faster. We achieve this by delivering quality as a continuous process throughout the SDLC—not just QA. The result is a sustainable process that delivers greater productivity and significantly fewer software defects.

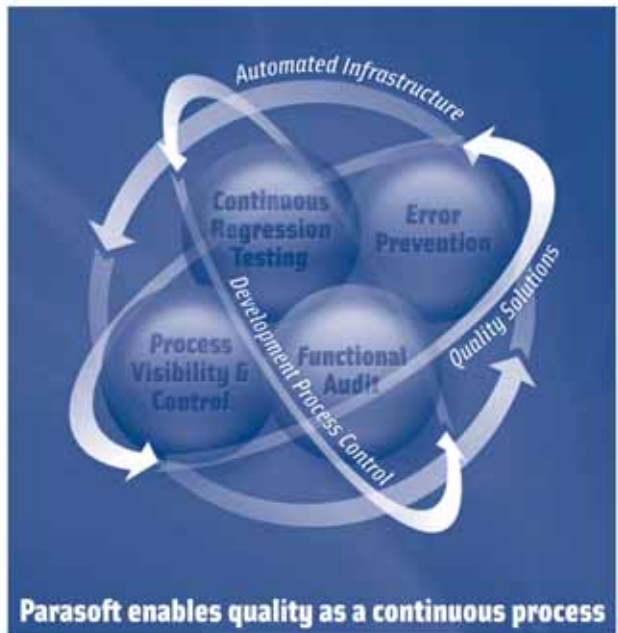
Parasoft solutions support:

Error Prevention: Parasoft delivers an automated framework to ensure all software development activities meet uniform expectations around security, reliability, performance, and maintainability. We provide a foundation for producing solid code by exposing structural errors and preventing entire classes of errors.

Continuous Regression Testing: Parasoft’s continuous regression testing immediately alerts you when modifications impact application behavior. This continuous quality practice provides a safety net that reduces the risk of change.

Functional Audit: Parasoft’s continuous quality practices promote the reuse of test assets as building blocks to streamline the validation of changing business requirements. This enables your team to execute a more complete audit of your business application.

Process Visibility and Control: SDLC quality metrics are fragmented across key systems such as requirements, build, and source control management. Parasoft aggregates and correlates this system data, delivering a comprehensive view of your development processes.

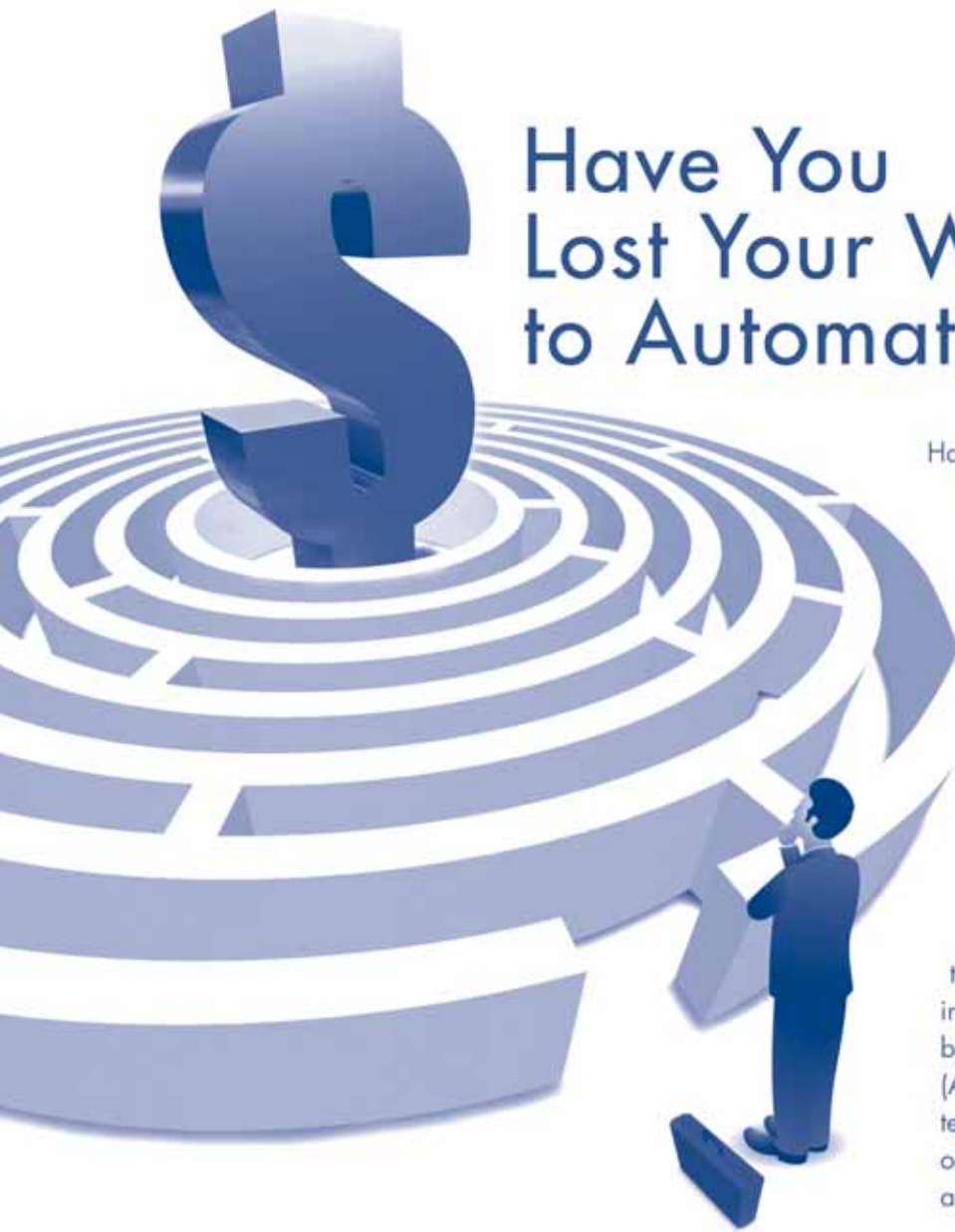


Parasoft enables quality as a continuous process
so you can deliver better software faster.



Parasoft Corporation
101 E. Huntington Drive, 2nd Floor
Monrovia, CA 91016
USA

Phone: 888.305.0041
Fax: 626.256.6884
Email: info@parasoft.com
www.parasoft.com



Have You Lost Your Way to Automation ROI?

Has what started as a straight forward investment in functional test automation degenerated into an endless cycle of maintenance and disappointment? Test automation often becomes a maze of higher than expected implementation costs and unfulfilled expectations. Utopia Solutions has the experience and expertise to help you realize the value and potential of test automation.

For over 15 years, our consultants have been successfully implementing test automation across a wide array of industries and technologies. Our services are built on our Automated Testing Framework (ATF) – a collection of proven processes, technology and knowledge focused on providing high-value test automation solutions.

Our Velocity™ Functional Test Automation (FTA) packaged services offer the fastest time to value for companies looking to rise above the chaos that can be test automation. Our solutions can be scaled to meet your needs. Whether you need an automation strategy, focused mentoring or a turnkey implementation, Utopia can deliver the experience, expertise and value you need.

Are You Ready to Benefit From Automation?

Find out with Utopia's online Functional Test Automation Assessment. Whether your company is just considering test automation or has already made the transition and is experiencing implementation issues this assessment will provide you with a non-biased, straight talk evaluation of your organization's readiness to truly benefit from test automation. The complimentary online assessment is based on our popular Velocity FTA Assessment packaged service – visit www.utopiasolutions.com/fta to learn more.

For more information on Utopia Solutions please contact:
Scott Clark (678) 775-6734

Utopia Solutions • www.utopiasolutions.com
Chicago • Dallas • Atlanta • Boston • San Jose

UTOPIA
SOLUTIONS

Business Partner 
INVENT



IT Management Briefing on the Challenges of Effectively Managing Software Maintenance

If software is built right the first time and users needs do not change, software maintenance is not be needed. However, this is not the case in the real world. Software is built with defects that need to be corrected, user's needs change, software architecture deteriorates, and technology changes. The activities required to keep software operational often exceeds the cost of building and acquiring new software. In performing these maintenance activities, IT management faces these three challenges:

Scope challenge – the types of maintenance that needs to be performed.

Process challenge – building the work processes needed to effectively maintain software.

Impediment challenge – the impediments that need to be addressed by software maintainers.

This briefing document explains these three challenges and offers solutions for each challenge.

System Maintenance Scope Challenges

Systems maintenance includes any activity needed to ensure that application programs remain in satisfactory working condition. This encompasses a broad range of activities involving error correction, need specification, system and program design, coding, testing, training, and monitoring the implementation of change. The management of maintenance is the management of change.

The broad interpretation of the maintenance function has resulted in the term being modified to indicate specific aspects of maintenance. For example, the following types of maintenance exist:

- **Corrective maintenance** - Maintenance that is conducted for the purpose of eliminating an existing error or problem.
- **Deferred maintenance** - Maintenance that is needed but is postponed until appropriate resources are available.
- **Preventive maintenance** - Maintenance that occurs in anticipation of problems. For example, an application may install a fix to prevent a problem that has already occurred in another application.
- **Emergency maintenance** - Unscheduled maintenance that is designed to eliminate a current problem situation which has stopped or otherwise affected successful operation of the application.
- **Data maintenance** - The addition, modification, or deletion of systems data specifications.
- **Program maintenance** - The addition, modification, or deletion of involved program instructions.
- **Structural maintenance** - Modifying the architecture structure of the software for maintenance efficiency, effectiveness, and change in technology.
- **Documentation maintenance** - Updating system and/or program documentation for system changes.
- **User documentation maintenance** - Updating user manuals to reflect system changes.
- **Scheduled maintenance** - Maintenance that occurs at a predetermined time.

From a management perspective, system maintenance is defined as all of the activities involved in keeping application systems working in a condition satisfactory to all involved parties. To perform software maintenance effectively, IT management must define the scope of software maintenance.

The increasing need for IT services directly affects the maintenance effort. Much of the demand placed upon IT involves changing the existing software applications. Problems associated with software maintenance places additional pressures upon IT management to satisfy all of their user needs on a timely basis.

System Maintenance Process Challenges

The systems maintenance environment is not always a controlled environment. IT departments have more difficulty in scheduling maintenance than they do in scheduling new systems development. This is because many of the systems maintenance requests (as the multiple definitions of maintenance imply) do not take place under the direction of the IT Department. Also, maintenance requests do not flow in an even stream, but usually in bunches, often at inopportune times.

Managing the maintenance function is tantamount to managing change. The process of managing change is subject to the following challenges:

- **Uncontrolled operating environment** - The operating environment includes all of the vendor-supplied software utilized to support applications systems. This includes programs, and so forth. Controls are necessary in this environment to ensure that the right version of the program is placed into operation at the proper time and to ensure that it interfaces with the proper files and other applications systems. The less controlled this environment is, the more responsibility the applications systems/programmers will have for maintaining the integrity of the operating environment. Insufficient controls in the operating environment can lead to applications system problems as well as ineffective application performance.

- **Insufficient maintenance program** - Maintenance, like many other segments of IT has grown rapidly in size and scope. What a few years ago appeared not to be a problem has grown in many organizations to be a function that consumes most of the IT resources. Many organizations are experiencing problems in controlling maintenance efforts. The result is the IT function is sometimes directed by forces other than the management of the function.
- **Poorly-designed systems and programs** - Maintaining well-designed application systems is a challenge for most systems analysts/programmers. As systems become more integrated and more complex, maintenance in one area of an application is more likely to impact another area. Many organizations still operate computerized applications developed many years ago. Many of these applications do not have today's structured systems design and extensive systems documentation. Maintaining these poorly-designed programs and systems is not only more time consuming, but also is more problematic.
- **Extensive user change requirements** - Many users are unfamiliar with the systems maintenance process. They neither understand the cost and effort required to make the requested changes nor do they know how to modify the change requests to make them more feasible. Without this type of understanding, many users issue a continual stream of change requests to the data processing department. If maintenance personnel do not have these requests, if they are late in honoring them, users become vocally discontent with IT.

IT management must address these process challenges in establishing an ongoing effective maintenance program. The key to success is in understanding the maintenance pressures and then in developing solutions that permit the maintenance process to function in an orderly fashion.

System Maintenance Implementation Challenges

Software maintenance personnel face several impediments as they strive to install changes. The more common implementation impediments and recommended solutions follow:

Impediment #1

- **Impediment** - Lack of maintenance information: Before they can effectively manage maintenance, organizations must have sufficient information about the function to establish effective solutions.
- **Solution** - Develop information collection procedures: Procedures need to be developed to collect information about the maintenance function. The accumulation of this information into maintenance statistics will enable management to track the function and make necessary adjustments.

Impediment #2

- **Impediment** - Unapproved changes made: Without proper controls, changes can be implemented through the request of users, management, and/or programmers. Any request may be honored.

- **Solution** - Develop maintenance approval procedures: Procedures need to be developed that ensure that no changes are made to application systems until those changes have been approved.

Impediment #3

- **Impediment** - Inconsistent approach to implementing system changes: Individual projects should not be permitted to establish their own methods of implementing changes.
- **Solution** - Develop standardized methods for installing changes: Procedures should be developed that provide a step-by-step standard approach for accepting, implementing, and placing changes into production.

Impediment #4

- **Impediment** - Users dissatisfied with implementation schedule: The timeliness of installing changes or the priority of installing changes proves unsatisfactory to users.
- **Solution** - Develop a priority system for implementing changes: The data processing project team in conjunction with the users should regularly review system changes and establish a priority for implementing those changes. Priorities may change, but the user will always be a major voice in establishing priorities.

Impediment #5

- **Impediment** - Repetition of the same error in multiple systems: A Type X error occurs in one system, only to be followed a short time later by the occurrence of the same error in another application.
- **Solution** - Develop a method of informing all systems and programming personnel of common errors. Analyze each error to determine whether it is unique to a specific application system or whether that type of error may be encountered in other applications. If the error could be common to many applications, inform systems programming personnel who will be affected.

Impediment #6

- **Impediment** - System difficult to maintain: The structure of the system and/or programs, or the lack of documentation make it difficult for the systems analyst/programmer to determine how and where to implement a needed change.
- **Solution** - Replace or retrofit difficult-to-maintain systems: The only solution to a poorly structured or documented application system is to redesign or retrofit that system so that it meets the current systems, programming, and documentation standards.

Impediment #7

- **Impediment** - Unable to adequately control the installing, operating, and deleting of different versions of application systems: The current method of assigning system and programming version numbers may make it difficult to perform maintenance. This also leads to a lot of "dead wood" in the source and object program libraries.

- **Solution** - Develop a controlled operating environment for installing changes: The source libraries should establish an efficient method for labeling different versions of the same program or system and for placing that into production at the required time while deleting obsolete versions.

Impediment #8

- **Impediment** - Problems occur when changes are installed: Changes cannot be installed without causing problems for users or operating personnel. Changes installed in one part of the system cause problems in another part or in another system.
- **Solution** - Perform regression testing, train users, and monitor the change process: Implementation problems are frequently associated with inadequate testing, training, and monitoring procedures. A successful maintenance function spends the time and effort required to assure that installation problems are minimal.

Scope of Systems Maintenance

The self-assessment questionnaire provides an overview of the scope of activities included in the maintenance function. Systems maintenance does not often have the same excitement as developing new applications. Newer systems are based on newer technology, thus they appear to provide the major data processing challenge. On the other hand, people concerned with systems maintenance may be using twenty-year-old technology, but it can be as challenging or more challenging as attempting to master new technology.

Software Maintenance Action Plan

Building an effective and efficient software maintenance program involves these three actions:

1. Define the scope of software maintenance activities – those activities that maintainers must perform such as updating system documentation.
2. Develop standards that must be met during maintenance activities and build the processes needed to meet those standards.
3. Identify the common impediments that maintainers may encounter and offer solutions to those impediments.

IBM

YOU CAN OPTIMIZE YOUR IT INVESTMENTS

Manage constant change. Meet high expectations. Control costs. Support core business strategies. Be more responsive. How can you innovate to continuously outperform and differentiate you from competitors?

Enter IBM. With a unique combination of industry expertise, business insights and proven technology such as Rational software, IBM can help align your company's IT infrastructure with business processes to meet core objectives. With accountability at the project management and results level, we can help deliver results, create an operating environment that optimizes IT investments, drive innovation and enable productivity and growth. To find out how you can build a resilient and flexible foundation for your business, visit ibm.com/solutions/itsolutions

Software Assurance, Attacking Security Threats Head On

BY TOM TICKNOR
CSTE, CSQA
Quality Assurance Institute

Businesses today are whole heartedly seizing the opportunities created by our quickly evolving environment of globalization. A rich expanse of new ideas, resources, cost savings, and markets lay open before the enterprising companies that shed old models and embrace the world as part of their operating structure. But, is the news all good? Unfortunately, the answer to this question is no. Together with the tremendous opportunities afforded by our new global view also come unforeseen and unprecedented risks to our infrastructure, our business community, our economy, and our national security. Risks that we, as a community of quality professionals, are in a unique position to acknowledge, understand, and act to mitigate. What exactly, though, is the nature of these threats?

The U.S. leads the world in information technology. However, the U.S. information technology industry has become more and more dependent on a complex, highly interconnected, and therefore, essentially anonymous network of suppliers, contractors, developers, and integrators from around the world. This fact must lead to serious concerns surrounding the nature of software security.

Consider the millions of lines of highly layered computer code in production at this moment. Now, consider the hundreds of thousands of additional lines going into production every day. This is computer code that drives everything from a toddler's learning game to the reservation systems of major airlines, the transaction systems of world-wide stock exchanges, and the weapons' systems of the Department of Defense. What more attractive target or better hiding place for those who wish to do harm or gather information for unscrupulous purposes. In a March, 2007, CSIS report on Foreign Influence on Software, author James A. Lewis states, "Some intelligence analysts believe that software offers one of the best mechanisms for technical intelligence collection by a range of adversaries."¹ Keith Rhodes, Chief Technologist at the Government Accountability Office indicates in a November, 2007, article in the *Government Computer News* that "We're long done with random acts."² The article goes on to assert that "Hacking today is the result of systematic exploitation by professional criminals who expect to earn a profit from their efforts."³

Sadly, the problem reaches deeper than vulnerability to intrusion created by weakness and flaws left in the code. Rather, the problem is as serious as intentional and malicious sabotage by the insertion of code expressly for the purpose of espionage and criminal activities. Moreover, this is not a threat than can be passed over because the code was developed down the hall in your IT department. What third-party software is imbedded in that code? What code generators or utilities were used? What about the operating system or the network software that will support the code, the coding language itself, the hardware, or the firmware? All of these elements are subject to the global IT supply chain and therefore susceptible to intentional and malicious sabotage.

It is imperative that, as quality process and testing professionals, we work together to reduce the risks associated with this situation. Software development processes must include strong procedures for software assurance. Testing practices must allow for comprehensive coverage of these security considerations. Currently, work is underway to understand training and certification requirements necessary to support information technology professionals as they deal with this critical and complex issue on a day to day basis. Working with the Department of Homeland Security, the Quality Assurance Institute's 2009 Software Quality Assurance Body of Knowledge workgroup and the 2009 Software Testing Body of Knowledge workgroup, are integrating, throughout these common bodies of knowledge, the skills necessary for today's software QA and testing professional to deal with the growing threat.

No longer can software safety and security be someone else's problem. Attending to this matter today in a responsible, proactive manner will ensure that as businesses move forward into the future as wholly global entities, they do so in stewardship for the safety and security of all nations and peoples.

For further information on this topic...

- **Build Security In** <https://buildsecurityin.us-cert.gov/>
- **The Software Assurance Community of Practice Portal** <http://www.us-cert.gov/swa>
- **SwA Common Body of Knowledge with Guiding Security Principles** <https://buildsecurityin.us-cert.gov/swa/people.html>
- **State-of-the-Art Report on Software Security Assurance** <http://iac.dtic.mil/iatac/download/security.pdf>
- **Software Assurance in Acquisition: Reducing Risks to the Enterprise, v1.0** <https://buildsecurityin.us-cert.gov/daisy/bsi/resources/dhs/908.html>
- **State-of-the-Art Report on Software Project Management for Software Assurance** <https://buildsecurityin.us-cert.gov/daisy/bsi/resources/dhs/906.html>
- **CSIS Report: Foreign Influence on Software** http://www.csis.org/media/csis/pubs/070323_lewisforeigninflubook.pdf
- **Common Attack Pattern Enumeration and Classification (CAPEC)** <http://capec.mitre.org>
- **Common Weakness Enumeration (CWE)** <http://cwe.mitre.org>
- **The National Vulnerability Database** <http://nvd.nist.gov/nvd.cfm>

1 James A. Lewis, "Foreign Influence on Software Risks and Recourse," March 2007, (Washington D.C.: Center for Strategic and International Studies) vii.

2 William Jackson, "Programmed for security: Two initiatives take on the challenge of providing better software development," *Government Computer News*, November 5, 2007, http://www.gcn.com/print/26_28/45342-1.html (accessed November 5, 2007).

3 Ibid.



TECHNOLOGY BRIEF

Deterioration of the Web – Rebuild It From Scratch

BY TIM PELLAND

OAI – Director of Technology and Education Services

How many of you would say that the Internet is irreplaceable? The Internet is a structural nightmare and getting worse by the second. Just today, you probably paid a bill, searched for something through Google and then fired off an e-mail to a co-worker. More than 1.2 billion people use the Internet every day. Imagine if the Internet were ever replaced. Why would anyone ever feel the need to restructure something that so many people have come to rely on for every day activities?

You can bet that the federal government and a very large segment of the computer science academy is doing just that at this very moment. Last May, the National Science Foundation (NSF) awarded BBN Technologies, the contract to build the new Internet from the ground up. This project is called GENI – Global Environment for Network Innovations. A wish come true for many who are confined by a system that was never designed for today's demands.

Why start from scratch? Otherwise, it would be like trying to keep an old car running. Maintaining an older model demands time and effort from the owner. As technology progresses, it increasingly becomes so advanced that the previous models are no longer compatible. In order to “save” this investment the owner ends up having to seek out and purchase updates, revisions and other specialized items just to keep the older version running. Eventually the owner knows they have to put the old model away and spend the money on a new one. The same goes for the Internet, and it has become apparent that it is time to put the old away and start the process to find a new model.

As radical as it seems to completely rebuild the Internet, the NSF decided the best solution is to start from scratch. That's right, a complete reconstruction from the bottom up, and I say it's about time. The current architecture was never designed to operate in the mode or capacity that is placed on it now. There have been many efforts to improve the Web over the years. In 1994 many universities, government agencies and non-profit organizations started their own private high-speed network known as Internet2. All were band-aids to the current structure without a thought of replacing the current infrastructure.

The Internet, as we end-users and technology geeks know today, is not going away overnight. It is projected the new Internet will basically look the same, but the infrastructure will operate in a different way. Right now when you want access to the Internet, you're required to use a telephone, cable wire, or go to your nearest WiFi hotspot to find a network. That would change because the Internet would be everywhere you are. If you have ever flown and had to go through an airline hub, that is like our current Internet structure. Any airplane trying to land at a hub during rush hour gets backed up and you have to wait in line for arrival. Internet data essentially does the same thing with every router it comes to. At certain points these data packets come to the same router, and like planes landing in rush-hour at a hub, data is forced to wait until its turn. Sometimes, data is forced to another router causing even more delays. Under this current method, sometimes the data gets there, but sometimes it doesn't.

Once BBN develops the new Internet, data will move faster by choosing direct paths and rushing straight through from the start and do so more reliably, like a non-stop flight to your final destination. Imagine a networked heart monitor that will alert you and your doctor that your heart has a problem and needs immediate attention. Under the current structure that same networked heart monitor would have to wait for all the e-mails about Viagra to get through the routers. The new structure would be based on urgent needs of the data. Less important items, like routine e-mails, can be set aside to allow for the transport of the critical data in much faster time increments.

The biggest challenge for the BBN is to enable capabilities that don't exist yet. You and I may not be around when the next generation of the Internet is implemented, but it could happen. Can you imagine the day when we have to explain to our children, “When I was your age we had to connect to the Internet through a landline cable tied to a wall.” And if BBN can make this new infrastructure work, Al Gore won't be able to take credit for inventing the Internet anymore.

If you have any additional comments about any of our programs or Web sites drop me a line at
tpelland@qaiworldwide.org,
I'd love to hear from you.

Software Deterioration



BY WILLIAM MARINARA
QAI – Director of e-Products

Bit rot, code rot, software decay - doesn't really matter what you call it - only that you know that all software will no longer viable at some point, and will need to be updated or replaced. Sometimes the deterioration problems are due to environmental changes such as the introduction of new operating systems or hardware, or it could be due to the changeover to a new programming language. However, more frequently the source of the problem is due to inadequate system design or the use of poor programming techniques. And finally but most importantly, a key factor in deterioration is caused by the misunderstanding of the user requirements.

From an educational standpoint, the solutions to some of these problems are straightforward. We can become an active member in a user's group or QAI chapter to learn about new operating systems or about new hardware from our peers. Similarly, taking advanced courses in system design and programming techniques can eliminate these as a source of deterioration. But we also have to be aware that like human languages, computer languages change over time and in some cases like Latin, some languages die (anyone remember Fortran or Pascal?). So it is vitally important to anyone who intends to stay in the software industry to keep a watchful eye and an open ear for any stirrings that would indicate the need to learn a new programming language.

In a past issue of the Journal we discussed the process of thinking outside the box - but not all problems require such a thought process. I think it's safe to say that all the programmers who developed applications prior to the mid 1990's knew that the year 2000 was going to arrive and yet very few incorporated a four-digit year format into their code. As a result, many of us spent long and late hours working to update code prior to the arrival of the millennium. A little forward thinking can go a long way in reducing the amount of deteriorating code.

As for scope creep, having processes in place that require board review and acceptance of any requirements changes can limit scope changes to those critical few and a post project review can assist in pointing to the causes of why those critical few were overlooked initially. How misunderstanding requirements can be minimized can take a bit of detective work.

Ellen Gottesdiener, in her white paper 'Software requirements: Using models to understand users' needs¹, wrote "To deliver the right product, you need to articulate your requirements early on, before the cost of fixing requirements errors kills your development budget, generates customer ill will and jeopardizes your business." She further states "Understanding user needs is both an art and a science — a combination of discovery, investigation and decision making. Successful projects engage users early and then explore and reach closure on their requirements by using *analysis models* — representations of user requirements."

An Internet search of 'Software Requirements Models' produces over 39 million hits so it's obvious that there are plenty of resources available.

A good starting point for many may be the NIST Baldrige National Quality Program web site Quality Reference Archive page². This page lists several past Baldrige award winners and their processes. Many of these can be modified to suit a variety of applications.

Another source for requirements modeling information is the QAI Process Warehouse. In the warehouse, for reducing the number of requirements changes to a project, one process model calls for the use of a structured interview to fully explore and identify the information needed to produce the end product that the client envisions. The chart below shows the information needed and who the best source of that information might be.

Knowing that misunderstanding the application's requirements is the largest contributing factor to software deterioration, it is equally important to understand that knowing where to find models that fit your needs is as much a needed skill as good programming techniques or in-depth knowledge of the latest technology. Regardless of your current position or level of influence, creating and maintaining an up-to-date list of models and their uses as well as any first hand knowledge of their implementation would be time well spent.

1. http://searchsoftwarequality.techtarget.com/tip/0,289483,sid92_gci1245443,00.html
2. <http://www.quality.nist.gov/Archive.htm>

Selecting the Person to Interview						
Categories of Workers / Information Needed	Senior Management	Department Management	First Line Supervision	Clerical Functions	Staff Functions	
User problems	X	X	X	X	X	
The User's mission	X	X			X	
Organizational policies, procedures, and laws	X	X				X
User policies and procedures		X	X			
Background information			X	X		
Undocumented events			X	X		
Explanation of problems		X	X			
Cause of problems			X	X		
Value of possible recommendations	X	X				



Geriatric Issues of Aging Software

BY Capers Jones
 Chief Scientist Emeritus
 Software Productivity Research, Inc.

Abstract

Software has been a mainstay of business and government operations for more than 50 years. As a result, all large enterprises utilize aging software in significant amounts. Some companies exceed 5,000,000 function points in the total volume of their corporate software portfolios. Much of this software is now more than 10 years old, and some applications are more than 25 years old.

Maintenance of aging software tends to become more difficult year by year since updates gradually destroy the original structure of the applications and increase its entropy. Aging software may also contain troublesome regions with very high error densities called "error-prone modules." Repairs to aging software suffer from a phenomenon called "bad fix injection" or new defects accidentally introduced as a byproduct of fixing previous defects.

Introduction

As the twenty-first century advances more than 50% of the global software population is engaged in modifying existing applications rather than writing new applications. This fact by itself should not be a surprise, because whenever an industry has more than 50 years of product experience the personnel who repair existing products tend to outnumber the personnel who build new products. For example there are more automobile mechanics in the United States who repair automobiles than there are personnel employed in building new automobiles.

The imbalance between software development and maintenance is opening up new business opportunities for software outsourcing groups. It is also generating a significant burst of research into tools and methods for improving software maintenance performance.

What is Software Maintenance?

The word "maintenance" is surprisingly ambiguous in a software context. In normal usage it can span some 23 forms of modification to existing applications. The two most common meanings of the word maintenance include: 1) Defect repairs; 2) Enhancements or adding new features to existing software applications.

Although software enhancements and software maintenance in the sense of defect repairs are usually funded in different ways and have quite different sets of activity patterns associated with them, many companies lump these disparate software activities together for budgets and cost estimates.

The author does not recommend the practice of aggregating defect repairs and enhancements, but this practice is very common. Consider some of the basic differences between enhancements or adding new features to applications and maintenance or defect repairs as shown in table 1:

Table 1: Key Differences Between Maintenance and Enhancements

	Enhancements (New features)	Maintenance (Defect repairs)
Funding source	Clients	Absorbed
Requirements	Formal	None
Specifications	Formal	None
Inspections	Formal	None
User documentation	Formal	None
New function testing	Formal	None
Regression testing	Formal	Minimal

Because the general topic of "maintenance" is so complicated and includes so many different kinds of work, some companies merely lump all forms of maintenance together and use gross metrics such as the overall percentage of annual software budgets devoted to all forms of maintenance summed together.

This method is crude, but can convey useful information. Organizations which are proactive in using geriatric tools and services can spend less than 30% of their annual software budgets on various forms of maintenance, while organizations that have not used any of the geriatric tools and services can top 60% of their annual budgets on various forms of maintenance.

The kinds of maintenance tools used by lagging, average, and leading organizations are shown in table 2. Table 2 is part of a larger study that examined many different kinds of software engineering and project management tools (1).

Table 2: Numbers and Size Ranges of Maintenance Engineering Tools
(Size data expressed in terms of function point metrics)

Maintenance Engineering	Lagging	Average	Leading
Reverse engineering		1,000	3,000
Reengineering		1,250	3,000
Code restructuring			1,500
Configuration control	500	1,000	2,000
Test support		500	1,500
Customer support		750	1,250
Debugging tools	750	750	1,250
Defect tracking	500	750	1,000
Complexity analysis			1,000
Mass update search engines		500	1,000
<i>Function point subtotal</i>	<i>1,750</i>	<i>6,500</i>	<i>16,500</i>
<i>Number of tools</i>	<i>3</i>	<i>8</i>	<i>10</i>

It is interesting that the leading companies in terms of maintenance sophistication not only use more tools than the laggards, but they use more of their features as well. The function point values in table 2 refer to the capabilities of the tools that are used in day to day maintenance operations. The leaders not only use more tools, but they do more with them.

Before proceeding, let us consider 23 discrete topics that are often coupled together under the generic term “maintenance” in day to day discussions, but which are actually quite different in many important respects (2):

Table 3: Major Kinds of Work Performed Under the Generic Term “Maintenance”

1. Major Enhancements (new features of > 20 function points)
2. Minor Enhancements (new features of < 5 function points)
3. Maintenance (repairing defects for good will)
4. Warranty repairs (repairing defects under formal contract)
5. Customer support (responding to client phone calls or problem reports)
6. Error-prone module removal (eliminating very troublesome code segments)
7. Mandatory changes (required or statutory changes)
8. Complexity or structural analysis (charting control flow plus complexity metrics)
9. Code restructuring (reducing cyclomatic and essential complexity)
10. Optimization (increasing performance or throughput)
11. Migration (moving software from one platform to another)
12. Conversion (Changing the interface or file structure)
13. Reverse engineering (extracting latent design information from code)
14. Reengineering (transforming legacy application to modern forms)
15. Dead code removal (removing segments no longer utilized)
16. Dormant application elimination (archiving unused software)
17. Nationalization (modifying software for international use)
18. Mass updates such as Euro or Year 2000 Repairs
19. Refactoring, or reprogramming applications to improve clarity
20. Retirement (withdrawing an application from active service)
21. Field service (sending maintenance members to client locations)
22. Reporting bugs or defects to software vendors
23. Installing updates received from from software vendors

Although the 23 maintenance topics are different in many respects, they all have one common feature that makes a group discussion possible: They all involve modifying an existing application rather than starting from scratch with a new application.



Alternative Thinking About Quality Management:

MAKE FORESIGHT 20/20

Don't miss HP's "Application Quality & Performance Testing" session at the Chicago QUEST Conference on Thursday, May 1, at 10:15a.m. Product Solutions Track.

HP Software
Proud to sponsor: QUEST Chicago 2008

hp.com/go/quality

© 2008 Hewlett-Packard Development Company, L.P.

Each of the 23 forms of modifying existing applications has a different reasons for being carried out. However it often happens that several of them take place concurrently. For example, enhancements and defect repairs are very common in the same release of an evolving application. There are also common sequences or patterns to these modification activities. For example, reverse engineering often precedes reengineering and the two occur so often together as to almost comprise a linked set. For releases of large applications and major systems, the author has observed from six to 10 forms of maintenance all leading up to the same release!

Geriatric Problems of Aging Software

Once software is put into production it continues to change in three important ways:

1. Latent defects still present at release must be found and fixed after deployment.
2. Applications continue to grow and add new features at a rate of between 5% and 10% per calendar year, due either to changes in business needs or to new laws and regulations, or both.
3. The combination of defect repairs and enhancements tends to gradually degrade the structure and increase the complexity of the application. The term for this increase in complexity over time is called “entropy.” The average rate at which software entropy increases is about 1% to 3% per calendar year.

Because software defect removal and quality control are imperfect, there will always be bugs or defects to repair in delivered software applications. The current U.S. average for defect removal efficiency is only about 85% of the bugs or defects introduced during development (3) and has stayed almost the same for more than 10 years. The actual values are about 5 bugs per function point created during development. If 85% of these are found before release, about 0.75 bugs per function point will be released to customers. For a typical application of 1000 function points or 100,000 source code statements, that implies about 750 defects present at delivery. About one third, or 250 defects, will be serious enough to stop the application from running or create erroneous outputs.

Since defect potentials tend to rise with the overall size of the application, and since defect removal efficiency levels tend to decline with the overall size of the application, the overall volume of latent defects delivered with the application rises with size. This explains why super-large applications in the range of 100,000 function points, such as Microsoft Windows and many enterprise resource planning (ERP) applications may require years to reach a point of relative stability. These large systems are delivered with thousands of latent bugs or defects.

Not only is software deployed with a significant volume of latent defects, but a phenomenon called "bad fix injection" has been observed for more than 50 years. Roughly 7% of all defect repairs will contain a new defect that was not there before. For very complex and poorly structured applications, these bad-fix injections have topped 20% (3).

Even more alarming, once a bad fix occurs it is very difficult to correct the situation. Although the U.S. average for initial bad-fix injection rates is about 7%, the secondary injection rate against previous bad fixes is about 15% for the initial repair and 30% for the second. A string of up to five consecutive bad fixes has been observed, with each attempted repair adding new problems and failing to correct the initial problem. Finally the 6th repair attempt was successful.

In the 1970's the IBM Corporation did a distribution analysis of customer-reported defects against their main commercial software applications. The IBM personnel involved in the study, including the author, were surprised to find that defects were not randomly distributed through all of the modules of large applications (4).

In the case of IBM's main operating system, about 5% of the modules contained just over 50% of all reported defects. The most extreme example was a large data base application, where 31 modules out of 425 contained more than 60% of all customer-reported bugs. These troublesome areas were known as "error-prone modules."

Similar studies by other corporations such as AT&T and ITT found that error-prone modules were endemic in the software domain. More than 90% of applications larger than 5,000 function points were found to contain error-prone modules in the 1980's and early 1990's. Summaries of the error-prone module data from a number of companies was published in the author's book *Software Quality: Analysis and Guidelines for Success* (3).

Fortunately it is possible to surgically remove error-prone modules once they are identified. It is also possible to prevent them from occurring. A combination of defect measurements, formal design inspections, formal code inspections, and formal testing and test-coverage analysis have proven to be effective in preventing error-prone modules from coming into existence (5).

Today in 2007, error-prone modules are almost nonexistent in organizations that are higher than level 3 on the capability maturity model (CMM) of the Software Engineering Institute. However they remain common and troublesome for level 1 organizations, and for organizations that lack sophisticated quality measurements and quality control.

If the author's clients are representative of the U.S. as a whole, more than 50% of U.S. companies still do not utilize the CMM at all. Of those who do use the CMM, less than 15% are at level 3 or higher. That implies that error-prone modules may exist in more than half of all large corporations and in a majority of state government software applications as well.

Once deployed, most software applications continue to grow at annual rates of between 5% and 10% of their original functionality. Some applications, such as Microsoft Windows, have increased in size by several hundred percent over a 10-year period.

The combination of continuous growth of new features coupled with continuous defect repairs tends to drive up the complexity levels of aging software applications. Structural complexity can be measured via metrics such as cyclomatic and essential complexity using a number of commercial tools. If complexity is measured on an annual basis and there is no deliberate attempt to keep complexity low, the rate of increase is between 1% and 3% per calendar year.

However, and this is an important fact, the rate at which entropy or complexity increases is directly proportional to the initial complexity of the application. For example if an application is released with an average cyclomatic complexity level of less than 10, it will tend to stay well structured for at least five years of normal maintenance and enhancement changes.

But if an application is released with an average cyclomatic complexity level of more than 20, its structure will degrade rapidly and its complexity levels might increase by more than 2% per year. The rate of entropy and complexity will even accelerate after a few years.

As it happens, both bad-fix injections and error-prone modules tend to correlate strongly (although not perfectly) with high levels of com-



1. Solve performance problems and bottlenecks.

2. Decrease testing time & improve our QA process.

3. Call Checkpoint Technologies!

Checkpoint Technologies, an HP Business Partner and Training Partner, is a recognized industry leader in software testing solutions, services, and education. We have a complete set of tools and techniques that will help you build a performance validation solution. For details: call toll-free +1 877.441.4448. Or e-mail: test_experts@checkpointtech.com.

See us at QUEST Chicago:
Tuesday, April 29—Making Test Automation Live Up to Its Promises—1-Day Tutorial
Wednesday, April 30—Test Automation in an Agile Environment



plexity. A majority of error-prone modules have cyclomatic complexity levels of 10 or higher. Bad-fix injection levels for modifying high-complexity applications are often higher than 20%.

In the late 1990's a special kind of geriatric issue occurred which involved making simultaneous changes to thousands of software applications. The first of these "mass update" geriatric issues was the deployment of the Euro, which required changes to currency conversion routines in thousands of applications. The Euro was followed almost immediately by the dreaded year 2000 or Y2K problem (6), which also involved mass updates of thousands of applications. More recently in March of 2007 another such issue occurred when the starting date of daylight savings time was changed.

Future mass updates will occur later in the century, when it may be necessary to add another digit to telephone numbers or area code. Yet another and very serious mass update will occur if it becomes necessary to add digits to social security numbers in the second half of the 21st century. There is also the potential problem of the Unix time clock expiration in 2038.

Metrics Problems With Small Maintenance Projects

There are several difficulties in exploring software maintenance costs with accuracy. One of these difficulties is the fact that maintenance tasks are often assigned to development personnel who inter-

Need to improve quality across the life cycle, reduce cost and risk, and improve application delivery outcomes?

Then you need Compuware quality and delivery management solutions.

For more information, visit www.compuware.com

COMPUWARE

475 Martingale Road, Suite 800
Schaumburg, IL 60173
(630) 285-8560

leave both development and maintenance as the need arises. This practice makes it difficult to distinguish maintenance costs from development costs because the programmers are often rather careless in recording how time is spent.

Another and very significant problem is that fact that a great deal of software maintenance consists of making very small changes to software applications. Quite a few bug repairs may involve fixing only a single line of code. Adding minor new features such as perhaps a new line-item on a screen may require less than 50 source code statements.

These small changes are below the effective lower limit for counting function point metrics. The function point metric includes weighting factors for complexity, and even if the complexity adjustments are set to the lowest possible point on the scale, it is still difficult to count function points below a level of perhaps 15 function points (7).

An experimental method called "micro function points" is in development for small maintenance changes and bug repairs. This method is similar to standard function points, but drops down to three decimal places of precision. Thus changes that involve only a fraction of a standard IFPUG function point can be measured. The micro function point method should become available by 2008.

Of course the work of making a small change measured with micro function points may be only an hour or less. But if as many as 10,000 such changes are made in a year, the cumulative costs are not trivial. Micro function points are intended to eliminate the problem that small maintenance updates have not been subject to formal economic analysis.

Quite a few maintenance tasks involve changes that are either a fraction of a function point, or may at most be less than 10 function points or about 1000 COBOL source code statements. Although normal counting of function points is not feasible for small updates and micro function points are still experimental, it is possible to use the "backfiring" method or converting counts of logical source code statements in to equivalent function points. For example, suppose an update requires adding 100 COBOL statements to an existing application. Since it usually takes about 105 COBOL statements in the procedure and data divisions to encode 1 function point, it can be stated that this small maintenance project is "about 1 function point in size."

If the project takes one work day consisting of six hours, then at least the results can be expressed using common metrics. In this case, the results would be roughly "6 staff hours per function point." If the reciprocal metric "function points per staff month" is used, and there are 20 working days in the month, then the results would be "20 function points per staff month."

Best and Worst Practices in Software Maintenance

Because maintenance of aging legacy software is very labor intensive it is quite important to explore the best and most cost effective methods available for dealing with the millions of applications that currently exist. The sets of best and worst practices are not symmetrical. For example the practice that has the most positive impact on maintenance productivity is the use of trained maintenance experts. However the factor that has the greatest negative impact is the presence of "error-prone modules" in the application that is being maintained.

Table 3 illustrates a number of factors which have been found to exert a beneficial positive impact on the work of updating aging applications and shows the percentage of improvement compared to average results:

Table 3: Impact of Key Adjustment Factors on Maintenance
(Sorted in order of maximum positive impact)

Maintenance Factors	Plus Range
Maintenance specialists	35%
High staff experience	34%
Table-driven variables and data	33%
Low complexity of base code	32%
Test coverage tools and analysis	30%
Code restructuring tools	29%
Reengineering tools	27%
High level programming languages	25%
Reverse engineering tools	23%
Complexity analysis tools	20%
Defect tracking tools	20%
"Mass update" specialists	20%
Automated change control tools	18%
Unpaid overtime	18%
Quality measurements	16%
Formal base code inspections	15%
Regression test libraries	15%
Excellent response time	12%
Annual training of > 10 days	12%
High management experience	12%
HELP desk automation	12%
No error prone modules	10%
On-line defect reporting	10%
Productivity measurements	8%
Excellent ease of use	7%
User satisfaction measurements	5%
High team morale	5%
Sum	503%

At the top of the list of maintenance "best practices" is the utilization of full-time, trained maintenance specialists rather than turning over maintenance tasks to untrained generalists. Trained maintenance specialists are found most often in two kinds of companies: 1) Large systems software producers such as IBM; 2) Large maintenance outsource vendors. The curricula for training maintenance personnel can include more than a dozen topics and the training periods range from two weeks to a maximum of about four weeks.

Since training of maintenance specialists is the top factor, table 4 shows a modern maintenance curriculum such as those found in large maintenance outsource companies.

The positive impact from utilizing maintenance specialists is one of the reasons why maintenance outsourcing has been growing so rapidly. The maintenance productivity rates of some of the better maintenance outsource companies is roughly twice that of their clients prior to the completion of the outsource agreement. Thus even if the outsource vendor costs are somewhat higher, there can still be useful economic gains.

Let us now consider some of the factors which exert a negative impact on the work of updating or modifying existing software applications. Note that the top-ranked factor which reduces maintenance productivity, the presence of error-prone modules, is very asymmetrical. The absence of error-prone modules does not speed up maintenance work, but their presence definitely slows down maintenance work.

Table 4: Sample Maintenance Curricula
for Companies Using Maintenance Specialists

Software Maintenance Courses	Days	Sequence
Error-Prone Module Removal	2.00	1
Complexity Analysis and Reduction	1.00	2
Reducing Bad Fix Injections	1.00	3
Defect Reporting and Analysis	0.50	4
Change Control	1.00	5
Configuration Control	1.00	6
Software Maintenance Workflows	1.00	7
Mass Updates to Multiple Applications	1.00	8
Maintenance of COTS Packages	1.00	9
Maintenance of ERP Applications	1.00	10
Regression Testing	2.00	11
Test Library Control	2.00	12
Test Case Conflicts and Errors	2.00	13
Dead Code Isolation	1.00	14
Function Points for Maintenance	0.50	15
Reverse Engineering	1.00	16
Reengineering	1.00	17
Refactoring	0.50	18
Maintenance of Reusable Code	1.00	19
Object-Oriented Maintenance	1.00	20
Maintenance of Agile and Extreme Code	1.00	21
TOTAL	23.50	

In general more than 80% of latent bugs found by users in software applications are reported against less than 20% of the modules. Once these modules are identified then they can be inspected, analyzed, and restructured to reduce their error content down to safe levels.

Table 4 summarizes the major factors that degrade software maintenance performance. Not only are error-prone modules troublesome, but many other factors can degrade performance too. For example, very complex “spaghetti code” is quite difficult to maintain safely. It is also troublesome to have maintenance tasks assigned to generalists rather than to trained maintenance specialists.

A very common situation which often degrades performance is lack of suitable maintenance tools, such as defect tracking software, change management software, test library software, and so forth. In general it is very easy to botch up maintenance and make it such a labor-intensive activity that few resources are left over for development work.

Table 5: Impact of Key Adjustment Factors on Maintenance
(Sorted in order of maximum negative impact)

Maintenance Factors	Minus Range
Error prone modules	-50%
Embedded variables and data	-45%
Staff inexperience	-40%
High complexity of base code	-30%
Lack of test coverage analysis	-28%
Manual change control methods	-27%
Low level programming languages	-25%
No defect tracking tools	-24%
No “mass update” specialists	-22%
Poor ease of use	-18%
No quality measurements	-18%
No maintenance specialists	-18%
Poor response time	-16%
Management inexperience	-15%
No base code inspections	-15%
No regression test libraries	-15%
No HELP desk automation	-15%
No on-line defect reporting	-12%
No annual training	-10%
No code restructuring tools	-10%
No reengineering tools	-10%
No reverse engineering tools	-10%
No complexity analysis tools	-10%
No productivity measurements-7%	-7%
Poor team morale	-6%
No user satisfaction measurements	-4%
No unpaid overtime	0%
Sum	-500%

The last factor, or lack of unpaid overtime, deserves a comment. Unpaid overtime is very common among software maintenance and development personnel. In some companies it amounts to about 15% of the total work time. Because it is unpaid it is usually unmeasured. That means side by side comparisons of productivity rates or costs between groups with unpaid overtime and groups without will favor the group with unpaid overtime because so much of their work is uncompensated and hence invisible. This is a benchmarking trap for the unwary. Because excessive overtime is psychologically harmful if continued over long periods, it is unfortunate that unpaid overtime tends to be ignored when benchmark studies are performed.

Given the enormous amount of effort that is now being applied to software maintenance, and which will be applied in the future, it is obvious that every corporation should attempt to adopt maintenance “best practices” and avoid maintenance “worst practices” as rapidly as possible.

Software Entropy and Total Cost of Ownership

The word “entropy” means the tendency of systems to destabilize and become more chaotic over time. Entropy is a term from physics and is not a software-related word. However entropy is true of all complex systems, including software. All known compound objects decay and become more complex with the passage of time unless effort is exerted to keep them repaired and updated. Software is no exception. The accumulation of small updates over time tends to gradually degrade the initial structure of applications and makes changes grow more difficult over time.

For software applications entropy has long been a fact of life. If applications are developed with marginal initial quality control they will probably be poorly structured and contain error-prone modules. This means that every year, the accumulation of defect repairs and maintenance updates will degrade the original structure and make each change slightly more difficult. Over time, the application will destabilize and “bad fixes” will increase in number and severity. Unless the application is restructured or fully refurbished, eventually it will become so complex that maintenance can only be performed by a few experts who are more or less locked into the application.

By contrast, leading applications that are well structured initially can delay the onset of entropy. Indeed, well-structured applications can achieve declining maintenance costs over time. This is because updates do not degrade the original structure, as happens in the case of “spaghetti bowl” applications where the structure is almost unintelligible when maintenance begins.

The total cost of ownership of a software application is the sum of six major expense elements: 1) the initial cost of building an application; 2) the cost of enhancing the application with new features over its lifetime; 3) the cost of repairing defects and bugs over the application’s lifetime; 4) The cost of customer support for fielding and responding to queries and customer-reported defects; 5) The cost of periodic restructuring or “refactoring” of aging applications to reduce entropy and thereby reduce bad-fix injection rates; 6) Removal of error-prone modules via surgical removal and redevelopment. This last expense element will only occur for legacy applications that contain error-prone modules.

Similar phenomena can be observed outside of software. If you buy an automobile that has a high frequency of repair as shown in Consumer Reports and you skimp on lubrication and routine maintenance, you will fairly soon face some major repair problems – usually well before 50,000 miles.

By contrast, if you buy an automobile with a low frequency of repair as shown in Consumer Reports and you are scrupulous in maintenance, you should be able to drive the car more than 100,000 miles without major repair problems.

Summary and Conclusions

In every industry maintenance tends to require more personnel than those building new products. For the software industry the number of personnel required to perform maintenance is unusually large and may soon top 70% of all technical software workers. The main reasons for the high maintenance efforts in the software industry are the intrinsic difficulties of working with aging software. Special factors such as "mass updates" that began with the roll-out of the Euro and the year 2000 problem are also geriatric issues.

Given the enormous efforts and costs devoted to software maintenance, every company should evaluate and consider best practices for maintenance, and should avoid worst practices if at all possible.

References On Software Maintenance

1. Jones, Capers; *Analyzing the Tools of Software Engineering*; Software Productivity Research Technical report, Burlington, MA; April 6, 1999.
2. Jones, Capers; *Estimating Software Costs*; McGraw Hill, 2nd edition, 1998. (Third edition due in April of 2007).
3. Jones, Capers; *Software Quality – Analysis and Guidelines for Success*; International Thomson Computer Press, Boston, MA; ISBN 1-85032-876-6; 1997; 492 pages.
4. Jones, Capers; *Program Quality and Programmer Productivity*; IBM Technical Report TR 02.764; IBM San Jose, CA; January 1977.
5. Jones, Capers; *Software Assessments, Benchmarks, and Best Practices*; Addison Wesley Longman, Boston, MA; ISBN 0-201-48542-7; 2000; 657 pages.
6. Jones, Capers; *The Year 2000 Software Problem - Quantifying the Costs and Assessing the Consequences*; Addison Wesley, Reading, MA; 1998; ISBN 0-201-30964-5; 303 pages.
7. Jones, Capers; *Applied Software Measurement*; McGraw Hill, 2nd edition 1996; ISBN 0-07-032826-9; 618 pages.

Readings On Software Maintenance

- Arnold, Robert S.; *Software Reengineering*; IEEE Computer Society Press, Los Alamitos, CA; 1993; ISBN 0-8186-3272-0; 600 pages.
- Arthur, Lowell Jay; *Software Evolution - The Software Maintenance Challenge*; John Wiley & Sons, New York; 1988; ISBN 0-471-62871-9; 254 pages.
- Gallagher, R.S.; *Effective Customer Support*; International Thomson Computer Press, Boston, MA; 1997; ISBN 1-85032-209-0; 480 pages.
- Grubb, Penny and Takang, Armstrong; *Software Maintenance - Concepts and Practice*; World Scientific Pub. Co; 2003; ISBN 981-238-425-1.
- Kan, Stephen H.; *Metrics and Models in Software Quality Engineering*; Addison Wesley, Reading, MA; ISBN 0-201-72915-6; 2003; 528 pages.
- McCabe, Thomas J.; "A Complexity Measure"; *IEEE Transactions on Software Engineering*; December 1976; pp. 308-320.
- Muller, Monika & Abram, Alain (editors); *Metrics in Software Evolution*; R. Oldenbourg Verlag GmbH, Munich; ISBN 3-486-23589-3; 1995.
- Parikh, Girish; *Handbook of Software Maintenance*; John Wiley & Sons, New York; 1986; ISBN 0-471-82813-0; 421 pages.
- Pigoski, Thomas M.; *Practical Software Maintenance - Best Practices for Managing Your Software Investment*; IEEE Computer Society Press, Los Alamitos, CA; 1997; ISBN 0-471-17001-1; 400 pages.
- Polo, Macario et al; *Advances in Software Maintenance, Management, Technologies, and Solutions*; World Scientific Pub. Co.; 1996; ISBN 981-022826-0.
- Sharon, David; *Managing Systems in Transition - A Pragmatic View of Reengineering Methods*; International Thomson Computer Press, Boston, MA; 1996; ISBN 1-85032-194-9; 300 pages.
- Takang, Armstrong and Grubb, Penny; *Software Maintenance Concepts and Practice*; International Thomson Computer Press, Boston, MA; 1997; ISBN 1-85032-192-2; 256 pages.

spherion
Professional Services

technisource

Creating one of the strongest
brands in North America

More Capabilities, More Flexibility,
Continued Service Excellence

DSR
MANAGEMENT, INC.

A unique blend of small IT services company
with access to large and global resources

Specializing in:

- QA Solutions (Assesment, Automation)
- eBusiness Development & Support
- Embedded Solutions
- IT Staff Augmentation

Ranked by INC Magazine as a Top 500 Fastest
Growing Company 3 years in a row

Check us out: www.DSRMINC.com
Or
Call 847-328-6355

DSR Management, 500 Davis St. Suite 801,
Evanston, IL 60201



QAI Chapter Network

The QAI Chapter Network is dedicated to supporting and promoting software quality, the quality profession, and the quality practitioner. The Chapter Network provides a framework for IT QA / QC professionals to network with individuals from other geographic regions and to increase their knowledge and expertise in these areas. Chapter meetings and programs provide an important opportunity for local professionals to meet and share ideas and resources.

There are currently QAI Chapters in the following areas:

Sacramento, CA	Portland, ME	Indianapolis, IN	United Kingdom
Los Angeles, CA	Cromwell, CT	Minneapolis/St. Paul, MN	Ensenada, Mexico
San Dimas, CA	Wilmington, DE	Kansas City, MO	Monterrey, Mexico
West Hills, CA	Boston, MA	St. Louis, MO	Rio de Janeiro, Brazil
Westminster, CA	Baltimore, MD	Omaha, NE	Argentina
Salt Lake City, UT	Jacksonville, FL	Columbus, OH	South Africa
Phoenix, AZ	Lauderhill, FL	Solon, OH	Pakistan
Seattle, WA	Orlando, FL	Madison, WI	India
Houston, TX	Tampa/St. Petersburg, FL	Detroit, MI	Sri Lanka
Denver, CO	Atlanta, GA	Holland, MI	North China
Boise, ID	Charlotte, NC	Montreal, Quebec	Shanghai
Dulles, VA	Little Rock, AR	Ottawa, Ontario	Hong Kong
Glen Allen, VA	Research Triangle Park, NC	Toronto, Ontario	Korea
New York, NY	Chicago, IL	London, Ontario	Penang, Malaysia
New Jersey, NJ	Des Moines, IA		

For additional information on the QAI Chapter Network, or to learn how to start a QAI Chapter in your area, visit:

www.qaiworldwide.org/chapters/index.html

QAI's Software Certifications Examinations Available at Prometric Testing Centers

Since its inception in 1988, the Software Certifications division of the Quality Assurance Institute has scheduled quarterly examinations at locations around the world. QAI's global chapter network of software quality professionals has been the primary vehicle for proctoring examinations at regional locations. This approach has served the industry well over the last two decades; however, as a result of the Software Certifications' CSQA and CSTE becoming the recognized international standard for software quality and software testing professional certification, the need for expanding the frequency and locations of the examination has grown exponentially. To satisfy this increasing demand, the Software Certifications division of the Quality Assurance Institute has selected Prometric, the global leader in technology-enabled testing services, to deliver the examinations worldwide*.

"Going with the Prometric Testing Center model will ensure program enhancements and more consistency in the customer experience," said Tom Ticknor, Chief Operating Officer for QAI Worldwide. "It will also enable increased speed to market for exam innovations because our internal processes will be more streamlined. In a time when demand is growing for IT professional certifications, it is very important to us to deliver the best possible certification program and consistent testing experience for our customers."

Michael Brannick, President and CEO of Prometric went on to say, "We have recently seen a strong resurgence of the IT certification market in general and in candidate demand for quality, dependable testing programs. Prometric is widely recognized and established across the IT industry as the leading provider of certification exams, with some of the world's largest technology companies relying on us for their testing needs. The Quality Assurance Institute and Prometric working together is an excellent solution to this growing global need."

Certifications available through QAI's Software Certification division are: Certified Software Quality Analyst (CSQA), Certified Software Tester (CSTE), Certified Software Project Manager (CSPM), Certified Manager of Software Quality (CMSQ), Certified Manager of Software Testing (CMST), Certified Associate in Software Quality (CASQ), Certified Associate in Software Testing (CAST), and Certified Software Business Analyst (CSBA). For more information on the Software Certifications program please visit www.softwarecertifications.org.

* Locations in India, China and Brazil will not be available at Prometric Testing Centers until 3rd quarter 2008. Paper-based quarterly exams will still be given in these sponsored locations. Check the Software Certifications website for the sponsored examination schedule and locations.



A Commentary on the Book Rising Elephant

BY DENISE COPENING
CSQA, CSTE, PMP

This article is a follow up to “Edward Yourdon’s *Outsource: Competing in the Global Productivity Race in Perspective*”. It is in response to the question regarding the extent of outsourcing.

Mr. Yourdon posed that one of the most critical factors in determining the future is the magnitude of outsourcing. If one would take in the entire gestalt of the facts presented in Mr. Sheshabalaya’s book, the conclusions may be the impact of outsourcing is indeed great, beneficial to the world economy overall, and India is well on its way to becoming a major economic power. Mr. Yourdon’s book addresses facts and issues regarding IT outsourcing to India, however, it would not fair to compare it with Mr. Sheshabalaya’s book since these books were written for different purposes. The main thrust of *Rising Elephant* is to detail the current economic state of India, with an emphasis on outsourcing. Mr. Yourdon’s book is centered on what IT people in the US can do to maintain and enhance their careers. Mr. Sheshabalaya makes policy recommendations in his book, yet Mr. Yourdon stated directly that he was avoiding these. There is intersection on some points between these books; however, this author would rather refrain from drawing comparisons.

Rising Elephant by Ashutosh Sheshabalaya could easily be alternatively titled *Near-term Economic Outlook for India*. It is a detailed, copiously researched treatise on the current state of India in the world economy, including an educated look into projecting the future. One chapter has in excess of 280 endnotes. The author of this book previously wrote a 250-page research study on Indian information technology in 1997. His past projections for economic expansion have proven to be fairly accurate.ⁱ

This book is truly approached from a world view. The analyses include Western Europe, Australia, and the United States. Most similar material read by the author of this article concentrates on the United States. The focus is on economics and politics, including the white collar job migration to India. It contains an exhaustive compendium of detailed statistics which are overwhelming – both in quantity and the story they tell. Arguing with Mr. Sheshabalaya on any point supported by quantitative evidence would be very difficult since he provides more than a sufficient amount of credible references to back his position. He appears to be one of the definitive researchers regarding the Indian economy based on this work and his previous one. Mr. Sheshabalaya painstakingly debunks the “myths” he has identified about India and outsourcing in this book.

In the US media for the past few years, there have been many articles and news bytes about white collar outsourcing. The lesson this author has absorbed from this book is that it is important to discern actual facts from propaganda. The author has seen the attitude over

the last year or two shift to one of skepticism and fear in the US portrayed in articles in popular information technology publications. In conversations with US nationals, some present a factoid (such as Dell technical support moving back on shore) as proof for denigrating outsourcing in general and possibly even trying to convince oneself and others that it is a temporary phenomenon.

It is difficult to ignore that the boom provided to India’s economy regarding white collar outsourcing is part of a normal evolutionary process within globalization. The migration of these jobs started in the early 1980’s, although the popular realization of this dynamic occurred in the past few years as waves of outsourcing were manifested by major companies such as General Electric and IBM..

India has been laying the groundwork and infrastructure to assume high end white collar work for quite some time. It is not just call center work which is being outsourced. Indians and Indian Americans have played a significant role in the development of IT for several decades. Mr. Sheshabalaya repeatedly makes the point about CMM and the high percentages of IT companies located in India possessing this certification. He provides examples where certain major US-based companies have specifically set up IT development organizations focusing on CMM Level 5. He further adds that Indian firms are currently working on removing roadblocks to outsourcing such as lack of security. He provides an example where Indian firms are currently working with Carnegie Mellon regarding a security certification.ⁱⁱ Also, he mentions Six Sigma initiatives of larger Indian IT firms involving Business Process Outsourcing (BPO).ⁱⁱⁱ

The CMM ratings of some large outsourcers are not without critics. In a CIO Magazine article published in 2004, CIO’s are urged to perform due diligence when selecting organizations based on CMM ratings. The article states that the results of these assessments are often exaggerated. The actual case may be that only a small part of the company could be considered CMM Level 5.^{iv}

One of the new trends Mr. Sheshabalaya has identified is the outsourcing of higher level jobs to India. A chart in the book depicts projections that managerial jobs will constitute about 8% of all US jobs outsourced by 2010.^v Outsourcing of higher level work is not without challenge. In reaction to this trend, the author of a Network Computing article prognosticates that a portion of these jobs will be reclaimed domestically after the outsourcing experiment with these fails.^{vi}

There has been criticism that India does not have the infrastructure to support outsourcing. A relative of this article’s author regularly trades foreign currencies from New York and has experienced telecommunication disruptions when calling India.

Conversely, a blog a few days after the Tsunami disaster by a Business 2.0 writer proclaims minor disruption in India due to a fiber optic network which carries most of the data for US outsourced applications.^{vii} According to Mr. Sheshabalaya, projects to provide electricity to outreaching areas and build roads and highways are currently underway in India.^{viii}

India is poised for success in the world marketplace with its large number of English speaking people, good education system, and young demographic profile where 68% of its 1.05 billion people are under 35 years old. The size of India's middle class equals that of the population of the entire United States.^{ix} India is potentially facing a period of high growth similar to what was experienced in the 1950's in the US. Savers are becoming spenders and there is a housing boom underway.^x In contrast, the US and Western Europe, although there is excellent education available, have aging populations and fairly stagnant economies.

High costs of pensions, healthcare, and medical insurance in the US and Western Europe are making doing business in India more attractive. Escalation of healthcare and pension costs add to overall bill for employing US workers compared to outsourcing. Healthcare costs especially have been of concern in the US in recent years. Double digit percentage increases in medical insurance premiums in the past few years have become all too common. High medical costs for illnesses led to about 50% of all personal bankruptcies in the US. Most people in this situation who filed for bankruptcy protection actually had health insurance.^{xi} A recent Harvard study probes this topic in detail.^{xii} Americans with seemingly good insurance may face serious medical situations where they need to dip into their life savings significantly.^{xiii}

Moreover, the US is in a unique position relative to most countries where there is a dependency on employers to provide and fund healthcare insurance. Due to the rising costs, dependency on employment, and aging population, has the American worker, in effect, been priced out of the market? What portion of the so-called "jobless recovery" can be attributable to outsourcing of US jobs?

The relative value of wages is different in India than the US. Large Indian outsourcers have had problems retaining key people, in part, due to this. They have experienced a "brain drain" of some of these key individuals to the US.^{xiv}

Recently, the tide seems to be turning in the US regarding sentiments about off shoring.^{xv} In February 2004, the US Workers Protection Act was proposed seeking to ban offshore sourcing in government work. The "Jobs for America Act" would require corporations sending jobs overseas to report the number and reason. If 15 or more workers are being laid off and their jobs outsourced, this would require at least three months' notice to the displaced workers.^{xvi}

The above are only temporary measures. Educational gaps in this country have been behind the need to outsource and grant H1B visas. The long-term solution is to improve education within the US.^{xvii}

As Mr. Sheshabalaya emphasizes in his book, cultural understanding could improve our ability to work together and perceptions of what is happening in the world economy.^{xviii}

A likely parallel we in the US may comprehend is outsiders being called "the English" by a certain religious sect. We may want to consider expanding our views a bit, but this is all within the realm of an individual's choice and their level of comfort.

The point is made regarding. "What is an American corporation?" The book's author notes many Indian firms providing outsourcing services are indeed based in the US, thereby having a dependency and tie-in with the US economy.

A summary of Mr. Sheshabalaya's policy suggestions follow:^{xix}

- IT workers should train for emerging skill niches such as business process design, requirements analysis, contract management, business relationship management, and architectural planning. Sensitive sectors will remain to have domestic openings such as government and military technology, health care and pharmaceuticals. The article's author would like to add securities trading to this list.
- Reverse relocation to India could be considered. It is estimated that there are approximately 200,000 expatriates currently working in India. If one were to seriously consider the facts and conclusions proposed by Mr. Sheshabalaya in this book, then one might catch the beginning of a trend and reap the benefits of plentiful job and business opportunities and low cost of living.
- Governments could assist relocated workers. In the US, the Trade Adjustment Assistance Program does not include programmers and other information technology workers. There was a class action lawsuit on behalf of about 10,000 claimants in 2004 with the Labor Department cited as the defendant for illegally denying these benefits to IT workers.
- A tax could be excised on companies involved in outsourcing.
- The offshore locations could contribute to job insurance to benefit the displaced workers.
- Require technology companies to publish five-year staffing plans. It is interesting to note that before the Human Resource cutbacks over the past few decades in US companies that some companies had undertaken similar planning.
- Cross train workers in India.
- Provide tax credits and other types of support for Indian firms to hire non-Indians.

One could critique Mr. Sheshabalaya's policy recommendations as having a socialistic underpinning. Why not permit market forces to operate freely and let corporations make money which will trickle down and be of benefit to all? For example, most US companies are experiencing slow domestic growth. The size and age of the Indian market, let alone the predicted economic boom can potentially provide US companies with profits, which in theory, would benefit the US overall.

It is the opinion of the article's author that Mr. Sheshabalaya's policy recommendations are rooted in an emancipated view of the world economy and societies as a large ecosystem requiring strategy and a quid pro quo for sustenance in the long-term. The contrast between the US and India is made regarding how the long-term is viewed.

India's civilization is an ancient one versus the US being a relatively new nation with hundreds of years of history.^{xx} In the US, it seems like Americans historically have been off to the next "big thing" with little emphasis on continuity or long-term planning.

The author of this article has developed a few insights of her own regarding policy and what individuals may consider doing:

- In recent years, US companies have cutback training dollars for IT workers. Those seeking additional training have needed to self-fund it or forego it altogether. US educational institutions could offer discounts or subsidies provided by the government and/or private industry. Some states such as Illinois do offer training subsidies for the unemployed.
- Displaced IT workers seeking opportunities outside of IT need to choose careers based on an affinity for the work and entrance criteria such as education required, market saturation, and networking required, etc.

- Those deciding to open their own businesses need to carefully consider the economics of this decision and weigh the potential benefits and detriments of the opportunity. What this author has observed is that many Americans entering small business seem to jump on the same type of opportunities at the same time, resulting in marketplace saturation (remember frozen yogurt stands?). There is also a tendency to enter business that one may not enjoy doing the work – such as cleaning services. In order to glean large profits, one may need to stray from the normal path and also avoid businesses which are labor intensive (that is requiring staffs) and capital intensive.
- This article's author has known quite a few small business owners who have not achieved the profit goals they expected within a given timeframe and have found it necessary to return to work on either a part-time or full-time basis while maintaining the small business. Some people considering opening a small business may want to start the business while they are still employed to test its viability.
- Franchise opportunities should be weighed with at least the same consideration as changing careers along with the analysis required for a major investment decision. For most franchises, one must have a significant amount of capital and time to devote to learning the franchise ways of doing business. Franchising fees and royalties constitute a major portion of revenues. Franchise opportunities may require relocation since existing franchise owners may have first dibs on a new location. Franchises may not offer the autonomy one would want in owning their own business; operating procedures can be very rigid depending on the franchise.
- Local colleges and universities frequently will offer courses for those considering small business. It may be a good idea to take one of these courses before venturing out on one's own.
- An Internet business in and of itself is no guarantee of steady income. People need to consider Internet businesses offering some uniqueness and draw for customers for a good or service as opposed to acting solely as the middleman for these.
- Displaced IT workers could consider obtaining government grants for certain types of work or study or a loan from the Small Business Administration to assist in starting a business. Minority loans would also be available to those who qualify. Perhaps a policy decision could be made providing favored status for small business loans for displaced IT workers.

The social and economic changes due to outsourcing over the last decades have been sweeping, with a profound effect on Information Technology. Outsourcing has become an essential part of IT. Many US nationals in IT have had to re-think their careers in the "jobless recovery". Policy changes will be long-term such as educational reform and corporate tax incentives and are not something we can wait on to help us now. In a sense, the future has already arrived and it is our own resilience to the challenges posed which will see us through.

Denise Copenig Biography

Denise Copenig has Project Management Professional, Certified Software Quality Analyst, and Certified Software Testing Engineer certifications. She has an MBA degree from the University of Colorado and is a member of Beta Gamma Sigma. With twenty-two years' experience in IT and twelve years of Quality Assurance experience, she has facilitated JAD sessions, focus groups, designed and conducted educational programs, and developed Quality Assurance strategies, methodologies, and business and QA processes. She has held systems and business analysis and project leadership positions in the automotive, insurance, industrial and office product distribution, marketing intelligence and financial services and trading industries. Denise has previously been a speaker at a QAI Conferences and the PSQT Conference. She currently is an Associate with Olenick & Associates in Chicago. She can be contacted at dcopenig@olenick.com

Footnotes

- i Sheshabalaya, Ashutosh, *Rising Elephant: The Growing Clash Over White Collar Jobs and Its Challenge to America and the World*. Monroe, ME: Common Courage Press, 2005, comment in book jacket.
- ii Sheshabalaya, p. 190.
- iii Sheshabalaya, p. 61.
- iv Koch, Christopher, "Bursting the CMM Hype," *CIO Magazine*, 3/1/2004, p. 2.
- v Sheshabalaya, p. 8.
- vi Kaihla, Paul, "Offshoring Has Its Limits," *Network Computing*, May 18, 2004.
- vii Malik, OM, Blog at the following website address: www.gigaom.com/2004/12/fiber_survives_t.php
- viii Sheshabalaya, p. 60.
- ix Malik, Om, "The New Land of Opportunity," *Business 2.0*, July 2004, p. 74.
- x Malik, Om, "The New Land of Opportunity," *Business 2.0*, July 2004, p. 75.
- xi Gordon, Marcy, "Erasing Debts With Bankruptcy Gets Harder," *Chicago Tribune*, March 8, 2005.
- xii Miller, Rubin, "Medical bills pave way to poorhouse, study says - Many bankruptcies linked to illness," *Chicago Tribune*, February 2, 2005.
- xiii Kleiman, Kelly, "Sometimes It Does Not Pay to See a Doctor When You're Sick," *Chicago Tribune*, January 2, 2005, pp. 24-26.
Ms. Kleiman discusses her plight being non-insurable in this article. She tells the story of Henry Brandt, who had insurance coverage, however multiple heart surgeries necessitated his dipping into his life savings of \$150,000 to cover medical and living expenses.
- xiv La Monica, Paul R., "The Real Risk in Indian Offshoring," *Business 2.0*, July 14, 2004.
- xv Koch, Christopher, "The Ultimate Cost of Offshore Outsourcing," *CIO Magazine*, September 1, 2003.
- xvi Sheshabalaya, p.30.
- xvii Preston, Rob, "Offshore Outsourcing Prevention Starts at Home - A Foreign Concept," *Network Computing*, December 16, 2004.
- xviii Sheshabalaya, p. 202.
- xix Sheshabalaya, p. 267-274.
- xx Sheshabalaya, p. 6.

Testers

Gear Up and Get Better

Meet other testers, share ideas and get answers at
Microsoft Tester Center.

Visit, click and participate at
<http://msdn.com/testercenter>

