



Egypt–SPIN Newsletter

Issue 9, Jan. – Mar., 2005

Sponsored by SECC

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

Welcome to our 9th 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 6 months events hoping to provide our members of Egypt – SPIN with information to support their current interests.

Feb. 2005 to May 2005, **Personal Software Process (PSP)** and **Team Software Process (TSP)** training track is conducted in Egypt and is sponsored by **SECC** with collaboration with **USAID/ICT**. For more details, see the next page.

In this issue, you can find practical experience in process improvement (1st article), completing the software testing tutorial series (2nd article), and hot new topic in process improvement (3rd article).

Eng. Reem El Ansary shares her **experience in software process improvement in ITWorx** with the community. She gives important guidelines and some lessons learned.

Eng. Omar Kamal completes his series about **software testing techniques**. He gives the second part of the **control flow testing techniques tutorial**.

Eng. Ahmed El-Shikh introduces the **Six Sigma** breakthrough improvement methodology called **DMAIC** as the first article in his new series about **“Metrics-Based Process Improvement Series”**.

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

PSP/TSP Training Track in Egypt for the First Time.

By: the editor

With participants from 20 IT companies in Egypt, the **Software Engineering Competence Center (SECC)** in collaboration with **USAID/ICT** program had conducted a training track for the Personal Software process (PSP) and Team Software process (TSP) for the first time in Egypt.

The PSP/TSP training track is delivered by the Software Engineering Institute (SEI) and instructed by **Mr. Daniel M. Roy**. Daniel is the president of the Software Technology Process & People Inc. (STPP). He has around 38 years of experience in the software industry in many fields such as CMM, PSP/TSP, Risk Assessment, Real time systems development methodologies, software measurement, metrics-based improvement paradigms and human aspects of software techniques (MBTI and P-CMM). He also has a lot of publications in the field of CMM, CMMI, PSP/TSP, Formal Inspection and eXtreme programming (XP).

The **training track** contains four courses and one executive strategy seminar. The track is delivered twice on two rounds; each has participants from 10 companies. The first round started in February, 2005 and will end in April, 2005. The second round will start in April, 2005 and will finish in May, 2005.

The **executive strategy seminar** is a two full day's seminar that focuses on giving an overview on the PSP/TSP from different perspective. It starts with the strategic issues in the software business, goes through the common software production problems and highlights the benefits from introducing PSP/TSP to the company.

PSP for Engineering I: Planning 5-days course and **PSP for Engineering II: Quality** 5-days course are two courses, which form the heart of the track. They cover the main aspects in the PSP such as size measurement, proxy-based estimation and process measurement. They also conduct some advanced aspects such as design process, design verification, scaling up to TSP and integrating PSP with TSP. The attendance of these courses should plans to spend around 120 hours to complete both courses plus course assignment, pre-reading and post-class homework.

Managing TSP Teams is an important 3-days course that focuses on some managerial aspects regarding coaching TSP team such as TSP launching process and working process. It helps the project mangers, team leaders to understand their roles in applying PSP/TSP in the organization.

Finally, the 2-days course called **Introduction to PSP**. It gives a good overview for any role in the company that will be in contact with the TSP team such as test engineers, technical writer and other non-software engineers. The course covers the PSP framework, the personal process, measurement aspects in PSP and finish with and introduction to TSP.

By conducting this training track, the quality improvement efforts sponsored by SECC in Egypt has been focused on individual's skills improvements beside the overall company maturity level improvement. Hope that these tracks provide a good leverage for the software industry in Egypt.

Table of Contents

ITWorx Improves its processes	4
Software Testing Techniques Series: Control Flow Testing Tutorial. (Part 2.....	7
Metries-Based Process Improvement Series Let Us Discover another Buzzword "Six Sigma".....	14

ITWorx Improves its Processes

By: Reem El Ansary

About ITWorx

Founded in 1994 and privately held, ITWorx is a professional IT services firm focusing on the development of IT solutions for Global 2000 companies, and custom application development for some of the biggest names in the software industry. ITWorx is a Microsoft Gold Certified Partner, an Oracle Partner, an IBM Business Partner, in addition to being CMMI Level 2 and ISO 9001 certified.

Lessons Learned

There is no need to say why ITWorx seeks process improvement or the added values of process improvement. The journey of process improvement at ITWorx started in 2000. Before that the size of the company was too small (around 30 employees) so the performance depended mainly on the company's heroes not on an organization defined system.

Throughout the journey of ITWorx with process improvement we have learned a number of lessons that we would like to share with others to help them avoid our pitfalls.

The lessons can be categorized into three areas:

- o Organization management
- o Process definition
- o Process execution

Organization Management

The organization structures and the senior management commitment affect the achieved level of process

improvement. Once ITWorx decided to start the process improvement journey, the following activities took place:

Selecting an Executive Sponsor

Start with the CEO and work down. Having the right sponsor of the process improvement project may be all that is needed. After all it is often not "what the project can achieve" but "who wants it to achieve" that counts. At ITWorx, the Managing Director was the process improvement sponsor that facilitated a lot throughout our decisions and prioritization.

Restructuring the Organization

For sure, process improvement activities need a dedicated team and some organization restructuring. ITWorx restructured the organization chart to reflect the need for the process improvement initiatives. Three main groups were added to the organization chart: the Software Engineering Process Group (SEPG), the Software Engineering Measurements Group (SEMG) and the Quality Assurance group. The three groups report directly to the CEO.

Identifying the Goal of Process Improvement

One of the main goals of ITWorx was to build a resilient company. The way we see this goal achieved is to go for process improvement. The main goal was not merely to achieve CMMI Level 2. Achieving a certain level is only a way to evaluate how the process improvement effort is progressing.

Building Different Communication Channels

Effective communications channels are implemented to build stakeholders' commitment to the process improvement project and to the new processes. The objectives of these channels are as follows:

- Build acceptance and ownership of the system across the organization and the business units.
- Build commitment to the system.
- Sell the benefits of the system.
- Manage users' expectations and information needs.
- Provide information on available support and resources.
- Provide information about the system to the impacted audiences and obtain their input.
- Reduce uncertainty and fear of the new technology and processes.
- Develop a framework to coordinate all communication efforts.
- Ensure that consistent messages are delivered to stakeholders at all levels and that decisions, events and activities are communicated in a timely manner.
- Make communications an integral part of the daily activities of the project's team.
- Prepare a feedback channel from employees on the new system.

Process Definition

On the level of the process definition, ITWorx was able to define its own processes through forming Technical Working Groups (TWG) for defining each process. Each TWG is responsible for making a gap analysis between the

current best practices that are already followed in the company and the CMMI process guidelines. Then the TWG has to define a process that make use of our best practices and at the same time conforms to the CMMI model. Through the processes definition we have learned several lessons that can be summarized as follows:

More Involvement in the Process Definition

Practitioners should be involved in defining the processes for several reasons. First, this way you are most likely to have their buy-in for the new way of the world. Second, because they are the real resources who know the weaknesses and strengths of the existing best practice or process.

Process Evolution

One of the most pitfalls that we went through was that we were always seeking perfection in defining a process. Therefore, we used to spend so much time in defining processes. We have reached that it is better to try to pilot a simple process and then evolve it by time. On defining a process, ITWorx usually takes into consideration the capability of the resources that will be performing it and the time constraints that may face them. The supporting tools needed for the process execution should be defined as well, to help the resources carry out their jobs successfully.

Setting a Priority for the Process

CMMI as a quality model should give you guidelines on the processes that you need to look at but the priorities of defining these processes should be derived from the business needs. The resources should feel the pain they face of not having a certain process to guide them in certain activities in order to follow the newly defined process.

Tailoring a Process

No standard process can be applied to all types of projects. Tailoring guidelines should be defined for each process. Quality Assurance auditors should understand the process very well in order to judge whether this tailoring is acceptable or not.

Process Execution

Executing processes is the most difficult part of the process improvement cycle. Successful process execution needs the resources to buy-in together with management commitment and client awareness of the importance of internal processes.

Process Pilot

Always try to pilot any new process before deploying it on the whole company in order to get the feedback from the pilot project and measure its effectiveness. The selection of the project to be used to pilot the new process should be wise enough in order not to affect the project itself. Listed below are the criteria for selecting a pilot project:

- The deadline of the project is not critical to the client.
- The effort estimate of the project includes sufficient contingency factor.

Process Training

Before the piloting or the deployment of any process, training sessions should be conducted by the TWG, which previously defined the process, to all the resources who are going to carry it out.

Process Measurements

The effectiveness of any process should be measured, to check whether

the newly deployed process affected the software life cycle positively or negatively. At ITWorx, we measure the progress of projects before applying a certain process, and then measure their progress after applying the new process.

Feedback Channel

A feedback channel should always be open for the practitioners to provide their feedback regarding the processes to the Process Improvement team. At ITWorx we created an e-mail account for the process improvement team where users can use to send their feedback on any aspect of the processes.

Biography

Reem El Ansary, the Quality Manager at ITWorx since 2000, is responsible for the quality assurance and the process improvement. Reem graduated from the AUC with CS major & Business Administration minor in 1992. Before joining ITWorx Reem was a Technical Team Leader at Saudisoft.

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:

reem.ansary@itworx.com

Software Testing Techniques Series: Control Flow Testing Tutorial. (Part 2)

By: Omar Kamal

Previous Article Summary:

The previous article introduced the Control Flow Graph (CFG) as one of most important techniques used in software testing at its different levels. The way the CFG is developed was explained in details together with different coverage criteria. This article will continue to demonstrate the way test cases are developed from the selected paths, how to convert the CFG to another representation that allow the testing process to be automated, and finally how to develop an infrastructure for the automation process. Note that it is hard for readers how didn't read the first article to recognize technical concepts and terms in this article. For this reason, I encourage readers to carefully read the first article.

How to generate test cases from the CFG:

After selecting paths that satisfy any of the criteria mentioned before we write down the paths in a paper in order to find external inputs that force each path to occur. This process is called path sensitization. The set of input variables associated with a certain path is called an input test vector. Input test vectors together with their corresponding *Path-Ids* are entered in a spread sheet. The analysis process is repeated again for each path in order to predict expected outcomes. When a path is selected the set of outcomes that cross the boundary of the component under testing is called the output test vector for that path. Again, output test vectors together with their

corresponding *Path-Ids* are entered in a spread sheet. Doing so, we have the complete data that define all test cases needed to meet the selected coverage criteria. The following section will demonstrate the path sensitization and outcome prediction processes based on the previous article's case study.

Path sensitization and outcome prediction for room reservation example:

Figure 1 represents the requirements for the software application that we intend to test. We will choose strong coverage criteria (P_{∞}) to govern the path selection process. Note that there is a non-deterministic loop between node C4 and C5. For simplicity, we will select a path that covers this loop once and another path that never pass this loop. Loop testing techniques are out of the scope of this article. Further information about loop testing can be found in [BEZ1, BEZ2]

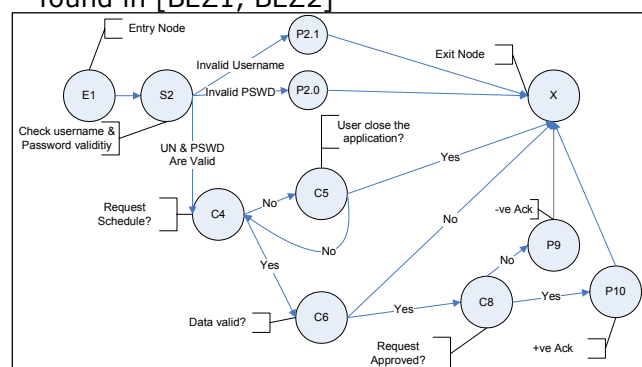


Figure 1: Control Flow Graph.

All combinations of paths are:

1. E1,S2,P2.1,X
2. E1,S2,P2.0,X

3. E1,S2,C4,C6,C8,P10,X
4. E1,S2,C4,C6,C8,P9,X
5. E1,S2,C4,C6,X
6. E1,S2,C4,C5,X
7. E1,S2,C4,C5,C4,C5,X

We won't develop test cases for all paths due the limitation of the article size. Let us select path 3 and develop its corresponding test input/output vector for the sack of demonstration.

Input test vector for path 3:

- o User enters a valid username & password.
- o User request Schedule.
- o User enters valid request data.
- o The administrator approves the user's request.

Output test vector for path 4:

- o The user should receive a positive acceptance acknowledge.

It may seem that by developing input/output test vector the test design process has reached its end. This is seldom true, there are cases where certain test cases has no meaning in the real world or even can never occur physically. It is required to filter out those test cases before starting the test execution phase.

Removing path redundancy and contradictions:

Sometimes, a path can never occur because two or more of the nodes can never exist in that specific path (node contradiction). Similarly there are cases where the existence of *node A* in a certain set of paths limits the choices for a following *node B*. For example, if a person has never been married (Condition A = never married) his/her (number of childrens should be limited to zero). Those situations generally occur when modeling large problems.

Therefore, one should examine the dependency between nodes carefully.

Test execution

The only left task is to put the testing process into action by developing the test execution environment. In general, there are two choices:

- o To develop the test code responsible for inserting input test vectors, fetching output test vectors, comparing actual output with expected output, and finally reporting the result.
- o To use commercially or 3rd party free tools that do the same thing for us.

A need for another representation for the model

Using the CFG in visualizing the code and developing test cases is very useful. In addition it is usually easy to develop by hand. Although this is true for small piece of logic, it may not be true for a medium-to-large logic flows. Another way of representing the model is suggested by Bizer [BEZ1] and modified a little by the author of this article. The following section describes the new model representation.

The Control Flow List

Each node is represented by a record in a list of statements. Each record is listed in one single line except for the "Switch" node which is represented in multiple lines. The record consists of a number of columns separated by a certain delimiter. The first three columns in the record as shown in Table 1 are:

- o *Node Id*: The node Identifier is a unique identifier for the node, it can be a serial auto-

incremented number or it can be an alpha-numeric string. The node identifier is used for referencing the node and hence establishing connections between nodes.

- *Node Type*: This field represent whether the node is an entry, processing, condition, switching, or an exit node.

- *Node description*: This field holds a meaningful text description for the node. For example, if the node is a condition node that checks on the following condition "if(Username == "Valid")" then the node description may be "Checking the username is valid".

Node ID	Node Type	Node Description	next reference	true reference	False reference	Case 0/Goto 0	Case 1/Goto 1	Case 2/Goto 2
E1	E	Entry	S2					
S2	S	Check username & Password validity				Invalid Username	Invalid Password	UN & PSWD are valid
3	G					P2.1	P2.0	C4
P2.1	P	The application pops up a message indicating the username isn't valid	X					
P2.0	P	The application pops up a message indicating the password isn't correct	X					
C4	C	Request Schedule?		C6	C5			
C5	C	User close the application?		X	C4			
C6	C	Data valid?		C8	X			
C8	C	Request Approved?		P10	P9			
P9	P	The user should receive a rejection ack.	X					
P10	P	The user should receive an acceptance ack	X					
X	X	Exit						

Table 1: Control Flow List

The first three columns are mandatory for each node regardless of its type. Depending on the *Node Type* the rest of the columns will be either applicable or not applicable.

- *For processing nodes (P):* Since this type of nodes do some kind of processing and just points to a next successive node, the only valid column is the *next reference* column. The *next reference* holds the node id for the next node.
- *For the entry node (E):* It is the same as the processing node, it just point to the next successive node. Hence, the only valid column is the *next reference* column.
- *For the exit node (X):* The exit node doesn't point to any node. Hence, the rest of the columns are left empty.
- *For the condition node (C):* There are two states for any condition. Either the condition is true or false. That is why; there are two valid columns for that type of nodes. The *true reference* node which holds the following *Node ID* if the condition evaluates to true. The *false reference* node which holds the following *Node ID* if the condition evaluates to false.
- *For the Switch node (S):* The switch node is a bit complex in it's representation since the

numbers of cases are not limited as the case for the condition node. In addition, a condition node has either a true or a false alternative which are self descriptive, but for the switch node we don't know implicitly the meaning of its possible alternatives. Accordingly, a switch node is represented in two subsequent lines. The first line holds the description of the switch node itself as-well as the description of all its cases. The line doesn't hold any reference to any node, it just describe the switch node with all its cases. Each case is described in a single column, starting from the 7th column.

For example: Let us assume we are modeling the following switch node:

```
Switch (Operating System)
{
    Case WindowsXP:
        doSomething;
        break;

    Case Linux:
        doAnotherThing;
        break;

    Case Solaris:
        doNothing;
        break;
}
```

The first line of representation will be as follow:

Node ID	Node Type	Node Description	Next Ref.	True Ref.	False Ref.	Case 0/Goto 0	Case 1/Goto 1	Case 2/Goto 2
S2	S	Check username & Password validity				Invalid Username	Invalid Password	UN & PSWD are valid

Now, since we are done with the first line let us move to the second line that completes the switch node representation.

The job of the second line is to connect the switch node with n-nodes corresponding to its n-cases. Hence, for each of alternative/case the *Node Id* that represents the following node is written below that case.

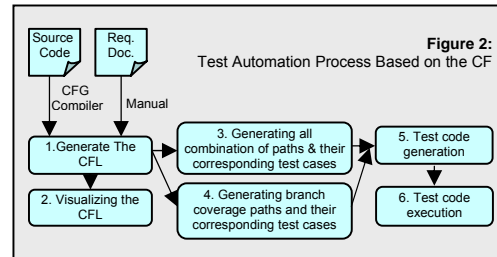
Since all cases should have corresponding go-to nodes we label this line as the Goto line. A Goto line is a must with a switch line, although both represent a switch node in the logical model. To make things uniform and allow automatic entry for the data using software application, we introduced the "Goto" node type that is tightly coupled with the "Switch" node type. The "Goto" (G) node has no equivalent meaning in the previous Control Flow Graph representation, it just simplify the data entry.

Test Automation Process Based on the CF Model:

As mentioned in the previous article, a (Control Flow Model) CFM can be generated from any phase and at any level of details. For example, a control flow graph can be generated directly from requirements or from a low-level design documents or from the code itself. If we develop the CFM from the requirements document then all test cases developed later from that model are related to the black-box testing process.

If on the other hand, the CFM is developed from the code then all test cases developed later from that model are related to the white box testing process.

The following list describe a simple approach for automating both the black-box and white-box testing process based on the CFM.



1: Generating the CFL:

The first step is to develop a CFL from the:

- Requirements document, which can't be done automatically.
- Source code, which can be done automatically using special compilers developed using Lex and Yacc, or using 4th generation languages like Perl, Python, or Ruby. The job of the compiler or the script is to travel through the code searching for "if" conditions, "switch-case" blocks, entry, exit nodes, and important processing nodes marked by special pre-defined tags and produce the CFL.

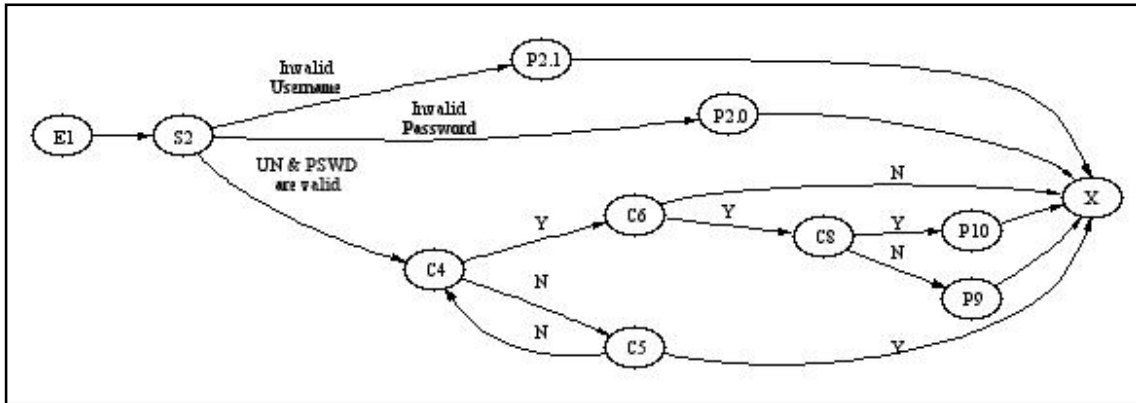
2: Visualizing the CFL:

Having a CFG will help finding discontinuities in the control flow because the human eye can catch path discontinuities from a graph easier than to catch it from the textual list.

The objective of this step is to automate generating a CFG from a CFL. There are a number of tools and software 3rd party libraries that produce visual graph from intermediate textual representation. I strongly recommend AT&T Graphviz library which is very powerful in drawing large graphs with proper layout management. Graphviz uses an

intermediate language/notation called "dot" to describe the graph to be generated. If we successfully develop an adaptor to convert the CFL to the "dot" notation and passing it to Graphviz, then the job is done. Figure

3, represent an example of a graph that was automatically generated from a source code using AT&T Graphviz together with a simple locally developed CFL-to-Dot Perl adaptor.



3: Generating all combination of paths and their corresponding test cases:

Generating all combination of paths (P_{∞}) is optional. Recall from the previous article, the strongest criteria in coverage is to cover all combination of paths which is a decision that has to be taken by the test engineer depending on the risk, time, and budget constraints. Anyway, there is nothing to loose if the task is fully automated. Developing a simple C/C++ program that takes a CFL as its input and produce the following output files:

- o A file containing the description of all paths.
- o An excel sheet or a "comma separated values" text file containing the values for all conditions and switch-case nodes which if forced the corresponding path will be taken.
- o A text file that holds all processing nodes that occur if their crosseponding path was selected.

4: Generating branch coverage paths and their corresponding test cases:

As mentioned before developing all combinations isn't economic specially if those test cases will be executed manually. Accordingly, there is a need to develop a program that computes the minimum set of paths that cover all branches. A number of algorithms exist that take all paths as its input and calculate the reduced set of paths. The simplest although time-consuming algorithm is descibed below:

- o Count the branch appearance frequency (BF) for each branch in the CFL. The frequency is calculated by counting the number a specific branch appears through all path combinations.
- o Sort the branches from lowest frequency to highest frequency.
- o Count the length for each path (L) computed as the number of nodes that compose the path.
- o Sort the paths according to their length from bigger paths to smaller paths.

- Now we have two lists:
 - Branches List.
 - Paths List.
- Select the biggest possible path which includes the lowest frequency branch.
- Analyze the selected path and remove out all the branches found in that path from the *Branches list*.
- Repeat the previous two steps until all branched are covered by as much big paths as possible.

The algorithm won't get the minimum set of paths but will get something so close to the minimum. The algorithm can be easily implemented using C/C++ or even 4th generation languages.

5: Test code generation execution:

The output of step 3 include the conditions that force each path to occur as-well as all processing that take place from the beginning till the end of that path. Generating the corresponding test code is not always an easy job. Using a data driven code generators, test templates, or test database can make the job feasible. In general using generic testing frameworks with or without tailoring simplifies the overall development and execution job. There is a great advantage in automating the generation of test code from the test case data. There will be a significant reduction of test code maintenance effort if such dream comes true. If it is too complex to automate it, try to semi-automate it, if it is still a complex job to do try to minimize the test cases using a weaker coverage criterion and develop the test code manually.

6: Test code execution:

Test case execution is the process of running its equivalent code then checking and reporting the execution result. A number of frameworks exist at all testing levels. For example, the xUnit is a family of unit testing frameworks which allow automatic unintended execution and reporting of test benches. The xUnit family supports a number of languages for example:

- JUnit for Java.
- CppUnit forC++.
- PUnit for Python.
- NUnit for .Net Platform.

Refernces:

- [BEZ1], Beizer, B., "Black Box Testing"
- [BEZ2], 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.

Email: omkamal@yahoo.com

Metrics-Based Process Improvement Series: Let Us Discover another Buzzword "*Six Sigma*"

By: **Ahmed S. El-Shikh**

In two previous articles, I had conducted a discussion about some of the **numerical aspects** in the software production process; I started with an article about the **measurement** aspects in the software and focused on the Goal-Question-Metrics, **GQM**^{*} approach and its modified version that called **GQ(I)M**[†] [1]. In the second article, I focused on how to use these collected data to describe the nature of the process capability and limits by applying the **statistical process control "SPC"**, also the relation between the statistical process control and **CMMI level4** process areas and practices [2]. Usually, when we start to talk about the numerical aspects in software, the mind will go to some aspects like the number of defects, time estimation or even the clearest example in software sizing which is the count of lines of code (LOC). This is not the whole story; there are a lot of numerical aspects in the software that can be in the spot.

If the God will, I will conduct a **series** of articles that focus on the **numerical aspects in the software engineering and metrics-based software process improvement** because they provide a solid and qualitative approach to evaluate the success degree of the process improvement efforts. In this article I

* Goal Question Metrics (GQM) is a measurement approach that binds the measurement activities to the business goals.

† Goal Question (Indicator) Metrics, GQ(I)M is a modified version of the GQM that introduced the concept of the indicator.

will conduct a step further in measuring and analysis build on my two previous articles.

As we know, according to the organization goals or due to a certain quality model requirements, the **measuring activities** can focus on measuring some attributes either in the software *product* or the *process*. Number of defects per module or components, line of codes (LOC) and reliability are examples of what you can measure in the product itself. Productivity (in LOC/hour as an example), estimation accuracy and the yield of the defect removal process are all examples of what you can measure in the production process.

After measuring, you can **analyze** what you had collected to discover your **process capability** through defining the control limits. When your limits are clearly defined, this means that your process voice is known. The voice of the process (**VOP**) is what your process can do for you, but the question here is what your customer needs. In other words, some time you can find yourself in the case of a difference between your customer's voice and your process's voice. Whatever your customer is, I mean internal or external one, you will need to try to minimize the gap between the two voices. In more technical terminology, you will need to conduct an **improvement project**.

The difference between your customer voice and your process voice can appear in different cases, see the following figures for illustration.

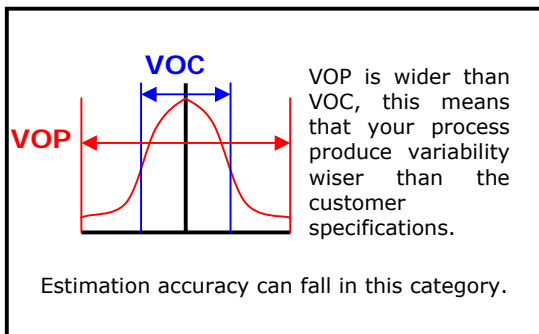


Fig-1: VOP is wider than VOC

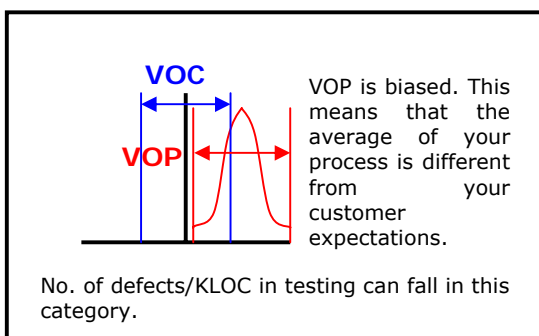


Fig-2: VOP is biased from VOC

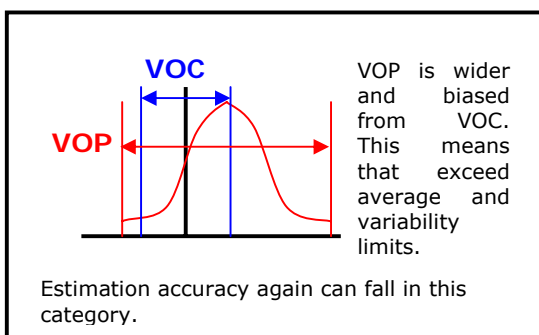


Fig-3: VOP is wider& biased from VOC.

The above three figures show three cases in which you can find your process (or product) different from your customer specification. The three figures show that your process behave normally (i.e. the output is normally distributed), even if your process output is not normally distributed, it can be transferred into normal by employing "Central Limit Theorem"*.

* The Central Limit theory is beyond the scope of this executive summary article. It

There are **several models** to improve the process. Some of them are based on logical steps to collect the best practices from your company. **IDEAL** approach is a clear example of this kind. **CMMI** itself is a model for process improvement that collected a lot of the best practices from a lot of companies and arranged them in goals and then in a higher level called process areas. This type of models can help the organization to make a shift in the whole maturity level of and day-by-day activates and practices. The question here is what can help in the case that number of defects per module is not satisfied for your senior management, the quality department may want to increase the yield of the defect removal process. Finally, you may want to improve your estimation accuracy and your team productivity. In all of the previous cases, you want to move from a level to level, a **limit to limit**, and more precisely from a number to number. In these cases, *Six Sigma* improvement methodology can be used as a **numerical-based process improvement framework**. Let us spend some time to discover what this word "**Six Sigma**" means.

First of all, let us introduce a **formal definition** for the "Six Sigma" methodology. Just be careful of the terminology, the definition that will be stated here is for the methodology itself not for the word "Sigma" or the number "Six". "Six Sigma" is: A comprehensive and flexible **system** for achieving, sustaining and maximizing business success. It is uniquely driven by close understanding of customer needs, disciplined use of facts, data, and statistical analysis, and diligent attention to managing, improving, and reinventing business processes [3]. The previous definition describes the Six Sigma as a *system*. This means

grantees the ability to use the normal distribution even if data is not normally distributed by using samples averages.

that there are inputs, processing and outputs. As we see, this definition is a highly technical wise. In more simple words, Six Sigma is: A high performance, **data-driven approach** to analyze the root causes of the business problems and solving them. It ties the output of the business process directly to the market place [4]. As I think, the definitions still talk about systems, data and root causes. In the case that you want to get an agreement from your senior manager to let your quality team conduct a new methodology, just use this definition: Six Sigma is a methodology to **align the company to its marketplace** and deliver more improvement (which means dollars) to the bottom line [4]. Finally, for your operational managers and project managers, tell them that Six Sigma is a methodology to (1) **dramatically reduce the process variation** (i.e. estimation error). (2) And move your product specification beyond customer specification [4].

The **history of Six Sigma** started in 1980's in Motorola Company. Motorola started to produce Six Sigma methodology to help it facing the increasing competition in the market. It started as a systematic approach to track the compatibility between the customer requirement and the production abilities and went further to finish with a formal definition of 3.4 defects per million opportunities, in more technical terminology, it tends to make the voice of the customer (VOC) to be the double of your voice of the process (VOP) or process capability coefficient (C_p) equal to 2 [2].

Six Sigma had been built on a **basic concept** that considers that "every thing is a process and all processes inherit variation". So, time estimation is a process, coding is a process, even employees' arrival to the company and employees' attendance are processes. Regarding to variations, SIX SIGMA is designed to deal with the natural

variations and common causes of variation inside the process. The assignable cases do not need improvement project, only corrective action can solve them.

Since we talked about the natural variations in the process, a formal **definition for the word "SIGMA"** can be given. The word "SIGMA" is a Latin symbol, which is used in statistics to identify the variation in the data and represent standard deviation.

From the **statistical point of view**, the word "SIX SIGMA" means that *your process has that ability to fit six times of its standard deviation inside the customer specification at each side of the average*. In other words, the voice of the process (VOP) is always equal to 3-sigma above and below the average. This means that your process produces 3 defects per one thousand opportunities because 97% of the values of the analyzed attribute are inside the control limits. So if the customer voice or specifications are equal to the process specification (VOC=VOP) this means that these 3 defects will go to the customer side from each 1000 opportunities. If you can adjust your process to have Six Sigma below and above the average to the customer specification (VOC= 2 VOP), by reducing the value of sigma, you can grantee to have only 3.4 defects for each million opportunities at the customer side.

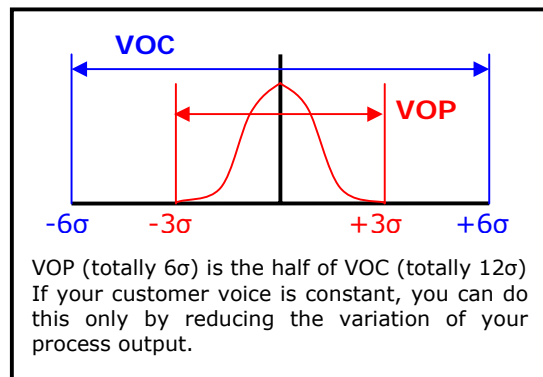


Fig-4: VOP to VOC in Six Sigma.

From the **mathematical point of view**, SIX SIGMA tends to let your process attribute meets customer specification. Using SIX SIGMA terminology, SIX SIGMA tends to let the process's symptom (Y) meets and exceeds the customer specification. The symptom (Y) depends on several root causes [$Y = F(X_1, X_2, X_3 \dots X_n)$].

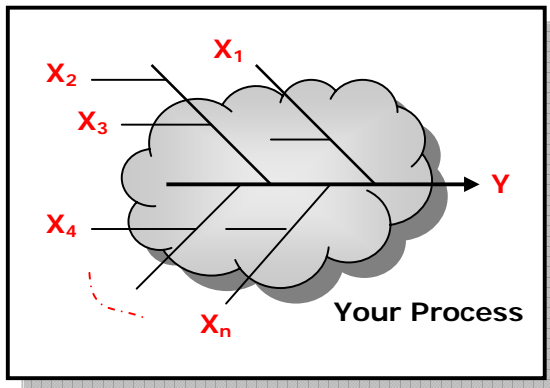


Fig-5: Root causes of the symptom under improvement in fish bone hierarchy.

The fish bone diagram is used as a tool to discover the most eligible set of the root causes, but be careful that this set does not contain the complete set of the root causes. It just contains the most eligible ones. It is only an approximate function for the symptom. The next step is to try to identify the most effective root cause in the equation. In other words, the root cause (X_i) that significantly changes the value of the symptom (Y) when its value changes. This is done by applying the scatter diagram. The scatter diagram is a simple way that can be used to identify the degree of the correlation between two sets of data. The more professional method is to use the regression analysis to do that.

There is a **well-defined framework** to conduct the required steps in that mathematical analysis, which starts from choosing a project for improvement, to collecting the

required data to identify the root cause and finally solve the problem and maintain the result. This framework is called **DMAIC approach**, which stands for the initial letter of words: **Define-Measure-Analysis-Improve and Control**. The figure 6 shows the normal sequence of the DMAIC. Note that each phase is done once.



Fig-6: The DMAIC sequence.

These 5 steps are done in this defined sequence and are stand for the basic and essential steps in the improvement process. Some opinions expand these 5-steps to be 8-steps. They are **RDMAICSI**. These letters stands for the initial letter of words: **Recognize-Define-Measure-Analysis-Improve-Control-Standardized and Integrate**.



Fig-7: The RDMAICSI sequence.

Generally speaking all of these five or eight steps fall into **four categories**:

- **Identification.**
- **Characterization.**
- **Optimization.**
- **Institutionalization.**

Identification category contains *recognize* and *define* phases. *Measure* and *analysis* fall into characterization phase. *Improve* and *control* phases cover the optimization category. Finally, *standardized* and *integrate* finish the cycle by forming the institutionalization category. By investigating these categories you can discover that they represent the **logical cycle for problem solving**. To solve any problem, you need to

identify it, then describe it and fully understand its characteristics. Characterizing your problem will enable you to deal with and guide you to find the required solution. Last step grants that you can utilize the solution in similar problems. Finally, integrate it with other parts of your system. Let us walk through these phases to take more details:

- **Recognize:** is the first step in which the company can discover the importance of numerical techniques for process improvement, the concept of Six Sigma and the types of problems that the Six Sigma fits for.
- **Define:** define phase starts by nominating several projects which are eligible for improvement, then evaluating these projects according to some selection criteria to select the most important project. Finally, do not escape from this phase until conducting a verification process. It aims to ensure that this project is an improvement project not a planning one*.
- **Measure:** the measure phase is the most important phase to characterize the problem. The symptom (Y) is defined in this phase and takes a numerical value that represents the improvement project baseline. You do that after defining the boundaries of the problem. The problem here is the process or the part of the process under

* Quality planning is one of the main branches of the quality triangle and is fulfilled in SIX SIGMA by using the Design for Six Sigma, DFSS methodology

the improvement activity. Some of the quality tools are essential in this phase, such as: **Flow diagram, Pareto diagram, Function Deployment Matrix (FDM) and Failure Mode and Effect Analysis (FMEA)**[†].

- **Analysis:** the analysis phase tends to collect a set of all possible causes (i.e. $X_1, X_2, X_3, \dots, X_n$) which affect the symptom (Y). After formalizing this set, the method focuses on utilizing some of the quality tools to identify the root cause(s). **Brainstorming, Cause-effect (Fish bone) diagram, data sheets, Histogram and Scatter diagrams** are a set of useful quality tools that can be used in this phase.
- **Improve:** you are ready to start the improvement activity in this phase after you get the objective evidence on the root cause(s) of the problem. All possible solutions are listed and selection criteria are defined to help in choosing the most suitable solution. The only quality tool that can be used in this phase is the **improvement selection matrix**.
- **Control:** just the most suitable solution is selected, you need to define some of controls that enable to maintain the solution and measure the actual (Y_a). The **actual** (Y_a) is then compared to the **desired** (Y_d). if there is a difference, you can

[†] Some of these tools are from the Ishikawa's seven basic tools; others are advanced quality tools such as the FDM and FMEA. The explanation of these tools is out of the scope of this executive summary article.

conduct a second cycle of improvement until you meet the desired (Y_d). You can use control spread sheets or even control charts in this phase.

- **Standardized:** this phase grants that the solution can be applied in day-by-day activities through the organization.
- **Integrate:** finally, in this phase you want to be sure that these day-by-day activities can integrate with the overall legacy parts of your process.

Improvement **project team** is a remaining important point to conduct. SIX SIGMA's DMAIC methodology suggests a team with a recommended structure and skills. The team should contain the following roles besides the **customer**, which is the one who will provide a lot of valuable information and can get benefits from the improvement effort:

- **Champion.**
- **Master Black Belt.**
- **Black Belt.**
- **Green Belt.**
- **Orange belt.**

The **champion** plays the role of leading the initialization and deployment of the Six Sigma approach into the organization.

The **Master Black Belt** is an expert selected by the champion to be in-house to provide training and coaching the reminder team members. He is a full time expert and gives 100% of his effort for Six Sigma.

The **Black belt** is the dedicated team leader for the Six Sigma improvement team. He trains the green and orange belt and leads the team through the DMAIC phases.

The **Green Belt** is the one who executes the project and do the required steps and using the required quality tools to achieve the desired improvement result. He can be a full or part time; this depends on the team size, project size, project duration and a lot of other aspects that affect the deadline of the project.

The **Orange Belt** is usually a team member that gets a fast training on the Six Sigma approach to gain the essential awareness, and to be able to share in the different phases in the project. This employee is very important in the improvement project even with being the lowest level of knowledge about the methodology. He is the most experienced team member with the process under the improvement, so he represents the information bank for the team.

SIX SIGMA methodology still contain a lot of aspects that need more and more articles to be explained. If the God will, I will complete the series of the numerical aspects in the software process with some articles about:

- The relation between the CMMI and Six Sigma methodology and how you can integrate both of them inside your company.
- Integrating Six Sigma with PSP and TSP to improve your productivity.
- Integration between Six Sigma and the agile software engineering methodologies such as extreme programming (XP).

This is just a proposal for the series. According to the feedback comments, I can change the sequence and focus on giving detailed case studies and applying tutorials.

References

[1] Ahmed S. El-Shikh, "*The Blind Manager, Can he drive through the competition? Software Measurements*", article in the 6th issue of the Egypt SPIN newsletter, 2004.

[2] Ahmed S. El-Shikh, "*Discover your capability and let numbers play their rule, Statistical Process Control*", article in the 7th issue of the Egypt SPIN newsletter, 2004.

[3] "*The Six Sigma Way*", book for Peter S. Pande, Robert P. Neuman And Roland R. Cavanagh, 2000.

[4] Dan Burton and Don MacAndrews, "*Enable statistical Process Control and Six Sigma for Software*". SEI presentation, August, 2000.

Biography

Ahmed S. El-Shikh, the Quality Assurance Manager in Systems Advisory & Solutions, (SAS). He holds a bachelor's degree in the control and computer engineering, 1998 and a Diploma in the "Total Quality Management" from the AUC, 2004. His interests include software engineering aspects, software quality management, statistical quality control and process Improvement approaches specially the "Six Sigma, DMAIC, DAMDV & DFSS".

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:

el_shikh@sas-sys.com