Egypt–SPIN Newsletter

Issue 10, Apr. – Jun., 2005

Sponsored by SECC

From the Editor (Ahmed S. El-Shikh)

Welcome to our 10th 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.

> 5 to 9 June, 2005, the **"Intermediate Concepts of CMMI"** course had been conducted in Egypt and is sponsored by **SECC** with collaboration with **USAID/ICT**. See the next page for more details.

This issue conducts some hot topics within three series and two stand-alone articles. Explanation of one of CMMI process area (1st article), discussion of the software industry in Egypt (2nd article), sharing real life experience in the field of software testing (3rd article), summarizing the content of a negotiation course (4th article) and introduction for Personal Software Process, PSP (5th article).

Eng. Sameh Zied *starts a* ***series for explaining the CMMI*** *version 1.1 process areas as presented in the "****Intermediate Concepts of CMMI****" course. The article describes the use of* ***Decision Analysis and Resolution (DAR)*** *process area in a software development organization.*

Dr. Ramiz Kameel *starts a* ***series to discuss*** *the* ***nature of the Egyptian software industry****. His article defines and categorizes the* ***current patterns of software production process*** *in the Egyptian community, also defines different roles in it.*

Eng. Omar Kamal *continues his* ***series about software testing****. His article introduces the* ***xUnit Testing Framework*** *as one of the most popular testing frameworks. He defines its main feature and explores its known extensions.*

Eng. Ahmed Abd El Aziz *Shares his understanding from a* ***negotiation methods*** *course that he had attend with the SECC. His article describes how to map and use these negotiation techniques in day-by-day activities in the field of project management.*

Eng. Ahmed hammad *introduces the* ***Personal Software Process (PSP)****. He summarizes his understanding and real life experience with the PSP from the official SEI training course. He shows how the PSP can help the single programmer building high quality software products.*

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 jaselshikh@yahoo.com

"Intermediate Concepts of CMMI" course in Egypt.

By: The Editor

With participation from 20 IT specialists in Egypt, **Software Engineering Competence Center (SECC)** in collaboration with **USAID/ICT** program had conducted a training course for the **"Intermediate Concepts of CMMI"** within the period of 5-9 June 2005 for the first time in Egypt.

The "Intermediate Concepts of CMMI" training course is delivered by the **Software Engineering Institute (SEI)** and instructed by **Chuck R. Myers** and **Richard E. Barbour**, Visiting Scientists at the (SEI). The course is delivered by two instructors because of its interactive nature with the participants.

The course is designed to different types of **audience**, such as: (1) Candidate lead appraisers for the SCAMPI Appraisal Method. (2) Systems and software engineers, SEPG, EPG process personnel who need more in-depth knowledge of CMMI models. (3) Candidate instructors interested in teaching the Introduction to CMMI.

The course has several **objectives** including help participants:

1. Establish links from their past model use and experiences to CMMI models.
2. Understand the relationships among model components, including both staged and continuous representations.
3. Understand how to interpret and apply CMMI models effectively.
4. Share, learn, and exchange ideas with other participants about practical implementation of each process area.

For successful participation in the course, SEI states a set of the **prerequisites** that have to be fulfilled before participation, including:

1. Complete a formal SEI authorized training of the "Introduction to CMMI" course.
2. Obtain experience with using the model before applying for the Intermediate class.
3. Carefully study the full content of one representation of the CMMI SE/SW/IPPD/SS model, Version 1.1, and Carefully study chapters 1-5 of the other representation of the CMMI, or the CMMI book (Guidelines for Process Integration and Product Improvement), by Chrissis, et. al., published by Addison Wesley.
4. Complete and submit a pre-class assignment.
5. Capability of communicating well in English.

This five-day course is a **prerequisite** for **"SCAMPI Lead Appraiser"** Training and **"CMMI Instructor"** Training. It introduces participants to detailed CMMI concepts, including the relationships among CMMI model components.

The course **contains** lectures, participants' presentations and class exercises and is presented in a facilitative style designed to create dialog among participants and instructors.

The following **topics** had been conducted in detailed:

- CMMI Product Suite.
- CMMI model representations.
- CMMI model components.
- Equivalent staging.
- Engineering process areas
- Project management process areas.
- Process management process areas.
- Support process areas.
- Quantitative management process areas.
- Acquisition process areas.
- IPPD process areas.
- Optimizing process areas.
- Overview of the SCAMPI appraisal method.

Course evaluation is based on three parts: (1) Pre-class assignment presentation delivery. (2) Interactive class participation in the discussion, and (3) Closed-book model knowledge exam that had been conducted at the start of day 5 of the course. **Be careful**, if you plan to attend this course in its second run; try to get well preparation to the closed book exam before you go to the course. Four days of interactive participation is very hard to be compound with full night study to pass the exam. A certificate of successful completion will be delivered to the participant if he successfully fulfills the first and second evaluation parts plus a score equal to or higher than 80% in the closed-book exam.

*To share participants' experiences and maximize the benefits from this course, the newsletter will conduct an **explanation series** contains an article for each process area. Each participant –who had already attend the Intermediate Concepts of CMMI course- is welcome to write an article about his assigned process area to be published in the coming issues of SPIN newsletter according to his/her time constrains. We would like to thank **Eng: Sameh Zied** for his valuable suggestion that had been recognized by most of course participants and lead us to start this series.*

By conducting this training course, the quality improvement efforts sponsored by SECC in Egypt has been focused on individual's skills improvements beside the over all company maturity level improvement. Hope that this course can trigger the appearance of significant number of SCAMPI lead Appraisers and CMMI Instructors to provide a good leverage for the software industry in Egypt.

Table of Contents

CMMI Process Areas Explanation Series: Decision Analysis and Resolution in Software Development

By: Sameh Zied

***Abstract* — This article describes the use of Decision Analysis and Resolution (DAR) process area in Software Development organizations. DAR is a Supporting Process at maturity level 3 of CMMI version 1.1 model. DAR is perceived to provide key advantage in certain decisions of importance to software development and acquisition. DAR is pervasive and may be applied informally to make daily project decisions.**

***Keywords* — CMMI, COTS, DAR, GG, GP, EPG, PA, SG, SP**

I. INTRODUCTION

THE purpose of this article is examine the use of formal decision process (DAR) for certain decisions that are related to Software Development and Acquisition. Decisions related to selection of Design have always been of concern. They have snow ball impact on the development, maintenance and phase-out cycles of the product.

Process Improvement programs involve many groups, for example, Engineering Process Group (EPG), Engineering, Testers, Project Managers, and others. Much of group work involves Decision Making. Being a Facilitator, EPG meets various teams to resolve issues and to arrive at decisions. Applying a formal decision process helps to arrive at higher quality "non-subjective" solutions and decisions. It also helps making teams work better together, by focusing at same time on same step.

II. BENEFITS OF DAR

DAR Cascades top management expectation for making "objective" decisions at certain situations. Making better decision could lead to:

- Reduction in rework, and
- Buy-in from more stakeholders.

The following figure shows a conceptual understanding of how DAR can serve to realize management objectives and vision.

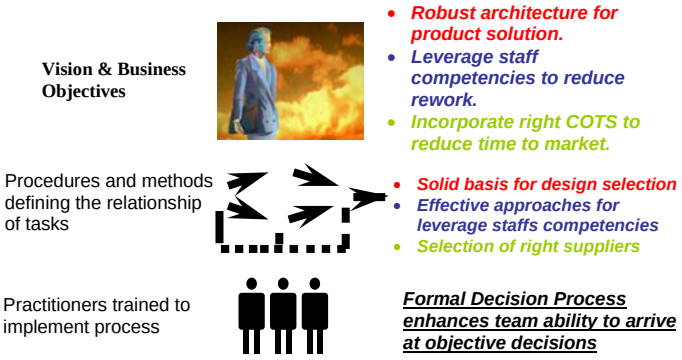


Figure-1: DAR linkage to Organization Vision

III. WHEN DAR IS REQUIRED

DAR can be invoked during the execution of any process when an issue is encountered, as shown in the next figure.

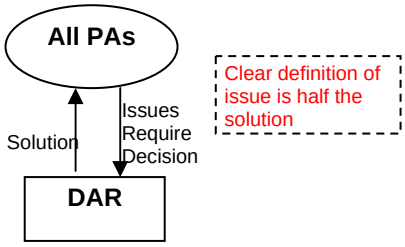


Figure-2: DAR is invoked from anywhere

DAR can only produce reliable results when the issue is clearly defined and communicated well among participants. It is to be noted that a key output of DAR is not only a solution but also the associated *risks*.

These risks can be reason for selecting an alternative solution, even with lower score.
Though a formal decision making process may be used from everywhere, the following sub-sections describe the key decisions that typically require a formal decision.

A. Technical Solution

A design solution is a robust to the extent of alternative designs are considered and trade-off is made. Design decisions should be made based on formal process to have documented basis for any future justification. Design Decision leads to the generation of another level of requirements and affects the scenarios of using the ultimate product.

B. Supplier Selection

Acquisition of Commercial Of The Shelf (COTS) products is normally part of software projects. Software licenses, supporting tools, and other products are key ingredients for most software projects. Selection of right products should be based on criteria that support project objectives and address its constraints.

C. Training Approach

Team development is key objectives for organizations to leverage the competencies of their staffs to do activities that are inline with objectives of the organization. Training approach can be self-study, formal class-room, on-job training or workshops. The selection of an approach will impact acquired competencies and ability of staffs to do their tasks.

D. Corrective Action

During Monitoring and Control of project activities, the project manager use measures to assess project status against plan. Deviations are addressed by corrective actions. When deviations exceed certain threshold, corrective action should be decided based on formal process. The Project Plan should have reference to the decisions that will require corrective action based on a formal process.

IV. DESCRIPTION OF DAR

The following diagram shows the specific practices `of a formal decision process based on the CMMI® DAR process area.

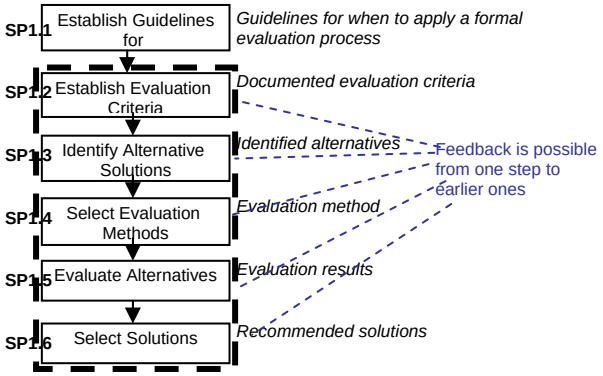


Figure-3: Specific Practices of DAR

The text inside each box describes process step, while the text to the right is the output from performing this step. Of critical importance is establishing guidelines for practitioners describing the decisions that require the use of DAR.

Every step of the process can have feedback to previous steps. This feedback leads to refinement of outputs from previous steps, especially evaluation criteria. The need to refine output of certain step only becomes evident upon executing a following process step.

Every step in previous figure corresponds to a Specific Practice (SP) of the DAR process area in CMMI®. These are explained in the following

paragraphs.

A. Establish Evaluation Criteria

Evaluation Criteria is traceable to a documented source and should ensure buy-in of relevant stakeholders. The rationale of selecting criteria should be documented. Recommendation of choosing evaluation criteria:

1. Simple criteria having most critical components and agreed-on by relevant stakeholders.
2. Criteria should be part of process assets of the organization and continuously improved.
3. Input from expert should be sought.
4. Weights should be aligned to business objectives.
5. Organization Process Assets and historical data should be used.

B. Identify Alternative Solutions

The driver for identifying alternatives is the Evaluation Criteria. When considering alternative solutions, it is recommended to consider the following:

1. Critical few alternatives are identified.
2. Techniques for alternatives generation (e.g. Brainstorming) can be used.
3. Organization Process Assets should be used.
4. Bias in favoring certain alternative should be controlled.
5. Proposed alternatives are documented.
6. Might need to revisit criteria based on alternative solutions.

C. Select Evaluation Methods

Practically REVIEW is the most commonly used Evaluation Method; however, selection of a method depends on the degree of clarity in defining the issue and availability of historical data. Examples of other evaluation methods are Simulation, Surveys, Cost studies, Engineering Studies and Testing. Level of detail of the method used should be assessed regarding its impact on cost, schedule, and risks. Combination of Evaluation Methods can be used

D. Evaluate Alternatives

Evaluation Criteria may be revisited based on analysis of results and discussions. Simulation, pilots, prototypes and modeling can be used to analyze evaluation criteria, methods and alternative solutions. Alternative solutions may be revisited and new ones are evolved. Evaluation Team should agree on score to every component of the criteria.

The evaluation process should be repeated till alternatives test well. The evaluation results should be documented so that to have basis of the decision

E. Select Solutions

Highest score alternative may not be always selected. Final solution is selected based on *Risks Assessment*. Reason of selecting or rejecting solutions should be documented and become part of Process Assets.

V. CONSIDERATIONS WHEN USING DAR

To get the desired benefits from using DAR, it is recommended to:

1. Define the issue clearly and early.

2. Involve right people.
   Right people involved in the process will generate valid key alternatives. This will lead to a solution that is in-line with organization business objectives and project needs.
3. Make organizational-wide guidelines.
4. Have a defined process for DAR.
5. Plan for DAR in the project plan.
6. Monitor Cost-Benefit of using DAR.
   Cost of using DAR in a project should not exceed certain threshold, which is normally a percentage of the project budget.
7. Enhance Evaluation Criteria.
   Evaluation criteria should be improved from an instance if using DAR to next one for a similar situation. For example, deciding on technical solution, can contribute to revise evaluation criteria from an instance of implementation to another.
8. Use Organizational Process Assets.
   This includes making analogy with previous similar situations. Then utilize applicable:

   - Evaluation Criteria
   - Alternative solutions
   - Basis of deciding on Selected Solution

VI.PERVASIVENESS OF DAR

DAR addresses *formal* decision making, however, the project manager always needs to make decisions in the course of all of project activities. The project manager may apply DAR paradigm to decide whether a project task has been completed or not. Completion of any task requires DAR type of thinking. A decision of completion of any task is bound by review and approval of its deliverables.

For example, the following figure shows how DAR may be used *informally* to decide the completion of a certain project task.

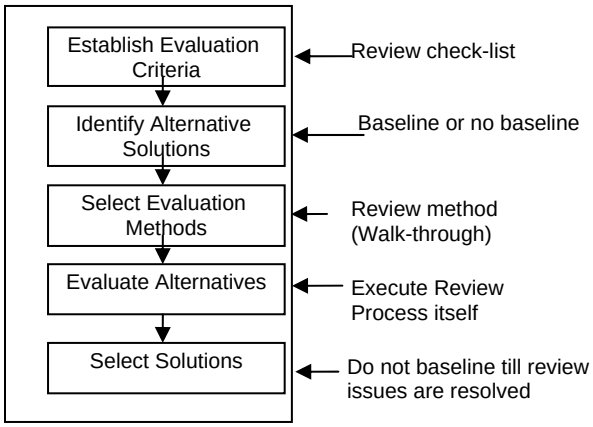


Figure-4: Applying DAR informally

The work products involved in a project task are equivalent to the outputs produced from the specific practices of DAR. This is high-lighted in the previous diagram. Normally, there is a review check list (act as evaluation criteria) to decide whether a task is completed or not. Base lining or not of a work product acts as alternative solutions.

VII. Conclusion

Decision Making is a management practice. Software development projects need to incorporate this practice to reduce subjectivity, and improve team work and overall quality. DAR is a formal decision making process that may be required from many process areas; however, certain situations are common in using DAR. These situations include selection of Design Solution and making Corrective Action due to unacceptable deviation.

Project Managers should apply DAR informally in their daily activities to

make better and more consistent decisions.

VIII. ACRONYMS

CMMI	Capability Maturity Model Integration
COTS	Commercial Of The Shelf
DAR	Decision Analysis and Resolution
EPG	Engineering Process Group
GG	Generic Goal
GP	Generic Practice
OPAL	Organization Process Asset Library
PA	Process Area
SG	Specific Goal
SP	Specific Practice

References:

[1] CMMI, Guidelines for Process Integration and Product Improvement. Mary Beth Chrissis, Mike Konrad and Sandy Shrum.

Biography:

Sameh S. Zeid is with ITSoft, an organization for providing off-shore software services. Sameh manages Process Improvement program that will lead to higher maturity levels. He has been involved with software development and management since 1985. Sameh played various roles on the technical and managerial levels for establishing core business solutions. In last one year Sameh has managed Process Improvement Program that accredited the organization CMM level-3. Sameh has a B.Sc. degree in Computer Science from Faculty of Engineering and Post-Graduate degree in Operations Research.

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Email:
szeid@itsoft.com.eg

Toward Egyptian Software Industry Series: Egyptian Software Production Community

By: Ramiz Kameel

OBJECTIVE

This article "Egyptian Software Production Community" is the first article of a series of articles "Toward Egyptian Software Industry" that concerns with the software industry improvement in Egypt. The present series of articles will not concern with infrastructure that required from community to push the software industry, such as; competition culture, basic educational system, or governmental efforts for supporting [1]. On the contrary, this series of articles will concern with the internal improvement of the software cycle of production. The present article will define and categorize the current patterns of software production processes in the Egyptian community, and the different roles in it. Ideally, the scope defining will be the aim of this article.

INTRODUCTION

Any software production community has three main objects that interact together for producing software; actors (or roles), processes, and patterns of interactions. Actor plays the driving role in the software production community. In most cases, one actor plays more than one driving role in same time, for that the "Actor" will be replaced by "Role". Second object, process is the mechanism that is followed in software production. These mechanisms have different natures upon the community characteristics and its relevant culture. Pattern, or interaction pattern, is the description of roles' definition and their relevant process or processes. In the next sections, an adequate description of the roles, processes, and patterns in the Egyptian Software Production Community (ESPC) will be presented.

ESPC ROLES

In this section, roles' definitions are presented based on the ESPC. These definitions belongs roles only regardless the definition of role actor.

VENDOR ROLE

This role is responsible to receive (not gather) the software characteristic, functional feature, and non-functional features. This role is, also, responsible to design software structure, develop, and implement it. Sometimes, the vendor role is responsible to perform and follow-up the pre-performed marketing and sales plans.

CUSTOMER ROLE

This role is responsible to use the software that produced to perform the same manual or semi-automated operations that are done. Customer role leads the software project toward building libraries of defined functional operations to feed the first level of outputs.

BENEFICIARY ROLE

This role, also, is responsible to use the produced software. But, this role concerns with higher levels of

outputs and the performance of the produced software. So, this role is responsible to lead the software producing process by a part of functional features and non-functional features.

ESPC PROCESSES

This section represents the main three processes in ESPC. Definitely, each process has the special characteristics. In the present section, these characteristics will be explored. No interaction or upgrade relations are bonding the different processes [2].

BUILDING PROCESS

Building process consists of sequence of steps that finally produce the required software. According to present definition, the building process is a process can used for tailored software. The steps of present process can be divided to two categories of steps; constructive steps, and re-constructive/customized/adaptable steps. So, the building process is re-iterative process by that definition.

CREATING PROCESS

Creating process is an invention methodology to produce the required software. In this process, the producer develops the required software according to predefined requirements. In this process, the iterative modifications of the requirements are completely finished in a stage before creating the software, on contrary of the building process that in it the modifications occur in a later stage after building the software itself.

MANUFACTURING PROCESS

Manufacturing process is a complicated synthesis plan of software production. This process follows an existed plan of production. A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [3].

ESPC PATTERNS

This section represents the main three patterns in ESPC. Each pattern will be represented by its dimensions according to the active sharing roles in it.

VENDOR VS. CUSTOMER/BENEFICIARY PATTERN (BINARY PATTERN)

In this pattern, the customer plays the same role of the beneficiary of the software additional to his main role, not the reverse. So, in this pattern, the actor of role (customer/beneficiary) gives the priority to the customer role. Actually, customer role plays the main guidance role during the software production phases. The beneficiary role effect appears after the finishing of the software production. This needs a lot of customization and re-customization.

VENDOR VS. CUSTOMER VS. BENEFICIARY PATTERN (TERNARY PATTERN)

This pattern represents the equivalence effect of each role in the software production process. In this pattern, there is no role that plays the part of another role. Each role shares in the process by its own inputs and expects its own outputs to be satisfied.

VENDOR/ CUSTOMER/BENEFICIARY PATTERN (UNARY PATTERN)

Unary pattern represents the one role in the software production process. In this pattern, the vendor plays the all three roles. This pattern represents the complete cycle of the product line.

Discussion

The Egyptian Software Production Community (ESPC) consists of several members, software companies (vendors), customers, and beneficiaries of software. Those are the main actors in the community. Ideally, for precision, it is preferable to define those members or actors as roles. Significantly, they have an obvious influence on the software life in Egypt. On the other hand, one can neglect the effect of any other role on the software life.

According to the definitions of processes and patterns, one can observe the linear matching between the present patterns and the present processes. The unary pattern utilizes the manufacturing process during the software production cycle, in which the software is produced according pre-defined plan based on adaptable marketing research [2]. The binary pattern utilizes the building process in which the software is built in iterative sequence until the core of software design doesn't sustain any further customization. The trinary pattern utilizes the creating process in which the software is created based on equal influence from the three roles.

Unfortunately, there is no enough statistical data represents the share of each pattern in the ESPC. But obviously, from experience in the ESPC, one can notice that the binary pattern (Vendor vs. Customer/Beneficiary) represents the greatest share in the ESPC. This pattern consumes a lot time in software production process due to its iterative nature. In most cases, the durability of the software is affected due to the repeatable customization processes.

In most cases of ESPC, the vendor gives the priority to his role over both other roles, in the unary pattern. This can referred to the absence of real role of quality and R&D in the software production process [1].

One can appear the absence of the gathering software feature role in the ESPC. In ideal community, the consultancy plays the role of gathering software features (at least defining the software scope) based on the customer requirement and the corresponding benefits.

Conclusion

Initially, the ESPC's stakeholders are invited to prepare a complete data information about the software industry in Egypt. This data information should include the sharing of each pattern in the software industry sector. The binary pattern has to be excluded from the ESPC to improve the efficiency of the software industry in Egypt.

The consultancy role has to be established in the ESPC, to support the beneficiary role beside the customer role. On the other hand, the consultancy role will play the connecting link among the customer, beneficiary, and vendor.

Manufacturing process and creating process should be supported by the stakeholders to be the main processes in software production in Egypt. This will need an establishment of several procedures, methods and activities to support this modification stage in the ESPC.

Modification stage in Egypt can be attained by emphasizing the role of R&D in software industry, establishing the proper Egyptian standards, and following a proper methodology in the software process improvement.

Future Work

As mentioned before this article is the first article in the series of articles "Toward Egyptian Software Industry". The next articles will concern with the established methods to improve the Egyptian software industry. Next articles will concern with Software Process Improvement SPI.

Acknowledgement

The author is grateful to, Eng. Ahmed A. Hday, for his valuable recommendations.

References

[1] Meer Hamza, 2004, How to achieve the expected quality in our software industry, Egypt-Spin Newsletter, SECC, Issue 7, July – Sept. 2004, Pages 9-10.
[2] Ahmed A. Hady, Chief Architecture, Egyptian Software and Systems, Prima Soft, Private Communication, 2005.
[3] Clements, Paul and Northrop, Linda. Software Product Lines: Practices and Patterns. Boston, MA: Addison-Wesley, 2002.

Biography

Ramiz Kameel is SPI Consultant of Egyptian Software and Systems; Prima Soft. Author holds a Ph.D. in Engineering. Author is SPI Consultant of Information Technology Institute - ITI.

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:
rekameel@primasoft.com.eg

Software Testing Techniques Series: XUnit Testing Framework. [Part 1]

By: Omar Kamal

Introduction.

In previous articles the author explored Control Flow Testing (CFT) as one of the testing techniques used for testing at different levels. The technique is used to design test cases which should be followed by test case execution. This article will introduce ***xUnit Testing Framework*** as one of the most popular frameworks explaining its main features.

Unit test frameworks.

Unit test frameworks are software tools to support writing, running, and result-reporting unit tests. In a test driven approach, implementation and test code are developed concurrently in a continuous test-code-test cycle. As figure 1 illustrates, an Object under Testing (OUT) is invoked within a testing environment together with its test driver. Test cases are written in the test driver, and each test case excites the OUT and examines its behavior. The expected behavior is compared with the actual behavior and the result is stored for analysis and tracking purposes. The process continues till all test cases are executed. The test driver is maintained and kept for regression testing.

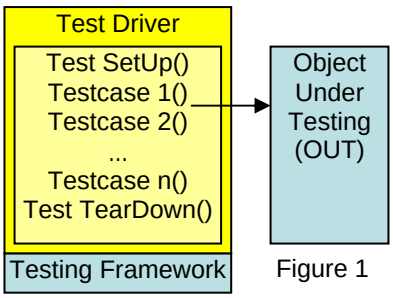


Figure 1

Challenges facing unit test frameworks.

The following are the most important challenges facing the development of any test framework:

1. Compiling and building unit test drivers should be as easy as their corresponding objects under testing.

2. Any software system is composed from various subsystems, which also include a number of objects that may be tightly coupled in terms of interaction. A challenge remains to find a way to isolate the OUT from all other components on the system.

3. The test framework should facilitate isolation of test cases from each other, which is commonly known as *"test decoupling"*?

4. Most of the times individuals responsible for carrying-out the regression test are non-programmers. Accordingly, it would be helpful if the test framework offers a simple and easy way for those individual to run the developed regression suite. The best testing framework will provide "*an un-intended automatic regression execution environment".*

xUnit test framework family.

In 1999 Kent Beck developed a unit test framework for the Smalltalk

language that was simple and easy to integrate with the production code. Later, Erich Gamma ported SUnit to Java, creating JUnit. Next a port for C++ language was developed and was named "CppUnit". Almost every commonly used language has its corresponding unit test framework for example:

- NUnit for .Net C-Sharp
- PyUnit for Python

Every now and then, new ports are developed for more languages based on the same model. These frameworks are known as the xUnit family of tools. All are free, open source software.

How do "xUnit frameworks" face the previously stated challenges?

1. Test drivers are developed as simple classes that inherit, use, or extend some important basic functionalities from the underlying framework. Test drivers are compiled and linked as any other source code file in the system. In doing so, it is easy to develop the source code together with its driver at the same time.

2. The Test Driven Approach (TDD) requires that you follow a test-code-test cycle. For example, a developer may receive a unit description document that specifies a unit to be developed. He/She starts by developing a test case that verify a specific part of the unit he/she intents to code. Next, this initial test case is executed. The test case should fail because the piece of code it tries to verify is not yet implemented. Now, the developer knows that the test code compiles and runs and is ready to verify the code. Then the effort is directed towards implementing the code. After the compilation process is successfully finished, the corresponding test case is executed again to verify the implemented code meets its specification. The test execution result may spot a defect in the code or in the test case itself. The implementation and test code are refined again to make sure bugs are removed from both of them. Finally, a new piece of code is added through this cycle "test-code-test" till the implementation of whole unit is finished with its corresponding test driver.

3. The OUT isolation problem appears when we try to isolate an already developed concrete class from its dependences in the system and develop its corresponding test driver, which may require interface re-work or introducing hooks, etc…. In contrast the TDD approach enforces concurrent development of the implementation and test code which illuminates the isolation problem early in the development cycle.

4. The xUnit framework execution scenario are carried as follows:

   - Test Setup
   - Test $Case_n$
   - Test Teardown

   Test setup is carried out prior to any test case execution and the test teardown is carried out directly after it. xUnit frameworks offer a way to automatically retrieve a test case by test case, while padding it with setup and teardown calls. Enough test

case isolation can be achieved if the OUT instantiation and initialization is carried out in the setup, and the OUT destructing and termination is carried out in the tear down.

5. At the time the developer finishes the unit under development a test driver is also finished and stored under the configuration management platform. xUnit frameworks offers a number of ways to execute those test cases. Test driver execution can be done through command line, Graphical User Interface (GUI), or scheduled scripts.

xUnit framework extensions.

The community effort wasn't only limited to writing ports for more languages but it includes developing add-on tools that extend the functionality of existing unit test frameworks. Example of such extensions is listed here.

1. XMLUnit

An xUnit extension to support XML testing. Versions exist as extensions to both JUnit and NUnit. This is covered in Chapter 10 of this book.

2. JUnitPerf

A JUnit extension that supports writing code performance and scalability tests. It is written in and used with Java.

3. Cactus

A JUnit extension for unit testing server-side code such as servlets, JSPs, or EJBs. It is written in and used with Java.

4. JFCUnit

A JUnit extension that supports writing GUI tests for Java Swing applications. It is written in and used with Java.

5. NUnitForms

An NUnit extension that supports GUI tests of Windows Forms applications. It is written in C# and can be used with any .NET language.

6. HTMLUnit

An extension to JUnit that tests web-based applications. It simulates a web browser, and is oriented towards writing tests that deal with HTML pages.

7. HTTPUnit

Another JUnit extension that tests web-based applications. It is oriented towards writing tests that deal with HTTP request and response objects.

8. Jester

A helpful extension to JUnit that automatically finds and reports code that is not covered by unit tests. Versions exist for Python (Pester) and NUnit (Nester). Many other code coverage tools with similar functionality exist.

The upcoming article

Next article (Insha'Allah) will examine the xUnit articture in more details. The article will explain how to write simple test cases and to execute them.

References

[1] Paul Hamill, "Unit Test Frameworks ", 1st Edition, O'Reilly, 2004.

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

Negotiations and Project Management Real Experience

By: Ahmed Abd El Aziz

Introduction

One of wonderful training courses I attended with SECC was Negotiation Skills course offered by Dr. Hesham Sadek. During the course and in the following few days I had a difficulty to map what I learned in the course to my work. I thought that it is directed mainly to sales people. Few days later, I found that I need what I learned in the course and I use it in many situations I never thought about during the course.

In this article, I will try to share with you my experience in applying what I learned in that course. I will mention some of the points and tricks we learned and how I applied – or did not apply them – in the real world.

Suggestions to Improve Negotiation Skills

Think … BATNA

BATNA stands for (Best Alternative To a Negotiated Agreement). BATNA has five sequential steps:

1. Preparing and Planning.
2. Definition of Ground Roles.
3. Classification and Justification.
4. Bargaining and Problem Solving.
5. Closure and Implementation.

Details of BATNA is out of scope of this article. But what is interesting here is that the bargaining itself is the fourth step preceded by here preparation steps. We do not just start negotiation. We have homework to do first. For example, if the project manager wants to acquire more resources to his project, he has to prepare for the negotiation with the HR manager. He must not ask for twenty programmers and wait for the HR to assign them. However he has to know first how many programmers may be available then try to negotiate about them. This makes the negotiation shorter and more effective.

Begin with a Positive Overture

Again to the previous example. Let's think about what happens when the project manager insists on getting 20 programmers. Most probably he will end with none. When the HR finds that he can give him only 3 programmers, he prefers not to give him any programmer at all as it seems that it will not benefit.

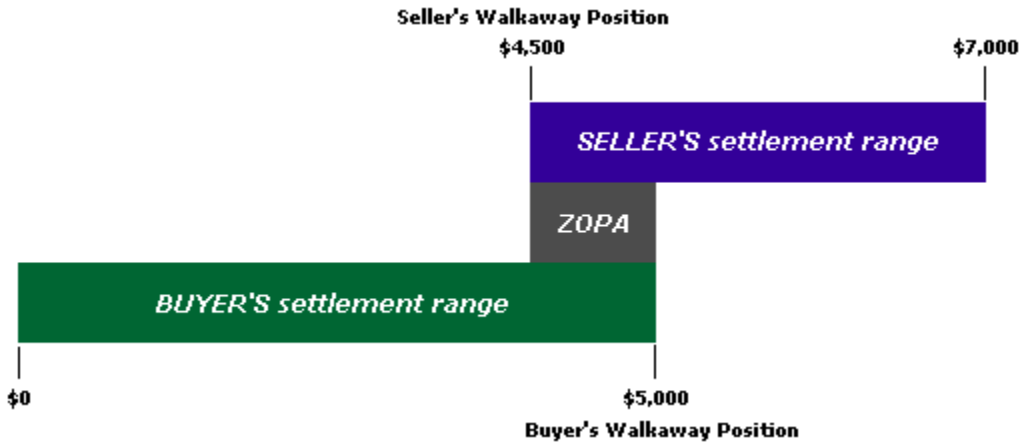
Address Problems, not Personalities

In many situations we mix between the problem and the person we are bargaining with. If I do not like him, I consider him as an enemy, regardless whatever he says. Everything he says is wrong and he aims just to win or gain personal benefits regardless what happens to me. I suspect every word he says. Instead of emphasizing on reaching a solution, I attack him and forget the main problem. This is completely wrong. In any situation I did this, I lost the negotiation. Otherwise I have to deal with others and appreciate that in some situations we agree and in some other situations there is a conflict between us.

Emphasis Win-Win Solution

Before the course I heart this expression many time "Win-Win".

In the course I learned a very important thing that is related to Win-Win. It is called ZOPA (Zone Of Possible Agreement). Let's have a look at the next figure:



We have a seller who wishes to sell his car by $7,000. However if he sells it by $4,500 he is still satisfied. This is represented by the blue area in the figure.

The buyer wants to get for free, like all of us. However he can pay up to $5,000. This is represented by the green area in the figure.

ZOPA is the gray area which lies between $4,500 and $5,000. Any deal within ZOPA is a Win-Win deal. The point is that you have to make sure that the deal is in ZOPA. Do not make it Win-Lose because if the other party loses, expect that there is some trick.

Sometimes I use the ZOPA but with time, not money. In most of the cases I negotiate the delivery date with the customer, and sometimes even with senior management. I know that whatever delivery date I say, he will try to shorten it. So I have to deal with that. Few hours before writing this article, I told my boss that some tasks will be finished after two months. He insisted they must be finished within one week. Finally we agreed to finish after one month!

Create an Open and Trusting Climate

Again, do not deal with the other as an enemy. I have an interesting story in this point. Some day I had a meeting with two guys from the customer's side. One of them is the project manager and the other one is a technical guy. They said they need a FAQ module. I told them that we need about three weeks to develop that module. The technical guy said that it could be done within few days as it is just a simple database application. I did not like what he said and started – in aggressive way – to discuss some of the details in the FAQ module. Finally he agreed that there are a lot of detail and we actually need more time, but I have just started to build a bad climate. The project manger trusts his technical gay at the end. If I do not get this technical guy at my side, I lose at the end. I stopped acting the same way later on. If I want to have him in my side, I start explaining my point by saying "I will do this in the way X because this is the best way as you know". In this way he becomes in my side and in most cases will not resist me.

Must Conclude a Deal

This point I was missing many times. I spend hours in meeting and discussions, then finally we get tired and want to leave without concluding what we agreed. What generally happens is misunderstanding and completely different expectations.

It is strongly recommended that at the end of any meeting, regardless it is for negotiation or something else, to conclude the meeting in very few minutes at te end.

Preparation for Negotiation

Analyze Your Audience

After all we are dealing with humans. We have to analyze them to get what we want. We have to know what are their interests and priorities. We need to know about their culture and behavior.

As an example I was working in a project with a very hard and near dead line. There will be a celebration in the client's country – in the gulf area – and a prince will be in the celebration. The client wants to do – as usual – everything in no time before the celebration. I knew that time is the most important factor for the client at that moment. I used this many times when we were negotiating about the requirements. It is not enough to say that it just takes six weeks to finish. Simply say "Yes I can do it, but do not expect me to finish before the celebration". That was the magic word. Whenever I say it, I get what I want.

Another interesting point I learned in the course. Part of our culture in the Arabic area in general is that we do not like to deal directly. Most of the time we prefer indirect ways. In one of the negotiation meetings about the project I told my boss there are three constraints; time, cost, and quality. Which one has the lowest priority? He said, "All of them are high priority". I was wrong because I went direct. When I went indirect, I said "Ok, we can create the menu the client asked for, but we will ignore the search service at the moment". He said "no problem, we do not need it now". However we both win. He got the feature that is of much importance to him; the menu instead of search; and I made the faster task; I need just half a day to develop the menu while I need two weeks to develop the search service.

How Can You Best Arrange Your Ideas

During preparation for negotiation, write down your ideas in equal manner. Later on review your ideas and try to arrange them. This way you will mwke sure you do not miss anything.

Format for "Yes" and "No" Message

People like who says "Yes" and do not like who says "No". But as a project manager I have to say no in many times. How can I do this without losing my customer or my management? In the course I learnt that when I want to say yes to something, I say it first then I explain why I say yes. When I want to say "No", I say it at the end or even I do not say it explicit at all. As I said in a previous example, I did not say to the client "I will not do this feature", but I said, "This feature takes three weeks to be implemented. It cannot be finished before the celebration. Do you think we have to finish it even if we missed the celebration?" Review my words again. The word "No" does not explicitly exist. But actually I said "No, I will not do it".

However you have to pay attention to saying "Yes". In negotiation, If you say

"Yes" quickly for something, this means you gave it for free for the other party. It is now out of negotiation. Now he will start to negotiate something else. Be careful not to simply leave something to the other party without getting something in advance. Some negotiators negotiate hardly about something they do not want, and in the suitable moment they leave it and ask for the thing they have in their hidden agenda and make the other party feel that he gets the big peace of cake. However they got exactly what they want.

That is not all about negotiation. I even did not go deeply in what negotiation is. I just wanted to show you some of the benefits I got from the course and how I applied them in my career.

Finally I would say thank you to Dr. Hesham Sadek for this interesting course and to SECC for their interest.

Biography

Ahmed Abd El Aziz, has more than eight years experience in the Software Development field. He is now a Project Manager in HARF Information Technology. He is certified as PMP and he has a B.Sc. Of Engineering from Cairo University and a Diploma in Computer Science and Information from Cairo University, Institute of Statistical Studies and Research (ISSR).

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:
aaz@harf.com

PSP, the CMM for Single Programmer

By: Ahmed Hammad

In this short article, I will try to give a simple overview of the PSP as we practice it, what I am describing here reflects my understanding and my practice. If you want formal articles that describe what PSP is, please refer to the references section in this article.

PSP is CMM level 5 applied to a single programmer. In other words, how a single programmer could apply engineering discipline on just himself. I really like this approach, as it comes to basics, the single programmer who is writing source code for the system.

Many small corporations in Egypt have small projects that a single programmer develops from start to end; such projects exist even in medium corporations, especially if they are using modern high level languages, modern development tools, and their large reusable code libraries. For the above reasons, I feel happy with the PSP training, as we are again going to basics and speaking practically; this could help us in our corporation to develop better.

In a nutshell, PSP is a way to develop software using a defined process. The process describes how to collect quantitive metrics regarding performance and quality; and how to find ways to improve continuously.

The magic of PSP formal training is that, we practice it through 10 real programming assignments; some of them are not trivial. So, the PSP training can't be taken by non programmers.

The relation between PSP and CMM

In Fig-1, we show the CMM levels and mark by "*" all key practices that at least partially are covered by PSP. It's clear that a large percent of CMM key practices are covered in PSP.

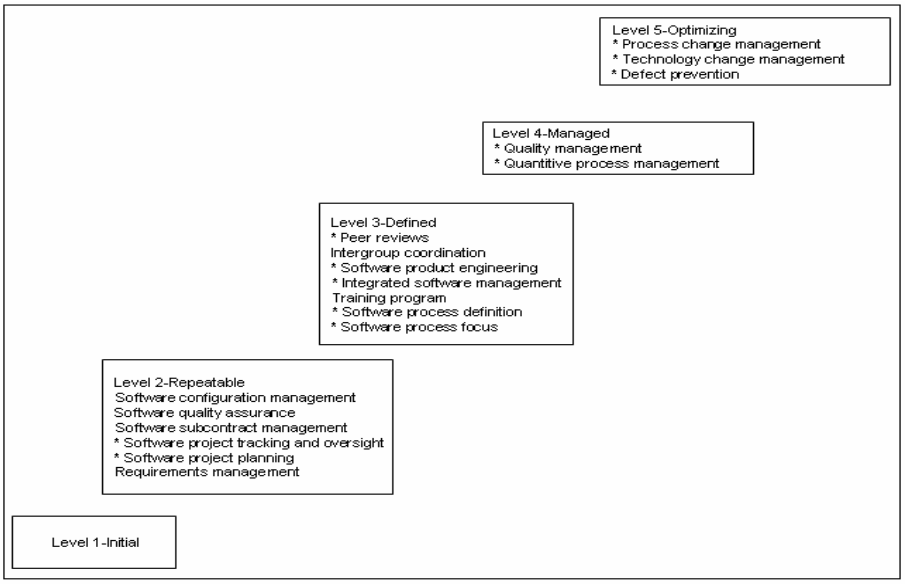


Fig-1 (from "A Discipline for Software Engineering by Watts S. Humphery")

PSP Levels:

As everything in life is developing incrementally, so is the PSP: We started with PSP0, PSP1.0, PSP2.0 and then PSP3. Every new process version introduces new changes in the process. This incremental approach is really natural; I don't expect any success if someone is going to practice PSP2 or PSP3 directly.

In Fig-2, we show the levels of PSP, in each level, new concepts are introduced and a new programming assignment is given to practice the new concepts in actual programming work.

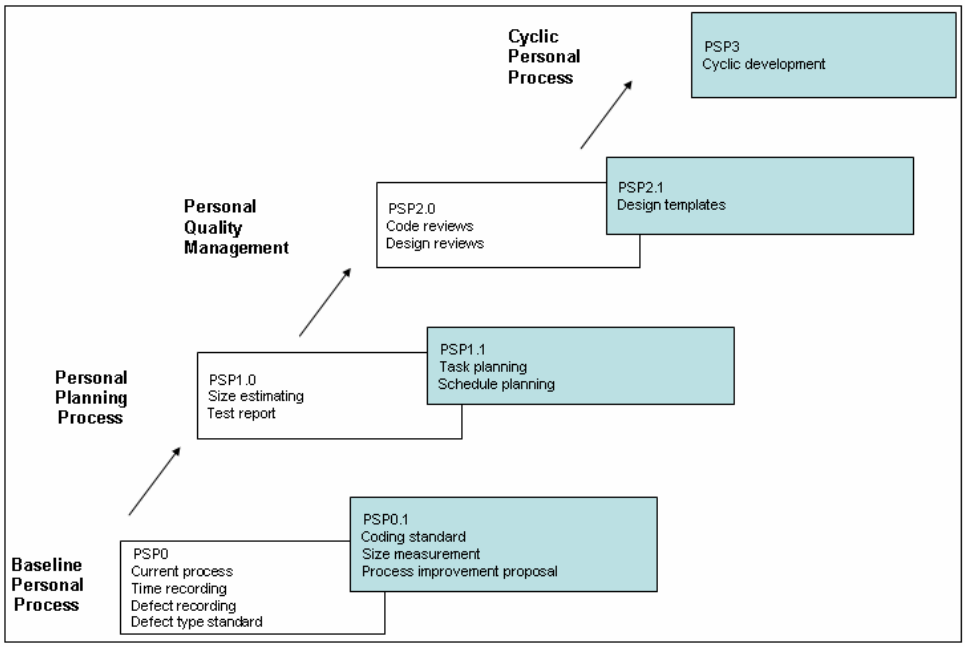


Fig-2 (from "A Discipline for Software Engineering by Watts S. Humphery")

PSP0: The Baseline Process:

The baseline is simply what we already do in developing software, getting requirements, creating design, writing code, and testing; but you would record all the time spent in each phase and would record all the defects you found and how much time it took to fix them.

In the Postmortem report (a report you write at the end of every assignment), you count how much defects are injected and removed from each development phase. Actually PSP0 is the basic training on using a defined process and writing basic reports.

We started the first assignment using paper forms which were tedious but helped us to concentrate on the new concepts rather than to use a tool that could distract us from the new concepts. In later assignments, we used a nice spreadsheet that is provided by the instructor.

In PSP0.1 we start the basis to measure size by following a coding standard. We also used the PIP (Process Improvements Proposal) which is a structured document to

record our process problems and improvements suggestions.

PSP1.0: The Personal Planning Process:

The size estimation OLOC (Object Lines Of Code) is used and compared to the actual. We use a size template to write all objects (Classes in C++ and Java) and how much functions will exist in each object and the expected lines of code per function.

Now with a history you can just estimate OLOC and let the spreadsheet calculate the expected time by using statistical linear regression once there was a correlation [linear relation] between your old LOC and time in previous projects. Of course the estimation will be based on your past data, so you should make sure to record consistent and accurate data to get quality estimates. The spread sheet will also compute automatically the time spent in each development phase based on your past history.

This was a great step, I don't have to estimate time anymore, I will just estimate size using familiar Object Oriented techniques and then estimate the lines of code. The time is automatically calculated using statistical linear regression.

Also in this phase we learn to write test report to record all test cases used to verify the program is correct.

In projects that span many days or weeks, task planning and schedule planning will be a critical issue. First you estimate size and time required (task planning), then make a schedule. The schedule will allocate already estimated tasks to real resources. The schedule tracking is so important, and a corrective action is necessary once a plan slip is detected.

PSP2: Personal Quality Management:

At this step, we already have a very good data about our defects, we analyze defects carefully and devise several approaches to minimize them:

Reviews:

Using structured design review and code review that is based on a checklist, we can greatly reduce defects even before the first compilation and test.
In order to improve the review effectiveness, a peer review is introduced. As we know, we get used of our mistakes, so that we no longer see them. Other minds/eyes will easily detect many of these defects.

Design process:

Through the focus on how to verify the completeness of a design, not necessarily how the design it self is made, verifying the design completeness will eliminate a large source of defects. Actually design completeness check can be done on many phases, like requirements; sure it will be ideal to have requirements completeness criteria before development.

PSP3: A Cyclic Personal Process:

Using PSP2 is perfect in small scale projects, but what if you have a large project? The idea is to divide the project into many PSP2 projects, so each cycle of development is based on high quality previous cycles. If the previous PSP2 cycles are badly done, the test will be a complex task, so we focus on doing PSP2 completely in each cycle.

Tool Support, the Dashboard:

Of course tool support will help us to make following the process easier, I tried to use process dashboard as advised by our instructor Dan Roy, and find it helpful. I encourage every trained PSP to use it, however I think it will be complex and unintelligible to anyone who is not familiar with the PSP through formal training or at least with the help of a formal PSP programmer.

Finally, we started to use PSP in-house, but we still have no concrete experience until now, I hope that anyone who would have that experience to share it with us.

References

[1] The Complete PSP Book: A Discipline for Software Engineering by Watts S. Humphery

[2] The SEI PSP page
http://www.sei.cmu.edu/tsp/psp.html

[3] PSP Body of Knowledge
http://www.sei.cmu.edu/tsp/psp/bok/index.html

Biography

Ahmed Hammad is a project manager in IESCOM, and has a twelve years of experience in software engineering and management.

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:
ahmjv@yahoo.com