Egypt–SPIN Newsletter

Issue 7, July. – Sept., 2004

Sponsored by SECC

From the Editor (**Ahmed S. El-Shikh**)

Welcome to our 7th issue of Egypt –SPIN newsletter. In each issue we are trying to put together relevant information in the form of articles and recaps from the previous 3 months events hoping to provide our members of Egypt – SPIN with information to support their current interests.

> Last issue we had announced that **ITsoft** had attained SW-CMM Level 3. This issue with a great pleasure, we announce the great achievement, which had been done by other four of the Egyptian companies, **DMS**, **Cairo Technology Development Center - IBM Egypt**, **EDS-Egypt** and **Raya Software**. DMS and IBM Egypt are now SW-CMM level-3, EDS-Egypt and Raya-Software are CMMI level-3. Congratulations to every one of the 5 companies.

We can consider this issue as a special one. In this issue we record and share the experience from one of the four successful companies (1st article), and wait for the contribution from other ones in the coming issues. Also, there are many new contributors in this issue. They are writing about the software industry in Egypt (2nd and 3rd articles), and quality of software processes and projects (i.e. sizing, estimation and risks) (5th and 6th articles).

Eng. Sherin M. Murad is sharing with the community the **DMS's Software Engineering Process Improvement Journey.** The article explains how DMS managed the quality improvement strategy, process management approach and quality function.

Dr Meer Hamza is writing about the **Software Industry** in Egypt. What is its relation with the education system, educational system pitfalls, company point of view of quality and the starting point to change the mind about quality.

Eng. Amr Shaltoot is suggesting an **Idea to Plan for the Software Industry Future** in Egypt. How can we change the software craft into a matured industry? Where we are and where are we heading to be in the future?

Eng. Omar Kamal is contributing in this issue with an article that summarizes his experience in the **Software Unit Testing**. He is discussing the importance of unit testing, its relation with regression testing. Skills required for testing team, implementation, pitfalls and other aspects.

Eng. Sally Mohamed is explaining the **Estimation Challenges** in the software projects. Factors affect the estimation accuracy, estimation model, the importance of the historical base line of performance and associated risks.

Eng. Yasser Taymour is exploring the **Risky Nature of the Software Projects Environment**. What are the project risks? Risk management process, identification, analysis, quantifying and risk management planning.

Eng. Ahmed El Shikh is introducing the concept of the **Statistical Process Control**. How can it be used to management of the software process and build performance baseline, its relation with software measurement, CMMI level-4 goals , practices and problem solving techniques.

We hope we succeed to give you an idea about what is going in our community. Please write to the editor your comments about our progress. We always ask you to submit short articles for publication that deal with your experience in defining, developing and managing software efforts as well as process improvement experience. Remember that our goal is to encourage an interchange between our readers. You can email spin@secc.org.eg or el_shikh@sas-sys.com

Table of Contents

DMS Quality Movement & Software Engineering Process Improvement Journey

By: Sherin M. Murad

In recent years, concern for quality has become a national and international movement. Today, quality is a key factor in international competition. This article discusses how the quality movement and TQM apply to software engineering in DMS. The article summarizes basic quality improvement strategies and identifies milestones in quality technology. Most of the discussion deals with the process management approach to quality improvement and the software quality technology required to apply that approach to software engineering.

A. Quality Technology Milestones

The milestones in the development of quality technology are:

1. Product inspection
2. Process control which requires understanding the capability of a process and setting control limits [Ishikawa85] for key process variables.

B. Quality Improvement Strategy

DMS believes in process-centered approach to do quality improvement. Process management provides a systematic approach to controlling and improving quality. Studying the process and analyzing its performance does help to develop improvement strategies.

DMS implements four steps based on Juran recommendation [Juran81]:

1. Study the symptoms of poor quality (defects and failures).
2. Develop a theory or explanation of the cause of poor quality.
3. Test the theory in production to establish the cause.
4. Implement corrective or improvement action.

Two types of causes of poor quality; *Special causes* show up as failures in the established process; the process is not working as intended and needs to be tuned or repaired because performance is less than expected. *Common causes* are basic weaknesses in the underlying process. Properly trained workers can identify and correct special causes. Only management has the authority to make the basic changes necessary to deal with common causes. Both types of problems can be addressed with the same approach: the Shewart plan-do-check-act cycle shown in Figure-1. The cycle shows obvious similarity to Juran's four steps.

C. Process Management Approach

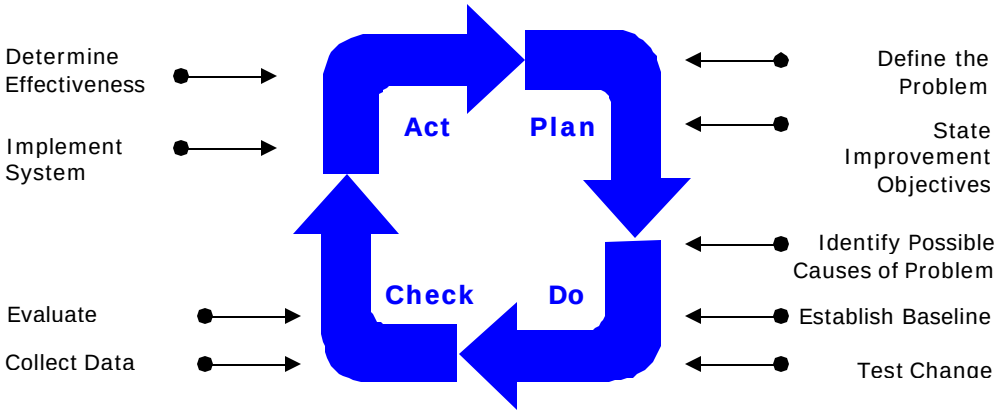
Our process management approach includes three essential activities:

1. Process definition
2. Process control
3. Process improvement

An undefined process cannot be controlled. An uncontrolled process cannot be improved consistently. Because improvement means change, attempting to improve an unstable process often leads to further instability. Figure 2 shows how these elements are connected by the

common threads of performance data and corrective action.

The process definition activity provides a prescription for performing work. Measuring the initial process performance establishes a baseline against which subsequent performance can be compared. The process control activity is concerned with identifying and correcting *special causes* of poor quality, to keep the process performing as intended. That is, it seeks to maintain key quality parameters within pre-defined control limits. The process improvement activity seeks to identify and rectify *common causes* of poor quality by making basic changes in the underlying process.



- Identify opportunities for improvement
- Develop plan for improvement
- Take corrective action on common causes
- Pursue continuous improvement

Figure 1: The Shewart Plan-Do-Check-Act Cycle

The three activities of process management can be mapped to the SEI process maturity levels. The process definition step coincides with the *Defined level*, the process control step corresponds to the *Managed level*, and the process improvement step matches the *Optimizing level*.

One can get the impression that the process management approach appears to ignore the software product. In fact, product quality is an integral part of process management; the software engineering process is defined around work products—they are the interfaces between activities. Moreover, process performance is monitored in part by looking at products. Examining the software quality functions in a mature process will clarify this connection.

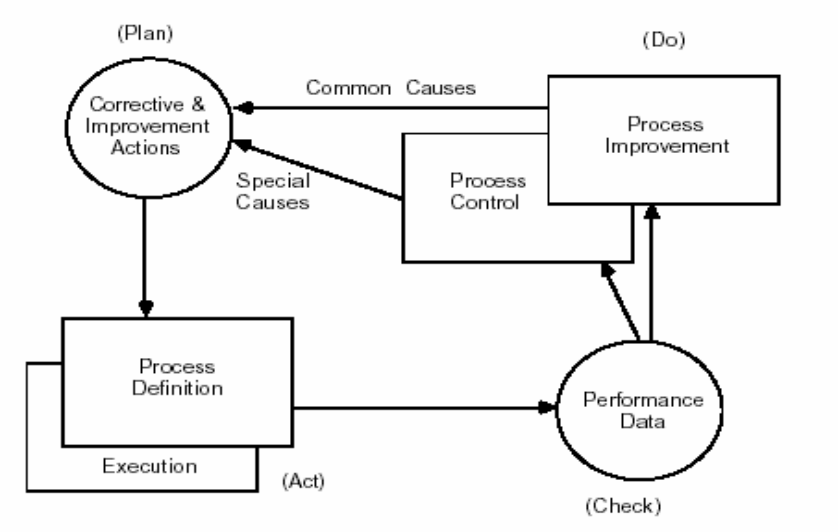


Figure 2: Process Management Approach

C. Software Quality Functions

Combining the historical view of quality technology milestones with the process management approach to quality improvement suggests that a fully mature software engineering process includes seven quality functions:

1. Process definition
2. Product inspections
3. Process audits
4. Software quality control
5. Process quality control
6. Software design improvement
7. Process design improvement

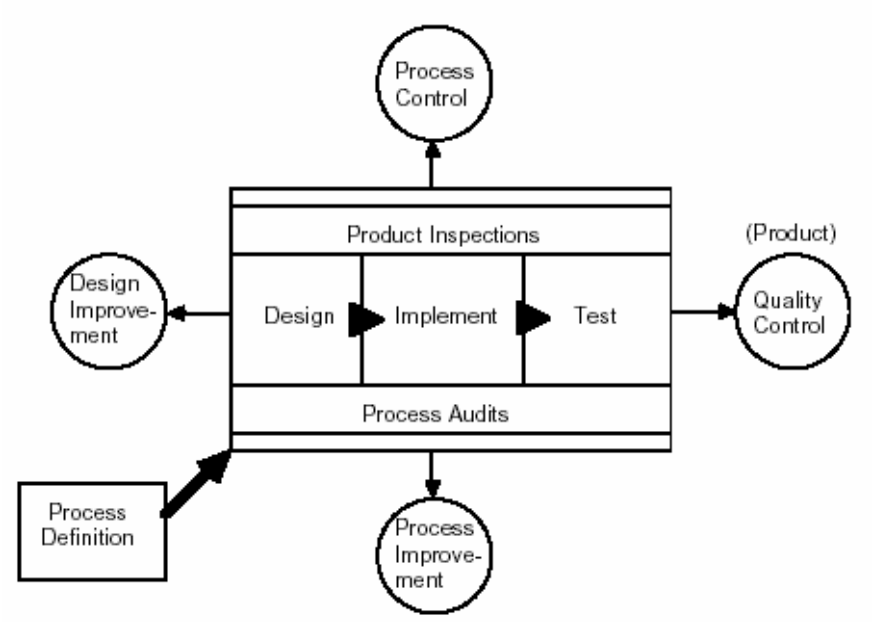


Figure 3: Software Quality Functions

D.1. Process Definition

Because the software engineering process is intellectual rather than physical, explicitly, defining that process—the steps, sequence, requirements, team composition, interfaces, etc.—can be challenging. Process definition leads to process control and improvement.

D.2. Product Inspections

The simplest quality technology involves performing peer reviews and inspections of both work in progress and final software products and documents. An inspection is a structured technique for comparing a software work product against its specification and other quality criteria [Fagan76]. Inspections result in corrections to software work products. Effective inspections require planning and training. Quantifying the results of inspections establishes a basis for process control and improvement.

D.3. Process Audits

An audit of the software engineering process focuses on the adequacy of the methods, tools, standards, and procedures in place in a project [Crawford85]. It also considers the conformance of the project to the prescribed defined process. The actual processes may, after all, depart significantly from the intended process. To be effective, process audits should suggest corrections to the software engineering process. Process audits should be oriented to improvement, rather than adversarial and oriented to problems. Poor quality and productivity may signal the need for an audit. An evaluation or assessment using the maturity model [TR23] can be considered a process audit.

D.4. Software Quality Control

Software quality control involves measuring the quality of a software work product and determining the need for corrective action. For example, if reliability is less than required, then rework and further testing must be performed [Currit86]. Often the customer will specify acceptance criteria for software products. Alternatively, the software enterprise may have internal criteria. These may be stated in terms of complexity, module size, reliability, etc.

D.5. Process Quality Control

Once established, a software engineering process can be expected to perform at a relatively constant level until some unanticipated problem or change occurs (a *special cause* of poor quality). Tracking actual versus expected performance on a control chart can highlight process problems [Gardiner87]. Once the software enterprise recognizes that performance has departed substantially from expectations, the search for a cause can begin. Maintaining process control means continuously correcting process problems.

D.6. Software Design Improvement

Until recently, hardware designers left many important product quality considerations to be handled by manufacturing engineers. Because software does not go through a corresponding manufacturing phase, the software engineer must deal with those producibility concerns directly [Card90]. That is, the software system must be designed to be easy to implement and maintain. This is in

addition to satisfying the customer's functional requirements.

D.7. Process Design Improvement

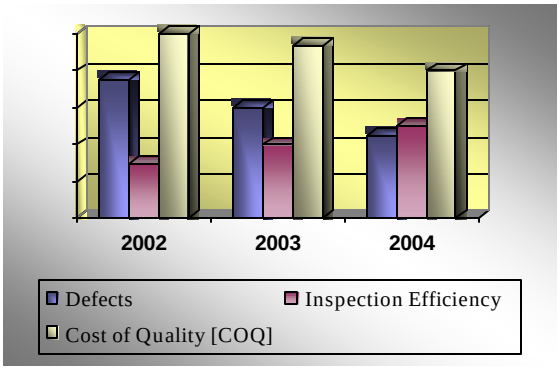
Once the process has been defined and controlled, management can turn its attention to improving the underlying process. This can be done by simplifying the process and by inserting appropriate new technology. [Turner78] suggests five questions that should be asked about each process element:

1. Is this activity necessary or can it be eliminated?
2. Can this activity be combined with another or others?
3. Is this the proper sequence of activities?
4. Can this activity be improved?
5. Is the proper person doing this activity?

Initial process improvement efforts should be concentrated at *leverage points*: those activities that require the most effort and produce the most problems. Improvement actions should be evaluated in situ by studying their effect on actual process performance.

E. DMS Achievements

1. Product's Defect Decreasing.
2. Inspection Efficiency Increasing.
3. Cost of Quality Decreasing.



4. Certified SW-CMM Level 3 using SCAPMI Method.

E.1 Why SCAMPI Method is chosen to be implemented for DMS Appraisal?

The decision was taken based on the following contrasting comparison between CBA IPI and SCAMPI features

Appraisal Method Feature	CBA IPI	SCAMPI
Fundamental approach / Data collection mechanism	Observation-driven (statements)	Practice Implementation Indicator Driven
Assessment Team Emphasis	Discovery	Verification
Finding sat aments generated	All strengths and weaknesses	Primarily for weaknesses; avoid crafting of "gratuitous" observations for strengths
Corroboration	2 different sources for valid observations, 2 different data gathering sessions, at least one data point reflecting actual work	Cooroboration built into the appraisal data types (direct, indirect, affirmation)

And, as a preparation step for transition to CMMI; by the end of the appraisal we have got a GAP Analysis Report and this is valid by using SCAMPI Method documents.

F. Conclusion

The quality movement is an international force. Its influence has begun to affect the way software is

developed and maintained. Many issues regarding the application of quality improvement principles to software engineering can be resolved only through experience. The SEI maturity model of the software engineering process, assessment procedure, and process group concept are tools that can help to install TQM in software organizations, where it can continue to evolve.

Biography

Sherin M. Murad is the Software Process Improvement Team Leader in DMS since August 2003. Sherin worked as software quality engineer (2001-2003), and played a various roles on Managerial and Technical Levels. Sherin has a Master Degree in Computer Science from Louisville University, Kentucky, USA.

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Author contacts:

E-Mail: shmurad@dmsegypt.net

How to Achieve the Expected Quality in our Software Industry?

By: Dr Meer Hamza.

Nowadays software industry has become an important objective for most countries, it is clear that it is now constituting an important portion of the GDP of a number of countries. It is also one of the signs of the development within countries.

The competition between companies in this field started 30 years ago. Competition was mainly on prices, as quality was nearly the same in all companies. Some countries enjoyed the privilege of having cheap labor cost, which allowed it to compete strongly on price. But now the case is completely different as all users wouldn't accept except a high level of quality dependent on the area of specialization.

In order to achieve the desired quality in the software industry a lot of researches have been done and standards and models in quality was developed and used. CMM as one of the pioneer models became an objective of the companies working in this area. After competing on price, we find India showing with more than 50% of the companies who are certified CMM level 4 and above.

The Egyptian Government has made a lot of efforts regarding that aspect, through establishing special agencies that help companies to achieve better quality. However, we think that quality is more a concept and a way of life than just models to be implemented. And in order to pass this concept to our people we have to start at a very early stage, they must learn in schools how to opt for and achieve quality. Pupils in schools and university students, especially those specialized in computing, must be taught that quality is something we try to achieve throughout our life. They must study to achieve the quality of learning and knowledge not just getting well prepared to exams through studying hard and taking private tuition, which leads no way but to passing exams and acquiring no knowledge at all. College students specialized in computing must have a big portion of software quality understanding through having subjects such as quality and total quality management.

Another pitfall in our educational system is the limited number of specialized colleges in the field of computer science and information technology as these colleges targets the aim of developing students specialized in computer systems with the thought of software engineering taking the basics and advanced levels in managing software projects, estimating the cost, defining and using software metrics to assess the progress and most importantly the quality from all its points of view including maintainability, reliability, readability, usability and all other aspects that makes up a good quality software system.

Our educational syllabuses even of computer colleges keep stressing on teaching language so and so and this technology and that technology but this do not make students that go for quality but those who want to finish the system as rapid as it could be regardless of what quality level is it and by this we will not make a progress, by this we need a severe change in the courses taught and its

contents to emphasis the measures of software and its quality.

The need to implement the quality concept is not confines only to the software industry; it must be a way life for our country. All religions urge attaining high quality standards in work.

If we managed to reach this degree of good quality work in all aspects of life, then we might care about implementing any model or knowing the software metrics and ways of measuring and implementing it in different industry fields that need software.

Biographies:

Dr. Meer Hamza is the Dean of College of Computing & IT in the Arab Academy for Science and Technology. His main areas of research include Software Engineering, e-Commerce, Databases and AI.

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Author contacts:

E-Mail: meerh@aast.edu

Understand the Word Export!

By: Amr Shaltoot

Good day to every one! Among the past several months, I was engaged in several occasions, regarding what is so called ICT Industry. A plenty of the conversations held by the esteemed attendees of such forums, committees, conferences, workshops, and other types of gatherings, asked me to share my thoughts about the future of " Software Industry "! I was shocked by the type of question. For a certain while I thought it might be safer to say a generic reply, and leave the subject for another time. Unfortunately I found my self answering in a detailed way.

The generosity of the attendees encouraged me to bring my solid abstracts about the whole vague term "Software Industry ", up to the surface.

However, where do we all stand, in the world of "Open-Markets" of the ICT. I think we even do not stand by the farthest edge of this huge profitable market.

Why? Some may start accusing me by: a stereotyped mind! Kindly, just give me the opportunity and let me explain more.

When the term "Industry" comes to adhere with any other type of understandable words like "Software", this will take our imagination to a certain type of pre-identified group of thoughts and meanings about the first word "industry". Many of us use the word "Industry" while some of its major components are not crystal-clear concepts in their minds. However, if we are discussing a "Live" "Industry" because many are now "Dead" ones, then "Software Industry" is one of them.

How do any craft become an Industry? When a systematic way of productions are established using production-lines! When a large amount of people sharing the same skills work in patterns to achieve a final product or products! When several feeding-networks established around this Industry! Or when universities start to make curriculum, to teach the new welling-to-join-the-Industry people to understand "Trial and error" made by the early patriots? I think none of the above turns the answer becomes complete!

Markets, is the answer to the question mentioned above. When you successfully, make a product (service or whatever) because it is demandable by a specific amount of capable-to-afford people, or whether you will make them demanding. Now, you may are paving the way to establish an Industry!

Do we have any of the above components? Unfortunately we have the most important part of the system or "the Industry", we have the capable-to-afford people. Most of the Egyptian software companies -earlier, owners liked to call it a house rather than a company, and nowadays, they would prefer to call it a factory are nested within wealthy markets, those in the MENA Asia, or Europe.

What about other components of the industry?

Twenty years ago, several Egyptian pioneers started their software companies, but where do they stand by now? In my opinion… not that far from the starting point if we to compare the same development time taken by other companies located in other geographical domains (USA,

Canada, Australia, Europe, Asia) – specially, when owned by Egyptians too! Of course these Egyptian companies started twenty years ago are now, much matured, more solid in management, and should be prepared too, to the new-game rules. But are they?

Is it possible to merge five or six major companies, to form a small giant, which can compete in the new-game market?

Fights are the good mark of competition! Yes, when Al-Alamia started more than ten years ago its case against Microsoft®, Netscape® against Internet Explorer®, EU against Media Player®, SCO® against SUN®, these were the healthy marks of tough competition. Where are our own – worthy, fights?

Again where do we stand? It will be good to our selves, if we to start a matured discussion, to answer this question, based on fresh valid statistics, as a step to the next discussion: Where are we heading to? Outsourcing? May be. Building a world-wide demanded application e.g. Audio/Video applications! May be!

Just tell me where is the demand? How profitable it is? How many people? (You will never be able to build industrial Audio/Visual applications with 30 or even 100 developers), capital is available, and good marketing, wrapped with a smart management… I tell you… you are just started planning!

Because to answer any of the above questions –and of course there are others yet to roll, you will need to conduct a professional research and studies hiring best offices world wide to build the picture. I'm talking about millions of pounds. It is not enough to bring figures about how much does India export, nor the total size of the outsourcing market. It is not enough to realize the vertical market of any specific applications e.g. CAD.

It is now important to form a Wise-Committee, which of course will be responsible to report directly to the Egyptian President. They must be young, high callipered, mix of expertise, honest, loyal to this land and understand what is a profitable investment!

The committee will consume at least two years to draw the picture. The decision will be tough and it should be, but the payback is awesome!

I will discuss in my next article, the possibility to form The Egyptian Software World Observatory for A Marketable Products and Services "EGYSWOB", as a function unit of the SECC.

Biography

Amr Shaltoot, Software Engineer, holds MBA in Strategic Marketing and Business Intelligence. Also he is a Doctoral student in Marketing Intelligence. He is interested in: Investment management, Strategic Marketing and Start Up-s.

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Author contacts:

E-Mail: amr@shaltoot.net

Unit Testing: Things to know, Things to avoid.

By: Omar Kamal

Introduction

Software systems are constructed by integrating subsystems which are build-up by software components known as units. As widely recognized in the software industry, testing is carried out at different levels: unit, integration, system, and finally acceptance testing. System testing appears to be the most cared about testing activity in our industry. The next following testing activity that engineers and managers care about is unit testing. Integration testing appears in either large scale projects or projects that heavily use "reuse strategies". This article will discuss unit testing importance, the relation between unit testing and regression testing, testing team skills, design for testability, unit test implementation, and related issues.

Importance of unit testing

Unit testing is the act of verifying that a component/unit is acting as defined by its corresponding unit design. A single defect that passes unit testing accumulates and may cause a larger number of defects that may or mayn't be catched in the following integration and/or system testing. That is why, unit test activities should be as important as system test activities, especially in large projects.

The following list summarizes the importance of unit testing:

- Reducing the testing scope reduces the debugging scope which translates into cost and time reduction.
- Unit testing doesn't need the whole software system to exists in-order to execute test cases, which allow early testing to be carried out. All what is needed is the unit test environment which won't block any other unit testing or system testing and should exist independently of other system test environment.
- Human resources needed for carrying out unit test cases may not exceed a couple of engineers for each component. The same applies for testing support processes. For example, debugging failed unit test cases shouldn't need more than the unit implementer and designer to discover the bug and fix it.
- To achieve high test code coverage (For example C1*, C2†, or P∞[3‡])for all units of the system using a strategy that uses the same system testing environment and infra-structure a lot of hard job needs to be done. It requires too much effort and time to compute those input test vectors that if entered at the system level will achieve C2 or P∞ coverage for units under testing.
- Test cases development should be guided by test coverage results. If coverage is not sufficient more test cases needed to be developed.
- Generating test input vectors

* C1: Node Coverage
† C2: Link Coverage
‡ P∞: Path testing

may require more than simple clicks on a GUI environment specially if the system interfaces with other external systems. Software systems may interface with external system using complex protocols, expensive handsets, or ATM machines, etc.... In some cases, there may be a need to purchase or rent simulators to simulate a certain protocol suite which may not be economically achievable. Building unit test stubs that simulate an external protocol or sub-system behavior reduces the costs and frequency of renting such expensive commercial simulators.

- Unit test case development doesn't need overall system knowledge or system mater expert's involvement as system test case development does.

- Different component testing can be carried-out concurrently and hence utilize human resources efficiently.

Unit testing and regression

The relation between regression and unit testing is often mis-understood from either a conceptional or an implementation point of view. The most serious misunderstanding appears in overlooking regression as factor when designing and implementing unit testing. Regression testing is the act of running the same test cases that previously ran on the same component under testing after code changes occur.

There is no need for regression testing if projects have a well defined static requirements and an implementation that never change. Practically, neither of those projects nor such implementations techniques exists. That is why, there should be means to measure the impact introduced by frequently changing the code on the system/component's desired behavior. Accordingly, regression should be considered one of the important factors when developing test environments at any level. Test automation reduces the effort and cost associated with regression cycles.

Unit testing activities in most of local Egyptian firms -at least that I knew- occurs only once. Regression (if exists) is at most carried out at the system or the subsystem level only. Although regression is crucial at the system level, it is still important to exist at unit test as-well. When a change occurs in a certain unit in a software system, it saves time and money to carry regression testing on that unit before carrying it on the system level.

Testing team skills

A number of incorrect ideas and myths exist in the mind of software engineers that affect test team selection. For example:

- Testing responsibilities are to execute and report test cases not to develop it.

- Software testing is a tedious job and has no innovative nature.

- Software tester should be aware of software coding with no need to be as professional as developers in their coding experience.

Different factors help in constructing those ideas that are far from the truth. Let us examine those factors and explain facts behind each of them:

- To successfully use white box

testing techniques, software tester examines source code closely which requires solid and wide coding experience. In addition, a great deal of development effort is needed to build custom unit test environment used to execute test cases. So, the talk about the software tester's weak coding skills is just a myth.

- Team should be educated and trained on different software testing and debugging techniques. Using the common sense to develop test case shouldn't be the norm. A lot of test cases may appear to have insignificant importance while the fact is completely at the contrary. Software managers always fail to recognize this fact, and with the rush to meet the deadlines they often put "software testing techniques" training at the end of their list. As a result, a false impression is taken about the tediousness of testing activities. As a result, engineers are left with an impression that testing doesn't require smart and innovative engineers.

Design for testability

An important factor that affects unit testing is the degree the software design takes into account the unit testing requirements. The way a system is decomposed affects the ease of testing development and execution. Every subsystem that a designer chooses to divide into smaller units needs test environments around those units to allow input test vector insertion and output test vector sensing. In general, organizing the design using commonly accepted design patterns facilitates the testing process for the following reasons:

- The reuse nature of design pattern will be inherited in their corresponding testing patterns; as a result the testing environment will be available for reuse every time their corresponding patterns are used.

- Using widely accepted and popular design patterns will provide software engineers with a great chance to surf internet resources looking for support and guidance in testing such patterns.

Interfaces between subsystems require data interaction. Complex data structure or objects (example: linked list) will complicate the design of unit test environment. If simple data-types are used to interact between sub-systems the job of building unit test environment will be much easier. Even if a decision is taken to use commercial 3rd party unit test environment for unit testing, it will be hard to find an environment that generates such complex data types.

A common pitfall is to ignore developing a data dictionary and limit the design documentation to object, class, sequence, and state diagrams. A data dictionary represents designers understanding of the intended meaning for data variables. There is no means to transfer such understanding to software testers except by close interaction between both teams or by documenting a data dictionary. Data dictionary should include valid data ranges represented by inequalities for every data variable especially system variables. Boundary values analysis which is one of the most important techniques can't be carried out without defining valid data ranges for each input/output data variable.

Testing implementation

- Unit testing is usually planned before implementation and executed directly after implementation. But a smarter idea is to avoid waiting till the unit coding is finished to start test execution. Imagine if the unit implementation is divided into small chunks and tested concurrently. In doing so, design and implementation assumptions that weren't explicitly stated are discovered and discussed by the team as early as possible.

- Test cases may fail because the testing environment itself is buggy. So, avoid using error-prone unit testing environment that is un-stable or requires a great deal of external support.

Things to take into account when planning for unit testing

- In general unit test environment is either developed in-house, or purchased as a generic framework that allows extensions and customization. Developing a unit test environment may require high investment in time and money. That is why, the decision of developing or purchasing a unit test environment should be guided by the risk associated with the overall testing process.

- The unit test environment should make it easy to add, delete, update and log test cases.

- If units/components are similar it is wise to build a generic unit test platform that allow testing any of those similar components but still allow customization.

- Two unit test environments may be used to test the same component with the same input test vectors. If one of the unit test environments is buggy, a group of test cases will fail. Comparing both test results will increase our confidence in the testing environment as well as the component under testing.

- It is commonly known that "it is preferred that the code author shouldn't test his own code". The idea seems reasonable but a code author may still execute test cases on his own code especially if they are executed for the first time. Doing so, the code author can provide feedback for the testing team about testing environment bugs that require modification and hints for test automation. One of the best approaches is to associate a dedicated peer for every programmer responsible for testing components developed by his pear.

- Problems that appear in testing documentation often have it routes in designer, developer or tester understanding of the system not in the document author ability in writing structured language. That is why, it is essential to document test cases as early as possible to allow such misconception to appear.

- When developing test case, it is considered and advantage to make each test case independent in its execution from its predecessor in order to reduce test case dependency and hence reduce blocking.

- Some times testing activities are not limited to functional black-box testing which only requires output examining for each input excitation. Checking internal code behavior may be needed to evaluate test case results. It is recommended to use tools for internal logging which are available all-over the Internet.

- The code under development often includes line of codes that are used for testing or debugging purposes. Releasing such code isn't a wise decision from performance point of view. It is required to develop means to easily separate and insert such excess code according to the need.

Things manager should avoid

- When selecting a software project manager, it is important to select a manager that has solid knowledge of the software test cycle and its required infrastructure and tools.

- Sometimes managers stressed by deadlines send conflicting signals to software engineers. Efficient software testing process should discover as much defects as possible and as early as feasible. Testing engineers may report huge number of defects that will delay the release of the product. Failed test cases require debugging and fixing effort which increases manager's stress. Unconsciously, managers view such efficient testing process as destructive and start limiting software tester's activities and scaling down their tasks.

- Managers facing continuous unexplained delays should avoid blaming the testing team without studying the real causes for those delays. Reasons for the delay in testing and debugging processes may be one of the following:

  - Unstable error-prone code.
  - Buggy test environment.
  - Highly dependent test cases that block test progress.
  - Inefficient debugging process and/or tools.

- Managers fail to notice that testing planning is as important as design and implementation planning.

- Quality managers' efforts for leveraging the testing process performance are resisted by project managers. Managers have a very common argument that they always say "Such process fits multinational organizations and isn't suitable for small software firms. Let us scale it down and gradually increase the testing process performance according to our needs". Although, the argument seems perfect but it won't work if it doesn't reflect the manager's believes. In addition, scaling down a certain process framework can never succeed without studying and/or piloting it for a while. Judging whether a process framework fits a firm or not should follow a more scientific procedure than a manager's feeling.

- Skill demands vary significantly from phase to phase according to the software lifecycle followed. Human resource managers sometimes insist in hiring permanent employees

without studying the nature of software projects. Project managers are forced to spread their needs uniformly according to such unjustified human resources hiring policy. Sometimes, a project may need too many designers at certain point of time, and then as the project progress the needs shift to software developers and so on. Accordingly, project managers face a continuous problem of assigning tasks to those under allocated resources. Even if engineers are reallocated to another job pool with a different nature not all of those engineers will keep their motivation the same as before.

- Software project managers should keep in their minds that a project should be delivered in a maintainable state.

- Software managers ignore defining common measures for bug criticality, which lead to inaccurate severity bug indicators and leave engineers in confusion. Managers may delegate such task to lead designers.

Defects handling

- It is always prefect for defect reporting tool to be centralized with remote access that allows multiple access reporting.

- Avoid ambiguous defect descriptions that explain nothing about the defect. Defect data should be enough to re-produce the error again.

References

1. Beizer, B., "Software Testing Techniques", 2nd Edition, Van Nostrand-Reinhold, 1990.

Biography

Omar Kamal, 7 years of experience in wireless telecommunications software development, training, and software quality management. Working as a senior software engineer in QuickTel Research and Development, used to work with Lucent Technologies, Hewlett Packard, and Etisalat. He Holds a bachelor's degree in telecommunications engineering from Cairo University, and master's degree in business administration from City University. Also he is a "Certified Quality Manager" by the American Society for Quality.

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Author contacts:

E-Mail: omarkaml@yahoo.com

Using Function Point to Support Estimating Software Earlier and More Accurately

By: Sally Mohamed

Software practitioners are frequently challenged to provide early and accurate software project estimates. It speaks poorly of the software community that the issue of accurate estimating early in the life cycle has not been adequately addressed and standardized. A U.S. government study on software development projects revealed the following:

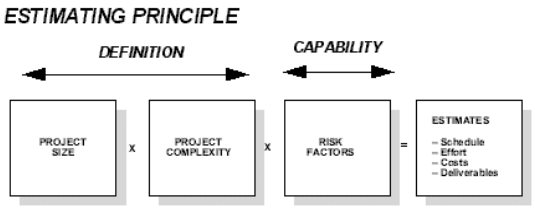
- 60% of projects were behind schedule.
- 50% were over cost.
- 45% of delivered projects were unusable

The Estimating Principle

At the heart of the estimating challenge are two issues: the need to understand and express (as early as possible) the software problem domain, and the need to understand our capability to deliver the required software solution within a specified environment. Then—and only then—will we be able to accurately predict the effort required to deliver the product.

The software problem domain can be defined simply as the scope of the required software. The problem domain must be accurately assessed for its size and complexity. To complicate the situation, experience tells us that at the point in time that we need an initial estimate (early in the system's life cycle), we cannot presume to have all the necessary information at our disposal. Therefore, we must have a rigorous process that permits a further clarification of the problem domain. Our capability to deliver is derived from the assessment of risk factors that impact our rate of delivery.

An effective estimating model considers three elements: size, complexity, and risk factors. When factored together, they result in an accurate estimate



Project Size

The project sizing technique that delivers the greatest accuracy and flexibility is function point analysis. Based upon logical, user-defined requirements, function points permit the early sizing of the software problem domain. In addition, the function point methodology presents the opportunity to size a user requirement regardless of the level of detail available. An accurate function point size can be determined from the detailed information included in a thorough user requirements document, or an adequate function point size can be derived from the limited information provided in an early proposal.

The function point method is dependent upon the identification of five elements: inputs, outputs, inquiries, internal stores of data, and

external references to data. During the early stages of development, these elements are exposed at a functional level (for example, we know that we will generate an output report, although we may not know the detailed characteristics of that report).

The first "level" of function point counting is to identify these five elements. As more information becomes available regarding the characteristics of these elements, such as data fields, file types, and so on, the function point count will become more detailed. During the early phases of a count, it may be necessary to assume levels of complexity within the system (for example, whether our report will be simple or complex). The value of using function points is that it allows for this distinction—in fact, requires it—early in the process.

Alternative sizing methods, such as counting lines of code is dependent upon information that is not available until later in the development life cycle. Other functional measurement methods, such as DeMarco's Bang and 3D, require detailed knowledge about system processing that is not available early enough for accurate counting, such as states and transitions.

Function points accurately size the stated requirement. If the problem domain is not clearly or fully defined, the project will not be properly sized. When there are missing, brief, or vague requirements, a simple process using basic diagramming techniques with the requesting user can be executed to more fully define the requirements.

Function points can be utilized in conjunction with the developed diagram to identify stated inputs, outputs, inquiries, internal stores of data, and external stores of data. For an average-size project, hours (not days) are required to complete the diagramming and sizing task.

Capability to Deliver

The capability to deliver software is based upon a variety of risk factors that influence a development organization's capability to deliver software in a timely and economical fashion. Risk factors include such things as the software processes that will be used, the skill levels of the staff (including user personnel) that will be involved, the automation that will be utilized, and the influences of the physical (development conditions) and business environment (competition and regulatory requirements). In fact, numerous factors influence our ability to deliver software in a timely fashion with high quality. Categorized here are some examples of influencing factors that must be evaluated to produce an accurate estimate.

The key to effectively utilizing these factors is centered on the development of a historical baseline of performance. An organization should develop profiles that reflect rate of delivery for a project of a given size, complexity, and risk factors. In turn, this

information can be used to predict and explore "what-if" scenarios on future projects.

An organization should develop profiles that reflect the rate of delivery for a project of a given size, complexity, and risk factors. In turn, this information can be used to predict and explore "what-if" scenarios on future projects.

Industry Data

Companies have not typically invested the resources to develop internal rate of delivery performance baselines that can be used to derive estimating templates. Therefore, industry data baselines of performance delivery rates are of significant value. The industry data points allow organizations to use these generic delivery rates as a means to "ballpark" their estimates. As they continue to develop an experience base of their own, they can transition from the use of industry data to use of their own data.

The desire for industry data is so great that many companies are willing to accept publicly available industry data at face value. Of growing concern is the fact that many providers and publishers of industry data have collected information that has not been validated, that is not current, or that is incomplete.

To avoid any such pitfalls, the following criteria should be applied when obtaining industry data:

- **For what industry is the data representative, and what is the mix of data?** Typically, industry data is a result of numerous data points that have been collected across a wide variety of industries. There is often no balance in the data; one or two industries represent the majority of data points. If they happen to coincide with your industry, then you are fortunate; otherwise, they may be of little or no significance.

- **What is the time period represented by the data?** Vendors claiming to have thousands of projects in their database are not as quick to tell you that the data has been collected over an extended period of time. If you are looking for current data, you should realize that the sampling of current data will be relatively small and may not be representative of thousands of projects.

- **How valid is the data?** Often data is collected from many sources. This limits the likelihood that the data has been validated. In addition, much of the data is of an empirical nature and, therefore, has some level of distortion.

Several opportunities currently exist for gathering, retrieving, and sharing of industry data. These organizations include The International Software Benchmarking Standard Group (ISBSG) and The Industry Software Practices Consortium (ISPC). Both organizations operate on the principle of a well-defined collection process that feeds a central repository and makes the data available for detailed access and comparison to industry best practices. The advantage of these industry databases is their accessibility of detailed data. The disadvantage is that they are relatively new and, therefore, have not yet collected a large population of data.

References

Function Point Counting Practices Manual (Release 4.2), International Function Point User's Group (IFPUG)

Estimating Software Earlier and More Accurately

By David Herron, David Garmus

Biography

Sally Mohamed, PMP has been involved with software development and management since 1995. Sally is the Quality Manager of HARF Information Technology; she has a B.Sc. degree in communication From Faculty of Engineering, Sally is also a Certified Function Point Specialist (CFPS).

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Author contacts:

E-Mail: smm@harf.com

Projects in Risky Environment.

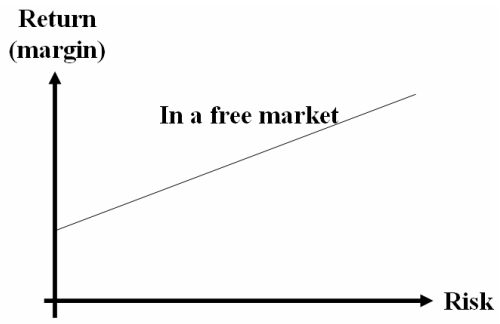
By: Yasser Taymour

Introduction

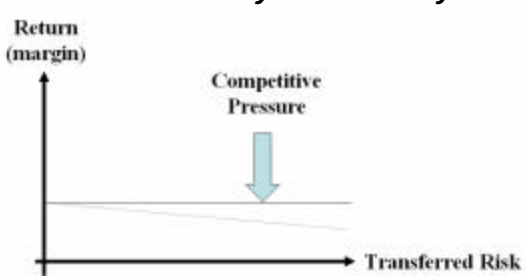
Did you finish your project plan? Are you sure that it includes all the details and you've accounted for everything? Then, what could go wrong? If it is so simple like that, why most of the project fails, or runs out of cost and schedule? Why we always face events that change the project plans and leads to disasters? The question here is, is there a way that we could follow to avoid the project unexpected events? Assessing and managing risks is the best weapon you have against project disaster. By evaluating your plan for potential problems and developing strategies to address them, you'll improve your chances of a successful, if not perfect, project.

Risk & Margin

In the free market, the margin risk is in direct proportion to the risk.



But in the reality the competitive pressures always pull the margin down. The result is "you have projects with high risk and low margin". In this case, risk management is a differentiation strategy.



The Importance of Risk Management

Project price is consists of three major elements (Project Cost + Risk Contingency + Margin), where the risk contingency is a percentage of the total probable risk cost, this amount of money related to the project probable risks. Whenever any probable risk happens, the related money is moved from the contingency to the actual cost. But if a sort of risks happened, the total contingencies amount will not be enough, where the MRGIN EROSIOIN started. The below graph shows the margin erosion.

So if the risks are not managed and controlled effectively, the erosion could get the entire margin or more. Then the project turned to loss.

Risk Definition

The risk is defined as "an uncertain event or set of circumstances that, if it occurs, will have an effect on the achievement of the business's objectives".

The risk impacts could affect one or more of the project parameters (Timescale, Resources, Cost, Quality, etc).

Risk Process

There are four major steps to assess and manage risks, and effective risk management requires all four of them:

1. Identify Risks
2. Analyze & Quantify Risks
3. Plan for Risks
4. Monitor and Manage Risks

To adequately analyze risk, you'll need a detailed plan. So, the best time to perform an initial risk analysis is just prior to saving your baseline and starting the project. But don't make the mistake of thinking that risk analysis is a one-time task. You'll want to re-evaluate the plan and your risk analysis from time to time throughout the project and whenever major deviations from the plan occur.

1- Identify Risk

There are many ways to identify risk but can simply review task list and schedule, then brainstorm and talk with experts. But generally there are some areas that usually contain/drive risks such like:

- Customer's accountabilities
- Third Parties
- External Factors
- Contract
- Politics
- New technologies
- Resources availability and skills
- Expectations
- Requirements
- Finance
- Etc.

The above risk areas/drivers hold almost 80% of the projects' risks.

2- Analyse & Quantify Risks

Quantifying risks is a discipline unto itself, where the Risk Manager should identify the attributes for each risk; the attributes of the risks determine the risk profile. The risk's attributes such as:

- Risk Causes
- Risk Drivers
- Probability (%)
- Impact
- Owner
- Risk Trigger
- Risk Release conditions
- Containment Actions
- Containment Cost
- Contingent Actions
- Contingency (Time & Cost)
- Etc.

By identifying the attributes of each risk, you will be able to plan, manage, and control the risk.

3- Plan for Risks

Once you analyse & quantify the risks you need to create action plans. You can plan for risks in one of four basic ways:

1- Avoid Risk, by changing the plan to avoid the risk, although this method could create other risks, but it is useful especially with the high impact risks.

2- Accept the Risk, if the risk containment cost is more than the contingency cost.

3- Mitigate the Risk, manage and control the risk to reduce the risk impact/ probability by identifying the Containment Actions and the related costs (Containment Cost).

NB: the containment Actions/Cost should be included in the project plan.

4- Transfer the Risk, by passing it to some one else such as insurance company, third party, Customer, etc.

4- Monitor and Manage Risks

Your risk management plan is in place. Now your job is to make sure that you and others on the project team act on it. Take any proactive actions necessary as described in your containment, mitigation, and contingency plans. Monitor your actions to see if triggers are occurring, and implement contingency plans as needed. Be sure to reassess and review the risk status regularly. Set the risk issues as essential part of the team meetings. Manage the residual risks.

Biography

Yasser Taymour, is the Quality Assurance & Partners Manager in Fujitsu Egypt he is responsible for auditing the project from the Quality Assurance and Quality Control prospects. Most of his experience in Fujitsu was in the Project Management Field.

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Author contacts:

E-Mail: **Yasser.Taymour@eg.fujitsu.com**

Discover your capability and let numbers play their rule. "*Statistical Process Control*"

By: Ahmed S. El-Shikh

In my last article I had talked about "**software measurement**", how it is important for managers in large, medium and small enterprises. Also, I went through a step by step guide for measurement program implementation, how to identify metrics categories, selection criteria, data collection methodologies and stations (i.e. probing points) , analyzing methodologies, how to build measurement culture inside your organization, tips for success, traps to be avoided and who will get benefits from all of these. Since the main focus was on the measurement process itself, I had just give a brief notation on data analysis technique, and stated that "**Statistical Process Control, (SPC)**" technique can be used to get a clear process snapshot image, long term history, accurate evaluation for process capability and testing the hypotheses about quality improvement results. In this article, we will put SPC under the lighting spot, how it is used as a quality management technique, specific goals and practices in the two process areas of CMMI level-4 and as a magnification technique (typically as the microscope) that use the measured and stored data as a test sample and magnify the hidden details in the process to enable you see amazing world of process features.

So, with a sufficient level of confidence, we can say that SPC is formally considered as a **normal extension** to measurement and analysis (MA) process area in CMMI level-2 or as a measurement and analysis **process area for adult**.

We had called the manager in an organization without a well defined measurement program the "**Blind Manager**" and considered him the same as a blind car deriver which can discover the disaster only after it happens. [Radice,1998] had called him the "**Chaos Manager**" whose job is only to drag the project out form the chaos, if he is lucky of course. What ever he is blind, chaos or even fire fighter, his maximum ability is just to get the process back to the brink of Chaos (i.e. a state which is erroneously considered to be "out-of trouble" in most operations).

SPC is the **useful** or even the **only way** that can enable all these types of managers to get out of this cycle of despair and move a process up to the threshold or the ideal state.

At the moment of writing this article, we have **five companies** that have conducted an assessment or appraisal at a maturity level-3 (two with CMMI and three with SW-CMM); I think that this is **the right time** for the software community in Egypt to raise the awareness about the important backbone of CMMI level-4 (i.e. the SPC).

This does not mean that SPC is applied only for large companies or higher matured ones; it can be applied in **SMEs** even in maturity level one (under some pre-requisites that we will talk about latter). I just mean that now the software community is aware enough about the quality techniques and CMMI, and it will be easier to go just a step further than build the whole story from its beginning.

[Radice,1998] had mentioned several different **definition for SPC**, but generally specking, we can say that SPC is a formal mathematical and statistical technique -that implies a set of tools- in order to describe the

natural behavior of the process and process variation.

The organizations that apply the SPC technique get a lot of **benefits** that can be summarized in the following:

- Identifying processes or process elements that need and should be statistically managed.
- Distinguish between processes that behave consistently or show stable trends (i.e., predictable)
- Identify processes that show unusual (i.e., sporadic or unpredictable) behavior.
- Identify any aspects of the processes that can be improved.

See [SEI/CMMI-Stage-2002] for more details. From the **business point of view**, we can summarize the long term benefits as:

- More precise estimation due to increasing in process stability.
- Improving the overall company performance in production, market share or simply more return on investment (ROI).

The **basic successful factor** in applying SPC is the robust foundation, which has implemented in level-2 and level-3, including the formally defined processes and implementation of measurement and analysis process area with accumulated high quality data that are stored in the measurement repository.

When you take the decision to **move from maturity level-3 to level-4** just be careful, you had already done some simplified data analysis, but not in the same depth as in SPC. You had just described the threshold limits that exceeding them will trigger immediate and unplanned corrective action, but in level-4 you will describe your objective (internally stated or allocated form outside) side-by-side with threshold levels in order to compare both of them. In level-4, you will use and understand your historical data – collected in level-2 & 3- to **predicate your future**.

This leads us to the **simple idea behind SPC**, "*History repeats itself*". So, from Logical and business point of view, ignoring SPC can be considered as mindless or even mad behavior.

It is evident and clear that **delaying consideration of maturity level-4**, until conducting it, will:

- Exhaust the measurement program's team in collecting a well specified new data.
- Delay archiving maturity level-4 by a time period that is long enough to build the required historical data.

So, the **golden rule** is to take CMMI level-4 into your account while you building the measurement and analysis process area in level-2.

Formally, **SPC appear in the CMMI** maturity level-4 in the following section:

- **OPP** (*Organizational Process Performance*) Process Area:
  - **SG 1**: Establish Performance baseline and model.
- **QPM** *(Quantitative Project Management)* Process Area:
  - **SG 1**: Quantitatively Mange the Project.
  - **SG 2**: Statistically Manage Subprocess Performance

SG 1 in OPP is all about the discovering and precisely describing the "**Voice of the Process, VOP**".
SG 1 in QPM is about formalizing the "**Voice of the Customer, VOC**", including voice of the internal

customers that appear in the organization objectives and voice of the external customers that appear as allocated specification or market force and competition. SG 2 in QPM is the fruitful phase in the SPC story; **Process Capability**[§] can be calculated as a ratio between VOC and VOP. It is evident that maturity level-4 is all about three specific goals that cover the basic three steps for calculating the selected processes' capabilities.

There are a **wide range of processes** and subprocesses that can be eligible for applying SPC, according to the **SIPOC**[**] process mapping methodology; we can use SPC for supplier evaluation, indoor elements such as process, product and resources, or for customer feedback analysis.

Although SPC is the backbone for CMMI maturity level-4, it does not mean that it is not necessary for lower level companies. Also, it is not a new fashion that has recently appeared in the world of the software engineering science of industry. **SPC is an interesting tool** for any software company whatever its maturity level or size. Some points that drive companies towards applying SPC are as follows:

- You want and need to satisfy your customers.
- Your competition is not relaxing.
- You want to have a competitive advantage.

[§] Some topics need a previous knowledge of statistics and problem solving techniques, which is outside the scope of this article. Some will be explored in brief. "*Measuring the Software Process, Statistical Process Control for Software Process Improvement*" for William A. Florac and Anita D. Carleton, is a good book for more details.

[**] SIPOC is a process mapping methodology that can be used different purposes, such as to determining measurement stations and propping points, for more details see "Juran Quality Handbook".

As we just see, the **main driving force** towards applying SPC is not a technical one, but it is the market.

I know that the **idea of applying SPC** can be more accepted from the mangers in the higher maturity companies, they had already touched the benefits of applying CMM or CMMI quality model, have a qualified team, a well pre-defined processes' infrastructure and a successful launched measurement program that enriched the measurement repository with a lot of helpful and valid data history.

The **basic barrier**, which will face the trials those tend to **apply SPC in SMEs** or lower maturity companies, is the top management commitment. Normally due to the lack of all previous points that had mentioned as power factors in higher matured companies.

The **second barrier** can be the lack of sufficient level of knowledge in SPC; very good skills in the field of mathematical computation, statistically techniques and the basic quality tools is required even if a computerized SPC kit is used.

The **last barrier** is that it can be difficult to take the agreement or commitment, from top management, to use or even investigate the SPC at all. In this case, you will need to take the "*Cognitive Dissonance*" concept into your consideration and do not try to change people attitude at first, be careful!

We can exceed this limit and say, to a high degree of confidence, that SPC can be applied even in the **lowest matured companies** (i.e. those in the level-1). Be calm, not all companies in level one can apply SPC, only ones that have initiated and lunched a measurement program, even if it is naive or do not completely cover all requirement of the

measurement and analysis process area in CMMI level-2.

You can **start with** a very simple process in your organization, which even does not require a well defined measurement program and therefore can be conducted in the lowest matured level, such as the "*attendance and departure*" process. It is the most powerful candidate that fulfills the previous conditions. If you try SPC, you will discover very strange patterns of employees' behaviors that were very difficult to be discovered without SPC.

Other simple process can be considered in your first steps towards the SPC horizon. Some processes are, by natural, automatically managed, measured and auditable (i.e. matured enough). Web hosting servers usually record a "*web log*" files that contain a large amount of real data with high quality degree. Analyzing web log files was my first naive trial with SPC, but I had learned a lot from it.

We are now ready to go throw a **brief journey throw SPC** technique. As we mentioned above, the main target of SPC is to quantitatively describe the natural of the process variation. Average and range can come to your mind as a helpful **statistical parameter** that can describe the process variation.

Each one of them has its **limitations**; average can not describe the divergence (loss of precision) of the sampled data, range can not describe the shift (loss of accuracy) in the average. You can think that combining both of them in addition to the standard deviation to characterize the histogram (simply the probability distribution) of the process.

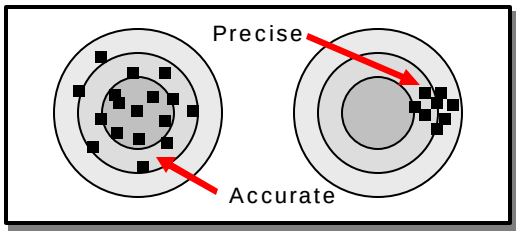


Fig.1: Precision vs. Accuracy

Although **histogram** is one of the quality basic tools, it is just a snapshot of the process behavior at the moment of process examination; formally it shows an accumulated image for the process history from the starting of measuring activity until the moment of histogram construction. We are in need to another tool that can discover a wide band in time domain.

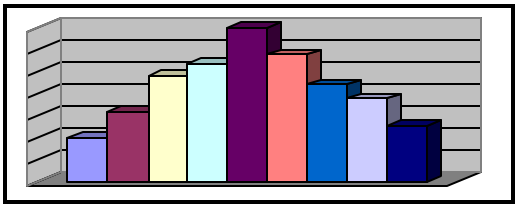


Fig.2: Histogram (normal distribution)

Control chart is the suitable tool for studying process variation, and it is also the heart of SPC. Control chart is a tool that shows the long term behavior of the process through sampling, recording and plotting process activities or outputs.

There are several **types of control charts** fall into two main categories, variable control charts such as (average & range), (average & standard deviation) and attribute control charts such as (np-chart) , (p-chart) and (u-chart). Some of them are more suitable than the other to the software industry.

After selecting your process, subprocess or process activity that you want to manage successfully, a systematic sequence of steps is required to build any control chart.

These are the steps:

- Choose a rational subgroup.
- Collect the data (skip this step if you had already collected them).
- Calculate the trial central and control limits.
- Plot the chart (limits & points).
- Determine, investigate and eliminate the special (assignable) causes that go behind limits.
- Calculate the revised (stable) control limits (Lower Control Limit, LCL & Upper Control Limit, UCL).
- Achieve your objective.

For more details, review the CMMI level-4 process areas, specific goals, specific practices and sub-practices and any SPC book.

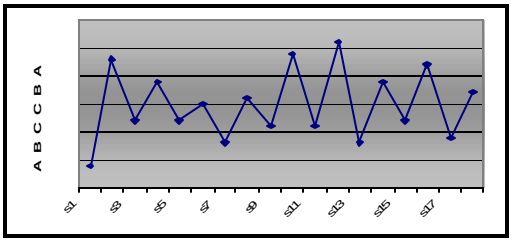


Fig.3: Control Chart

While studying the process variation, which is now visually seen on the chart versus time, you will notice **two types of variations**. Points outside control limits are due to assignable (**special**) causes that can be eliminated by the immediate corrective actions (better to be planned). The other type is the common (**random**) causes that need a long term improvement effort and investment to be eliminated.

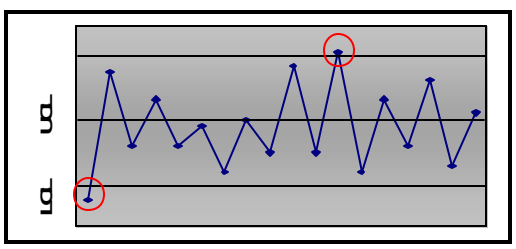


Fig.4: Special Case variations.

Note that **trial natural bounds** are calculated from the data recorded in CMMI level 2 & 3. Revised bounds, which describe your process voice (VOP= is your production ability), is calculated after taking corrective actions and maintained by preventive ones, which all are done in CMMI level-4.

Enhancement in central line, control limits and variation need improvement efforts, which will be addressed in CMMI level-5. One of the most famous process improvement methodologies is the "**Six Sigma**"[††].

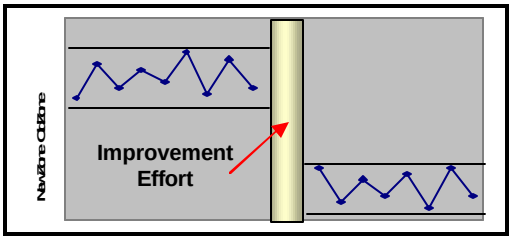


Fig.5: Process Improvement Result.

Just remember that, **mapping SPC activities to CMMI** maturity level does not mean that it can not be done outside the umbrella for CMMI, you can use some SPC computerized tool that help you analyze the recorded web log (i.e. do automatically the complete previously mentioned steps) in less than one minute.

If you had already defined your **process objectives** (i.e. calculated the voice of your customer whatever he is internal or external, VOC), you are now ready to calculate your process **capability index** (C_p) which equal to the ratio (VOC/VOP) or equal to ((USL-USL)/ (UCL-LCL)). Note that VOC is equal to the difference in specification (USL-LSL), and it is the

[††] Six-Sigma is a management strategy that maximizes customer satisfaction and minimizes the defects that create the customer dissatisfaction. It has (DMAIC, DMADV and DFSS) frameworks that will be conducted in future article.

real allocated or expected level of performance or specification limits from internal customer –top management-, external customer or marketing competition and force.

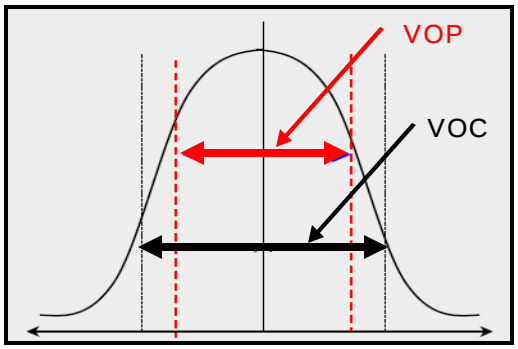


Fig.6: Special Case variations.

Internal customer process objectives can be constructed by many ways, from the simplest threshold level as a target to the using of the "**Quality Function Deployment, QFD**"‡‡ tool.

Now we are ready to state some **precise definitions**, which were impossible to be described without the SPC. In level-2, we had already conducted the definition of immature and mature process, but now we focus on the process stability and capability and tolerance. Best words to define **process stability** are what had been said by Walter A. Shewhart in 1931, "*A phenomenon will be said to be controlled when, through the use of past experience, we can predict, at least within limits, how the phenomenon may be expected to vary in the future*". Using the language of control charts, the process is said to be **stable process** if and only if no more than three points of each thousand is out of the revised control limits.

Process instability can appears in two forms. The first is when there are points out of control more than the normal number. The second appear in several situations, we will mention here only two of them due to space limitation:

- 7 points in a row at one side of the chart.

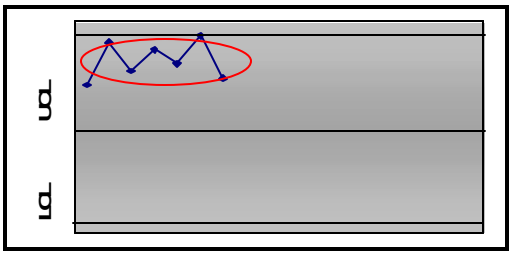


Fig.7: One side raw instability.

- 6 points in a row show steadily increasing or decreasing.

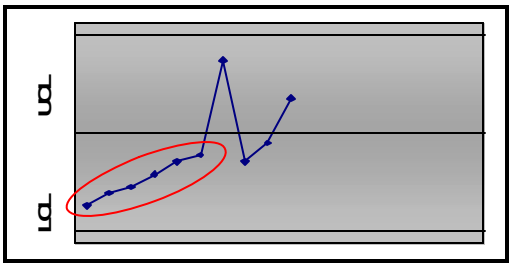


Fig.8: Increasing raw instability.

Instability analysis in the first condition is just due to assignable causes or a sample from a different population. In the second case it may be due to:

- Change or jump in level.
- Trend or steady change in level.
- Recurring cycles.
- Two or more populations.
- Mistakes in measuring.

The process is said to have a **tolerance** if the VOC (i.e. USL-LSL) is greater than VOP (i.e. UCL-LCL). In other words, the tolerance exists if the (C_p) is greater than the unity. Process tolerance enables the (intentioned or natural) **average offset** from the desired or allocated central line. This offset has the maximum value of [(VOC/2)-(VOP/2)] in any one of the

‡‡ Quality Function Deployment is a structure approach that enables the integration of customer requirement (i.e. VOC) in the *product* or *process* development process.

two sides of desired central line. If this tolerance is allowed, the (C_p) is not the optimum indicator for the process behavior. It is better to use the (C_{Pk}) if you want to evaluate your process against the customer allocated specification. Also, use the (C_{Pm}) to evaluate the process capability against a performance target. It is beyond the scope of this article to go through the mathematical details of these different types of capability indexes. Try to read any specific SPC book to explore capability and performance indexes.

Control charts in not the only tool in SPC, Ishikawa, in 1991, defined the **seven basic quality tools** to be "Check List, Pareto Diagram, Histogram, Scatter Diagram, Graphs and Fishbone Diagram and Control Charts" of course. The seven basic tools is only a subset the "**Quality Analytic Tools**" mentioned in [Radice, 1998]. These tools can be used in problem identification, problem analysis or both.

SPC is very interesting subject; there are still a lot of topics that need more clarification. The most important notation is that SPC is **not the whole story**. If you stopped after exploring SPC world, you are just the same as who had investigate the test sample, of an ill man, under the microscope until he knew the root case of the disease, then he left the ill man without a remedy. The **quality remedy** will be conducted in the CMMI level-5, which focuses on the quality improvement methodologies. We will explore the horizon of **Six-Sigma integration with CMMI** in the next article, if the God willing.

References

[Barnard,1998] Julie Barnard, "Enhancing Our Data Analysis Capabilities: Applying SPC Techniques to the Inspection Process". The 98 Software Engineering Symposium, Sept., 1998.

[Carleton,1999] Anita D. Carleton and William A. Florac, "Statistically Controlling the Software Process". The 99 SEI Software Engineering Symposium September 1, 1999.

[Kaliappan,2000] M. Kaliappan, "Applying Statistical Control in Software Development" , in the proceeding of the second world congress on Software Quality (2WCSQ), Japan, Sept., 2000.

[Radice,1998] Ron Radice, "Statistical Process Control for Software Projects". Delivered at 10th SEPG Conference Chicago, Illinois, March 9, 1998.

[Radice,2000] Ron Radice, "Statistical Process Control In Level 4 and Level 5 Software Organizations Worldwide". Delivered at the 12th SEPG Conference Seattle, Washington, March 22, 2000.

[SEI/CMMI-Stage-2002] "Capability Maturity Model® Integration (CMMI℠), Version 1.1", (CMMI-SE/SW/IPPD/SS, V1.1) Staged Representation, 2002.

Biography

Ahmed S. El-Shikh is the QA Manager in Systems Advisory & Solutions, (SAS) (S.A.E.). His interests include software engineering aspects, software quality management, statistical quality control and process Improvement approaches specially the "Six Sigma, DMAIC & DAMDV".

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Author contacts:

E-Mail: el_shikh@sas-sys.com