Egypt–SPIN Newsletter

Issue 8, Oct. – Dec., 2004

Sponsored by SECC

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

Welcome to our 8th 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.

First of all, we would like to congratulate **ITWorx Company** and its team about their achievement of reaching **CMMI level-2**. By this achievement, ITWorx becomes the 6th company that enters the clique of high quality companies in Egypt.

It is pleasure to announce that on 28 December, 2004 the **SPIG for SMEs Awareness Seminar** had been conducted at the Smart Village by SECC. SECC had conducted that seminar to highlight the efforts in the field of generating an **adopted model** for Software Process Improvement (SPI), which is suitable for SMEs. The model is based on the SW-CMM, CMMI, and IEEE standards and covers seven process areas that had been selected by SECC, Motorola experts and local consultants. **Dr. Gamal Aly**, the SECC Director, demonstrated the different phases of the project, working group efforts and introduced the **Software Process Implementation Guide (SPIG)**. Each member of the consultant working group demonstrated a process area, showing its benefits, elements and structure in the implementation guide. The implementation guide helps SMEs to increase their abilities to start their own process improvement projects on a sold basis. It is written to be simple, easy to read and suitable for direct implementation in the level of day-by-day activities. SECC aims this guide to be a good leverage for SMEs in the process improvement journey.

By the 8th issue of the Egypt-SPIN newsletter, the newsletter has completed its 2nd year. We hope to complete our mission successfully in the next years. The SPIN committee appreciates any comments to enhance, develop and renew the newsletter.

This issue conducts some hot topics in software testing (1st article), discuss the future of software industry in Egypt (2nd article), share real life experience in the field of software estimation, risk management and agile project management (3rd, 4th and 5th articles respectively).

Eng. Omar Kamal starts the **first series** in the Egypt-SPIN newsletter. The purpose of the series is to discuss the **software testing techniques in the implementation level**. The first part introduces a tutorial on the control flow testing technique.

Eng. Amr Shaltoot continues his discussion about the future of the software industry in the Egypt. He also highlights the aids and barriers that face the welling industry for a **strategic point of view**.

Eng. Ahmed Adb El Aziz describes an easy approach for **software estimation**, which is suitable for object-oriented software projects, called Use Case Points. He started with historical

overview, and then gave a step by step guide of using it. Finally he gave his recommendation about using this methodology.

Eng. Yaqeen Hussam talks about the **risk management** in software projects according to the SEI paradigm. The main focus of his article is to give clear guidelines about how to prepare your action plan against risks.

Eng. Ahmed El Shikh introduces the concept of the **agile project management**, its benefits, suitable environment conditions, and differences from the classical project management methodologies.

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

Table of Contents

Software Testing Techniques Series: Control Flow Testing Tutorial. (Part 1)

By Omar Kamal

This article is the beginning of a series of articles aiming to introduce a group of important techniques that I found useful in designing test cases at different levels.

Those techniques may be used in either black box or white box testing depending on what are the sources used to develop those test cases. Test cases that are developed from requirements document are considered black box test cases and are used to validate the design and/or implementation.

Control flow Graph Technique.

The first technique that is extremely useful in testing a wide range of software applications is the Control Flow Testing technique. Test case designers responsible for testing software applications which depend heavily in complex logic and nested conditions benefit the most from that technique.

Historically, the flow of logic was presented by flow charts with standard notations and layouts. Flow charts were useful in visualizing high level business processes, but weren't practical in modeling more detailed low-level logic flows.

A control flow graph is similar to flow chart graph except that, it is much compact and ignores more flow details when compared to the flow chart.

A Software application that automates meeting rooms' reservation process is a good example that illustrates the difference between both charts. The following paragraphs were taken from the software's requirements document.

"The system should provide the user with a login screen, where a user can enter his/her username and password. If the user enters an incorrect password the system inform him/her that the password is incorrect. If the user password was correct, the system logs in the time and date, and shows him/her the latest rooms' schedule. The user may choose to request reservation of a meeting room from a starting date and time, to an ending date and time. If the user didn't request anything, the schedule screen remains the same and should be refreshed with the new schedule every 10 seconds.
If the user chooses to quit the application at that time, the system exits without confirmation.

In case the user asks for reserving a room. A dialog box appears with the information needed to reserve a room. The user enters the room number, the starting/ending date/time, and his/her full name. The system checks whether the data entered is valid. By valid we mean that the room number should really exist, and the date and time should be a working date and time and so on. If the data was valid the system acknowledges the user, and then stores the request in the database. The data is then queued for approval by room administrator and the system exits.
The room administrators logs-in the system and manually approve/disapprove each and every reservation, the system automatically sends a positive acknowledge for users with approved requests, and send a negative acknowledge for users with rejected request. Rejected request are also removed from the database."

The flow chart of the process is shown in figure 1. Notice that the chart contains both processing and condition symbols. A flow chart is usually used to visualize the steps in abstract manner and is seldom used for explaining detailed procedures. Detailed flow charts may be hard to develop, that is why they are rarely used by software tester although they may be useful if exist.

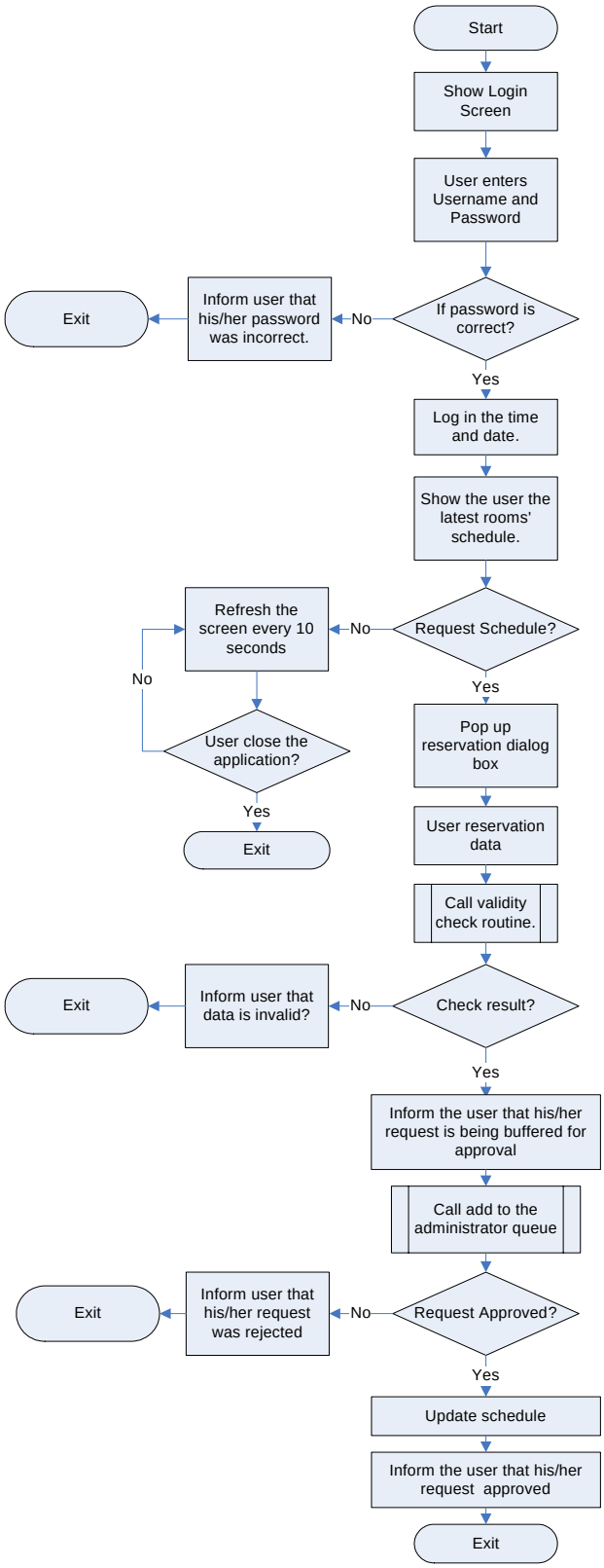
Another alternative graph, that is much simpler to develop and very useful for software testers, is the control flow graph. A control flow graph is a diagram that contains a reduced set of nodes as compared to flow chart nodes.

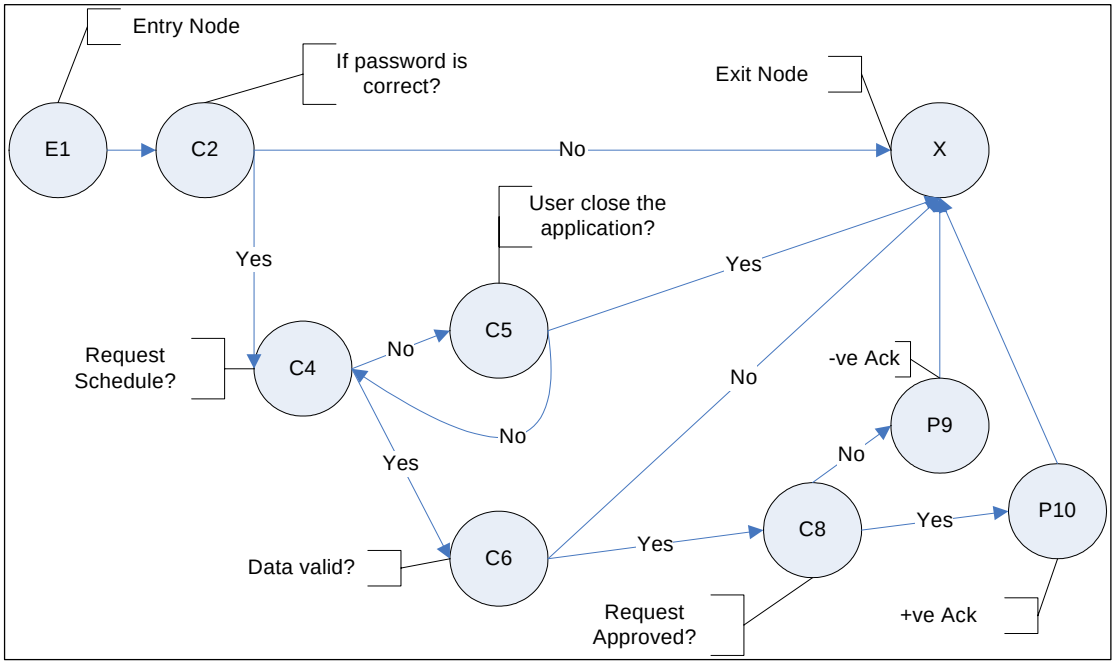
The control flow graph is developed by reading the requirements document carefully and underlining any of the following:

- Any word or sentence that represents the process entry (Entry Node).
- Any word or sentence that represents the process termination (Exit node).
- If, else-if, and else statements (Condition node).

Notice that processing nodes are ignored. Doing so, reduces the complexity of the graph, but also introduces more confusion. Processing nodes are included only when those are the only identifier between two adjacent control paths. For the sack of simplicity we will label the node as follows:

- The entry node starts with "E" and followed by an integer identifier.





- A condition node starts with the letter "C" and is followed by an integer identifier.
- A processing node starts with "P" and followed by an integer identifier.
- Finally, the exit node starts with "X" and may or may not be followed by integer identifier.

Nodes are connected using arrow with direction of control flow represented by the arrow head.

It just took us five minutes to sketch a control flow graph. The next question that rises is how to benefit from such graph?
As a software test designer, this graph is a model of the specifications and will be used to develop test cases.

The Technique:

Cover the graph from the entry node to the exit node. Write down the name of all nodes that appear in each path you select, for example:

- E1, C2, C4, C6, C8, P9, X.
- E1, C2, C5, X.

You may find huge number of paths which increases exponentially when the number of conditions increases. Should we select all path combinations (P∞) in-order to cover the graph?
The answer is it depends. There are number of test coverage criteria that vary in their strength. The following is a list of test coverage criteria [BEZ2]:

- Path Testing (P∞): Select all combination of nodes and branches. This is almost impossible for nearly all software applications except very simple cases with no loops.
- Statement Testing (Node Coverage) (C1): Select the minimum set of paths that cover all nodes. This is equivalent to say that we covered every statement in the specification at least once. We denote 100% node coverage using C1. If the model

represents a high level requirement specification, C1 is the minimum acceptable criteria that a testing team should achieve.

- Branch testing (Link Coverage) (C2): Select the minimum set of paths that cover all branches. Covering all branches will implicitly cover all nodes. We denote branch covering using C2. C2 is off course stronger the C1. A 100% branch covering is equivalent to 100% link coverage.

There are more coverage criteria than those listed in this tutorial. For further details refer to [BEZ1], [BEZ2].

The next section will demonstrate selecting paths based on both C1 and C2 criteria. Before starting the exercise, examine carefully the model, you will notice that there is a loop between "C4" and "C5" nodes. For the sack of simplicity we will ignore this loop by assuming that customer will choose to exit directly after the first schedule refreshment. (I.e. the loop will execute only once). Loop testing techniques are explained in details in [BEZ2] and are out of this tutorial's scope.

Node Coverage (C1):

Let us try to write down the set of paths that cover all nodes from the entry node to the exit node. Our first trial will be as follow:

- E1, C2, X.
- E1, C2, C4, C5, X.
- E1, C2, C4, C6, C8, P9, X.
- E1, C2, C4, C6, C8, P9, X.

Notice that we coverall all nodes with only four test cases. You can double check by verifying that all nodes are mentioned at least once. If you give it another thought, we can even remove the path "E1, C2, X" because all of its' three nodes are included in the next following path "E1, C2, C4, C5, X". So, we end up achieving 100% Node Coverage (C1) using only the three paths:

- E1, C2, C4, C5, X.
- E1, C2, C4, C6, C8, P9, X.
- E1, C2, C4, C6, C8, P9, X.

Branch Coverage (C2):

By now, you must have expected that you will pay more effort to achieve C2 coverage criteria than you did in achieving C1 coverage criteria. That is true, start from the entry node, write down every condition node that appears in your path twice, with it's' "Yes" and "No" branch. Continue till you cover all "Yes" and "No" links. Notice, that if a branch was already listed in an earlier path, you don't need to include it again. This is very important, because some test engineers have a common misunderstanding; they think that branch coverage is done by including all combination of all branches. That is incorrect; selecting all combination of branches is called "Path testing" not "Branch testing". Back to our example, the following set of paths cover all branches:

- E1, C2, X.
- E1, C2, C4, C5, X.
- E1, C2, C4, C6, C8, P9, X.
- E1, C2, C4, C6, C8, P9, X.

Notice that the branch from "C5" to "C4" is not included in any of the listed paths. Why?

Remember that we assumed that the loop between "C4" and "C5" will occur only once. If we plan to do our testing job flawlessly we need to add another path which includes this branch. To do so, we can assume that the loop will be executed twice.

- E1, C2, C4, C5, C4, C5, X.

Now we have one more path to add. We can delete the path "E1, C2, C4, C5, X" from our set and replace it with the new path. This replacement will do both jobs because it covers both C5 branches. Our final set of paths will be:

- E1, C2, X.
- ~~E1, C2, C4, C5, X~~.
- E1, C2, C4, C6, C8, P9, X.
- E1, C2, C4, C6, C8, P9, X.
- E1, C2, C4, C5, C4, C5, X.

We have four test cases developed to achieve Branch Coverage as compared to three test cases developed to achieve Node coverage. Notice that we can't delete the path "E1, C2, X" as we did in the before, because by deleting this path, we don't have any path that cover the branch (C2, X).

Predicate Coverage:

Conditions could be simple like node "C8" - "Did the administrator approved the request?" the answer is either a "yes" or a "no", or it could be a compound condition like the one discussed below.

Reviewing the requirements and revisiting our analysis for node "C2" will uncover a hidden defect caused by incomplete requirement specifications. The requirements states only the behavior if the password was correct or incorrect. What if the username was invalid?
The requirements should provide us with the system behavior for the following cases:

- Username is valid and password is valid.
- Username is valid and password is invalid.
- Username is invalid and password is valid.
- Username is invalid and password is invalid.

Actually, the third and forth case should be the grouped in one case. As the system would login only if a username matches an already stored username. So, the password in the third and forth case is a "don't care" input field. Accordingly, the case that if occurred allows the user to login should be *[if (username = VALID and password = CORRECT)]*.

Covering all predicates in all conditions is called "Predicate coverage". Predicate coverage is stronger than both branch and statement coverage criteria. Figure 3 shows how Node C2 is changed to reflect the compound predicate.

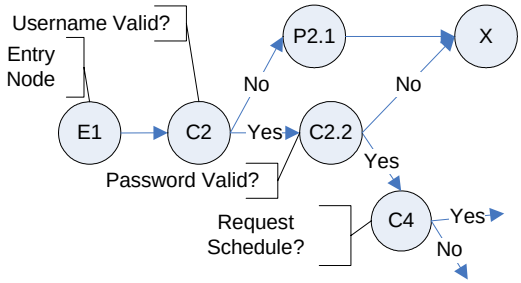


Figure 3: Node C2

The amount of test cases will increase if we choose to include all predicates associated with all conditions in the graph. Also, the chance for catching more bugs will increase proportionally.

Switch Node:

There are cases where a condition may include more than two predicates. Plotting those cascaded conditions may appear cumbersome and the graph will be loaded with more nodes which make it hard to visualize. That is why; a new type of node is introduced. A Switch node is the same as condition

nodes except that it has more than two branches. In the "username and password" case, the graph will be simplified if we replace the old condition nodes with the new switch node. Figure 4 illustrates the changes introduced by replacing two condition nodes by one switching node with three branches.

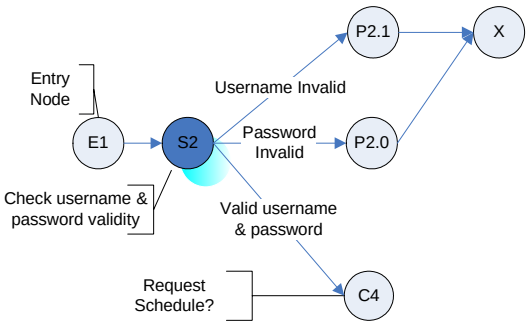


Figure 4: Switch Node.

The new Control Flow Graph (CFG) is shown at the end of this article. Introducing the new switch node will increase the number of paths from the entry to the exit node. I won't list those paths, and I will leave it as an exercise for the reader to practice.

The upcoming article:

This ends up the first part of the control flow testing technique. The next article will continue the discussion in the following topics Insha'Allah.

- The node linked-list format.
- Path synsitization.
- Outcome prediction.
- The Control flow technique limitation.
- Automating the control flow testing.

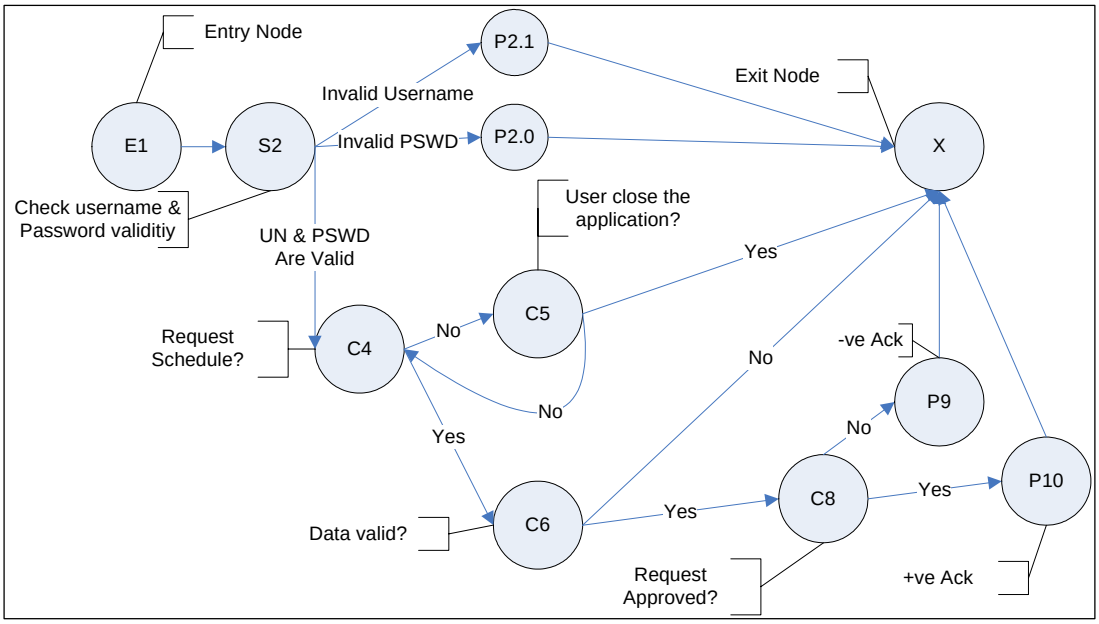


Figure 5: The New Control Flow Graph.

References:

- [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

Understanding the word Strategic!

By: Amr Shaltoot.

In the last article, I discussed –from a wide angle view, how we can be prepared for export. In this article I'll continue to share my thoughts about these paving concepts, to form a wealthy and healthy environment which leads to make 'others' be demanding on what I call it: a-software-that-have-an-edge, produced in Egypt and/or by Egyptian companies.

That edge can be:

a. An innovative[1] idea.
b. Enhancements on demandable applications that Already-Exists in the market, then to produce these new enhanced[2] applications under new trademarks and patents (in terms of processing speed, integrity, new-functions, more user-friendly…etc.). Also companies who made such enhancements can wisely sell these modifications to any of the current market producers of such applications.
c. Cost[3] (price and ROI factors must be taken in consideration: license metrics, Labor, support…etc.).
d. Quality, free-of-defects, and secure production.
e. Other discovered in the future edges.

The first step to do the above… is The Egyptian Software World Observatory for a marketable products and services (EGYSWOB)[4].

The EGYSWOB will first construct a Common Body of Knowledge (CBK), then by observing all and every[5] application in the market and even those does-not exist any more! EGYSOWB will produce detailed periodicals that describe:

a. Previously introduced applications and which are still in the market and which are not and why.
b. Currently introduced to the market, and what is the status of user response to the application or its new version. We all remember how many times, new introduced versions, damaged the solid market place acquired by the previous version of the same application! This happens to every industry, think about cars, or movies! This happens because of ***bad management***

[1] Innovations made must be marketable to the intended user-group.

[2] A marketing research must be done to users world-wide or nation wide (as per the intended user-group) to stand by the most important functions and bugs they want to enhance and how they want them to be.

[3] Cost can be categorized under enhancements, but I rather prefer it to be in a separate category.

[4] Please refer to September SPIN newsletter, ***Understand the word Export!*** http://www.secc.org.eg

[5] Whenever this term appears, a plan must be set to priorities items.

c. **A <u>Prediction</u> to what market may need!** This prediction can be done by analyzing the previous two functions and implementing market and statistical researches.

Can you tell: what BI "Business Intelligence" is?! How it was introduced to the market? And what Microstrategy® (founded 1989) did to respond to the market research about 3D reporting or multi-dimensional? How many opportunities like this one available to the Software Industry in Egypt? Call Centers, Financial Applications, and other highly penetrated areas are not the whole market. What if any of the formed Egyptian companies formed in the same year (1989) made a solution like what Microstrategy® did? Where would this solution reached for now…just tell me about management and HR! Just tell me about *RESEARCH!*

All the above will help Egypt to start the building of the Global Application Image professionally. The *horizontal* and *vertical* classification of industries in addition to *specialized* ones, will clearly pin-point any market gaps in application existence itself or lack of *Customer Satisfaction* of currently available solutions.

Building this image accurately, will consume at least two to three years, and a fortune, but will convert The Software Industry in Egypt, into a matured industry. It will help us construct an Agenda. This Agenda will be a guide. Those who will find it hard to believe that what they started 20 years ago seems to be unworthy, and will emotionally refuse results…will be somehow out of the 21st century game.

On the other hand, the *few* who will succeed to *change and adapt* to the newly discovered or predicted market needs[6], will harvest the gold and maintain brighter-knowledgeable future.

Below is a briefed quickly-made imagination to what can EGYSOWB organization-chart be:

Board of Trustees: Will keep an eye on investors and Venture Capitalists' investments, and what if the EGYSWOB on the track or not, in addition to establish self-correcting practices to correct any deviations if any. Remember perfection is ever never achieved itself, but you can always keep defects to an acceptable percentile of error (i.e. +/- 0.000005!)

Business and Technical Committee: Will be responsible for placing strategies that will align both business requirements and technical ones together.

Legal Advisory: Will provide all necessary legal advisory especially in the fields of property rights.

HR: This very critical department will perform head-hunting the best personnel in Egypt and the Arab world, beside any other necessary expertise needed world-wide.

Board of Directors: This is the heart of the EGYSWOB

Legitimate Reverse Engineering Unit: The deep-sensing operations and the factory of crucial and most-valued knowledge base for the EGYSOWB.

Marketing Unit: It is the unit which will provide EGYSWOB with Business Intelligence and every available business research in areas of focus to the EGYSWOB.

[6] Not just Customer requirements, but also market directions for the software industry itself, in terms of software engineering, visionary, and futuristic practical-senses.

Operation Unit: Contains all other helping units and sub-units and it acts like the spine for the human body.

Practices should eliminate it out with no mercy and apply hardest penalties to whomever the cause of this breach in robustness.

Only capable people honored with top-notch real and genuine achievements can join.

I believe that only research and real work can lead to a fruitful and solid successful business.

Outsourcing may be good for now, but 20 years from same 'now', poorer countries will see it an opportunity!

Finally, I'm not sure if I'm going to continue in this subject or shall I move to another one. Tell I finish my research ☺ and decide where to go…I wish all the luck to this industry in Egypt.

Biographies:

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

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Email:
amr@shaltoot.net

Software Estimation Using Use Case Points Method

By: Ahmed Abd El Aziz

Definitions

The following terms are used in the article:

API:	Application Programming Interface
AUCP:	Adjusted Use Case Points
EF:	Environmental Factor
EI:	External Input
EIF:	External Interface File
FPA:	Function Point Analysis
GUI:	Graphical User Interface
ILF:	Internal Logical File
ISSR:	Institute of Statistical Studies and Research
TCF:	Technical Complexity Factor
TFactor:	Technical Factor
UAW:	Unadjusted Actor Weights
UCP:	Use Case Points
UUCP:	Unadjusted Use Case Pints
UUCW:	Unadjusted Use Case Weights

Introduction

In object-oriented software development, use cases describe functional requirements. The use case model may be therefore used to predict the size of future software system at any early development stage. This article describes a simple approach to software estimation based on use case models: the "**Use Case Points Method**". The method is not new, but has not become popular although it is easy to learn. Reliable estimates can be calculated in a short time with the aid of a spreadsheet.

History of Use Case Points Method

In 1993m the "Use Case Points Method" for sizing and estimating projects developed with the object-oriented method was developed by Gustav Karmer of Objectory (now Rational Software). The method is an extension of "**Function Point Analysis**" and is based on the same philosophy. Karmer's work on Use Case Point metrics was written as a diploma thesis at the University of Linköping.

Domain of Use

Use Case Points method is most suitable for projects that follow the object-oriented methodology. Even though it could be used with any type of software, but in the case the use case analysis step will be only used for estimation and will not be used as the basis for the design of the project, which is one of the main advantages of the Use Case Points method.

Steps of Use Case Points Estimation Method

1. Use Case Analysis

Use Case Points can be counted from the use case analysis. The details of use case analysis are out of scope of this article. Please note that the use case analysis is part of the object-oriented methodology, therefore the use case analysis may not be considered as part of Use Case Points estimation method.

2. Classify Actors

Actors are classified as Simple, Average, or Complex. A weighting factor is assigned to each actor type. The following table summarizes the actor type, weighting factor, and classification rules.

Actor Type	Weighting Factor	Classification Rule
Simple	1	Another system with a defined Application Programming Interface (API)
Average	2	Another system interacting through a protocol such as TCP/IP.
Complex	3	A person interacting through GUI or a web page.

3. Calculate the Unadjusted Actor Weights (UAW)

Based on the results of actor classification step, UAW is calculated by the formula:

UAW = COUNT (Simple Actors)
+ COUNT (Average Actors) X 2
+ COUNT (Complex Actors) X 3

4. Classify Use Cases

Use cases are classified as Simple, Average, or Complex based on the number of transactions in the use case description, including secondary scenarios. A transaction is a set of activities, which is either performed entirely, or not at all. Counting number of transactions can be done by counting use case steps.

Another mechanism of measuring use case complexity is counting analysis classes which implement it, but this requires the analysis to be finished before using the Use Case Points method.

A weighting factor is assigned to each use case type. The following table summarizes the use case type, weighting factor, and classification rules.

Use Case Type	Weighting Factor	Classification by Transactions	Classification by Analysis Classes
Simple	1	<= 3	<= 5
Average	2	4 – 7	6 – 10
Complex	3	> 7	> 10

5. Calculate the Unadjusted Use Case Weights (UUCW)

Based on the results of use case classification step, UUCW is calculated by the formula:

UUCW =COUNT (Simple Use Cases)
+
COUNT (Average Use Cases) X 2 +
COUNT (Complex Use Cases) X 3

6. Calculate the Unadjusted Use Case Points (UUCP)

UUCP is simply the sum of UAW and UUCW.

UUCP = UAW + UUCW

7. Calculate the Technical Complexity Factor

The method also employs a technical factor multiplier corresponding to the technical

complexity adjustment factor of the Function Point Analysis method.

There are 13 technical factors. Each factor has its own defined weight. The estimator give each factor a degree ranging from 0 to 5 based on its influence on the project. The following table summarizes the technical factors and their weights:

Factor	Description	Weight
T1	Distributed System	2
T2	Response adjectives	1
T3	End-user efficiency	1
T4	Complex processing	1
T5	Reusable code	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Security features	1
T12	Access for third parties	1
T13	Special training required	1

The Technical Factor (TFactor) is the summation of the product of the weight of each factor by its degree.

The Technical Complexity Factor (TCF) is calculated by the formula:

TCF = 0.6 + (0.01 X TFactor)

8. Calculate the environmental Factor

The method also employs an environmental factor multiplier in order to quantify non-functional requirements such as ease of use and programmer motivation.

There are 8 environmental factors. Each factor has its own defined weight. The estimator give each factor a degree ranging from 0 to 5 based on its influence on the project. The following table summarizes the environmental factors and their weights:

Factor	Description	Weight
T1	Familiar with RUP	1.5
T2	Application experience	0.5
T3	Object-oriented experience	1
T4	Lead analyst capability	0.5
T5	Motivation	1
T6	Stable requirements	2
T7	Part-time workers	-1
T8	Difficult programming language	-1

The EFactor is the summation of the product of the weight of each factor by its degree.

The Environmental Factor (EF) is calculated by the formula:

EF = 1.4 - (0.03 X EFactor)

9. Estimate the Size of the Project

The size of the project is expressed in Adjusted Use Case Points (AUCP) – simply Use Case Points (UCP). It's the product of the UUCP, TCF and EF.

UCP = UUCP X TCF X EF

10. Estimate the Effort of the Project

Simply multiply the UCP by the number of staff hours per each single use case. Field experience has shown that effort can range from 15 to 30 hours per use case point, and its most probably 20 staff hours per each use case point.

My own recommendation is that you get this figure from your own historical information. Review past projects database and divide the total effort by use cases to get the effort hours per use case. As the number may differ from one project to another, it is strongly recommended that you use the number calculated from similar projects (e.g.; same business domain, same technology, may be same programmers …).

Quicker Approach

Sometimes – actually most of times – you receive a proposal or requirements list and you are asked to estimate within 48 hours maximum. You have no time to do neither detailed use case analysis nor classifying actors and use cases. What may you do then?

I recommend that you do the following:

1. List all actors and all use cases with no further details.

2. Do not classify actors and use cases into simple, average, or complex. Simply consider all actors and all use cases as simple.

3. Calculate UUCP using the formula:
   UUCP = COUNT (All Actors)
   X 2
   COUNT (All Use Cases)
   X 2

4. Continue with steps 7 – 10 detailed above.

Advantages of Use Case Points Method

Use Case Points method – like Function Points Analysis method – has many advantages over traditional estimation methods like lines of code in that:

- It expresses the project size from the user point of view, not from physical point of view.

- It is independent on the software development technology.

- Could be made in the early phases of the project life cycle. Once you know the client requirements, you can estimate the project. You do not have to wait to the analysis or design phase.

However Use Case Points Methods has two main advantages over Function Points Analysis Method:

- It is easier to estimate. FPA requires experts to do the estimation.

- It is based on use case analysis, which is the base for object-oriented design. In other words you benefit from the documents you produced before in more than one purpose. In FPA, you have to define Internal Logical Files (ILFs), External Interface Files (EIFs), and External Inputs (EIs) only for the estimation purpose and you do not use them anywhere during the life cycle of the project.

Disadvantages of Use Case Points Method

Use Case Points Method has the following disadvantages:

- There are international standards on how to count Function Points. The concept of Use Case Points, on the other hand, has not yet reached the level of standardization. Without a standard describing the appropriate level of detail in the requirement description, i.e., the use case model, there may be very large differences in how different individuals and organizations count use case points.

  For example the weighting factor of the use cases. Some estimators give simple, average, and complex use cases the weights 5, 10, and 15 respectively, while others weight them 1, 2, and 3 respectively as used in this article.
  Hence, it is currently difficult to compare Use Case Point values between companies.

- Although there are too many case studies and researches in Use Case Points Method, it is not still widely used compared to other estimation methods.

- There is still no certification in Use Case Points estimation.

Recommendations

By comparing the advantages and disadvantages, I strongly recommend estimating software with Use Case Points Method. Lack of standard, spread or certification does not necessarily imply that the method is not good. I tried it few times, especially the quick approach, and it gives reasonable estimates very quickly.

Why do not we build our Egyptian standard? Let's start with the techniques described here and share our results and experience. I recommend that we estimate by Use Case Points beside any other techniques we currently use and compare the results. Then we have to share our results and experience. If you agree with me, please contact me to discuss the best way to do this.

More Resources

By searching the Internet about "Use Case", you will find many results about Use Case Analysis and Use Case Points. Some of the URLs I see them useful are:

1. http://www.zoo.co.uk/~z0001039/PracGuides/pg_use_cases.htm
2. http://www.globaltester.com/sp7/usecasepoint.html
3. http://www.bfpug.com.br/Artigos/UCP/Damodaran-Estimation_Using_Use_Case_Points.pdf

4. http://www.visual-paradigm.com/VPGallery/usecase/

5. http://isds.bus.lsu.edu/cvoc/learn/bpr/cprojects/Spring1998/modeling/usecase.html

There are also many software tools that help you in using Use Case Points much faster. You may even create a simple Excel file to help in that.

References

1. Use Case Points, An Estimation Approach, by Gautam Banerjee, August 2001.

2. Use Cases and Function Points, http://www.ifpug.com/Articles/usecases.htm.

Biography

Ahmed Abd El Aziz, has more than seven years experience in the Software Development field. He is now the Internet Department Manager in HARF Information Technology. 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

Planning the Action

"An overview on Action Planning in the Risk Management Process "

By: Yaqeen Hussam

A. Introduction

All projects have some level of risk associated with them. Even if the product under development is simply another version of an existing system or product, risks may appear in areas such as:

- Changes in development personnel (and resulting experience levels with the product).
- Changing market conditions and customer expectations.
- Changing business conditions for the development organization.

In general, it is not practical to assume that all risks can be detected, eliminated or significantly reduced. Many will be accepted, while many others will merely be watched and acted upon if conditions change, but also the more you understand the risks, the better equipped we are to manage them.

Risk and opportunity go hand in hand. Success cannot be achieved without some degree of risk. Risk in itself is not bad; risk is essential to progress, and failure is often a key part of learning. But we must learn to balance the possible negative consequences of risk against the potential benefits of its associated opportunity.

B. Risk Management (SEI Paradigm)

Risk management is a process that is systematic and continuous and it can best be described by the SEI risk management paradigm.

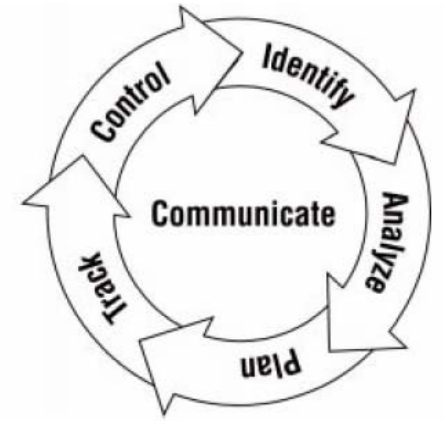


Figure 1: Risk Management, SEI Paradigm

Identify: Makes all known project risks explicit before they become problems.

Analyze: Transforms risk data into decision-making information.

Plan: Translates risk information into decisions and mitigating actions (both present and future) and implements those actions.

Track: Monitors risk indicators and mitigation actions.

Control: Corrects for deviations from the risk mitigation plans.

Communicate: enables the sharing of all information throughout the project and is the cornerstone of effective risk management.

C. Planning

The Importance of System's perspective

Action planning for risks is the determination and implementation of actions necessary to manage a program's risks. And it's where the integration of risk management processes with existing program management becomes most evident and we can say that planning, in general, is an integral part of program management (i.e. planning how to meet specific milestones or determining the best design strategy for meeting specified requirements) that's mainly due to the key factor that risk planning requires a system's perspective to maximize the effective use of scarce resources within a program.

Planning Objective

Planning the mitigation of risks is a proactive means of minimizing future problems with Actions range from accepting the risk (do nothing) to developing strategies. The key to planning for risks is to make as effective and efficient use of resources as possible to reduce the overall risk while maximizing the potential for gain to the program.

Planning Process

Strategies for the most important risks are generally planned first. Plans for other risks should leverage off these strategies for maximum return on investment in risk mitigation strategies (e.g., greatest reduction in risk exposure for the least expenditure of resources). All risks are reviewed, but only those that are important to the program will have any significant amount of resources expended on them. In general, it is not practical to assume that all risks can be eliminated or significantly reduced. Many will be accepted, while many others will merely be watched and acted upon if conditions change.

Risks are reviewed by a review group. This technical review is a periodic activity that not only ad-dresses new risks, but also reviews the status of existing risks. The review group should manage its review between visibility and attention so that critical risks should receive the needed attention while less critical risks do not need the full attention of top level management, but visibility into those risks can be provided if desired.

Once reviewed, risks are then assigned throughout the organization to be dealt with by the appropriate personnel. Planning, then, is a repetitive or cyclic process carried out throughout the organization and adhere a systematic approach to ensure the appropriate visibility and delegation of responsibility.

There are two basic phases in planning:

1. **Review and assign responsibility**

   The review and assign responsibility phase is performed as part of a periodic technical review by responsible group (and sometimes at the interorganizational level) to determine which risks can be dealt with easily and which ones will require (and justify) expenditures of time and effort to develop detailed analyses and strategies.

2. **Strategize.**

   If Actions required and responsibilities can't be detected, or visualizations can't be clear, we need to

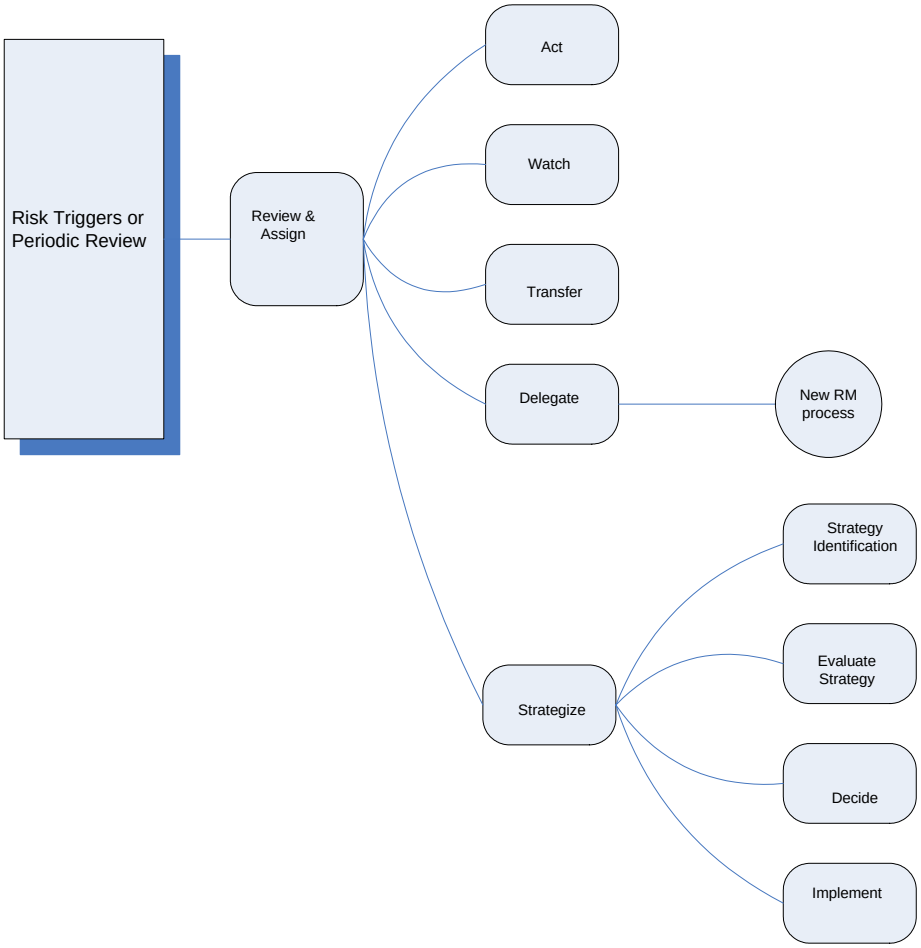


Figure 2: Planning Process – Overview

strategize, which is a kind of action, where we try to identify different strategies, evaluate them, and decide one (or more) to use.

D. Review and Assign Responsibility

The Review and Assign process step is the logical first step upon identification of a risk. Once a risk has been identified, then a decision is made regarding what to do with it, and who will be responsible for it. This activity occurs after a risk has been identified and is accomplished by an individual or organization with a total project perspective.

During a review, it is possible to identify and decide upon actions for some risks, while others may need further investigation or consideration before action can be taken. Also, there will be risks which are appropriate to delegate and can be handed to others to manage, while there can be also risks that will need new complete phase to strategize. Regardless of the type of action decided upon at this review, an assignment of responsibility is made for each risk. At the end of this review and assign activity, designated actions are carried out and recorded for each risk.

New risks are reviewed, unless they are designated as critical. Critical risks are reviewed immediately.

The review and Assign responsibility phase is usually introduces through two major processes, periodic technical reviews and action type determination.

Periodic technical reviews

Periodic technical reviews are periodic internal meetings held to review and assign responsibility for new risks, to review status, and to assess the progress of risk mitigation strategies (this is related to the tracking and control phase of SEI paradigm). The key element is that the review process occurs and is accomplished through methods consistent with the program's routine program management activities.

As planning is just a phase in the integrated SEI paradigm, new risks should be identified from routine risk identification and analysis, transferred from other organizations or identified as part of a baseline risk assessment.

And the result of these meetings of the periodic technical reviews is the determination of action(s) and the assignment of person(s) responsible for carrying out the action(s).

Action Type Determination

As each risk is reviewed, an initial determination of a type of action is made. A responsible group is assigned the action and the responsibility to report status back to the reviewing group or manager. The action decided upon, as well as the responsible person and risk information are documented and from this point on, additional information or actions should be is gathered and added as the risk moves through risk management activities.

When reviewing a risk, there are five likely types of actions that can be decided upon:

1. Act
Some risks are easily resolved by a quick-response or rapid turnaround type of action that requires little or no expenditure of resources. In this case the reviewer is prepared to make an immediate decision based on current information. These risks are immediately **acted** on.

2. Watch
Some risks are to be **tracked**, but expenditure of resources for any other type of action or investigation is not warranted at this time.

3. Transfer
If risks are identified by an organization but the authority and accountability to actually deal with the risk lies elsewhere these risks are **transferred** to the other organization.

Note that a successful transfer requires acceptance on the part of the receiver.

4. Delegate
There will be risks more appropriately addressed within another part of the organization or sometimes out of the organization. These risks are **delegated** down the chain of command to another person (or organization). The planning process is repeated by that person, starting with the review and assign responsibility step.

5. Strategize
Not all the decisions on the disposition of the risk can be made with current Information and resources. It is needed in these situations to either further investigate the risk itself or to

develop **detailed strategies** and schedules for mitigating actions. Alternative strategies are identified, evaluated, selected and implemented.

Strategize

Strategize is a process step that involves some expenditure of resources to perform and to identify possible alternative strategies, evaluating those strategies for maximum Benefit, deciding upon the appropriate strategies, and implementing them.

Strategies Types

Strategies include action(s) to take in the near-term or future. Future actions, or contingency plans, address actions to be taken when resources are available, a trigger event occurs, or when the risk becomes a problem.

There are generally four basic types of strategies:

1. Accept
Accept the consequences of the risk happening–essentially, do nothing. These risks are not tracked.

They are documented and put aside. If the risk becomes a problem, it is handled as a problem. This is an appropriate strategy for those risks which are simply not worth the effort to deal with. Changes in conditions which result in a risk that does need to be dealt may be reflected in the identification of "new" risks (re-identified).

2. Research
Some risks require in-depth investigation to determine the root causes; some potential strategies require investigation to determine what benefit will be gained. These risks require **research** (and sometimes prototyping), and upon completion of the research, the risk is reassessed, based upon the additional information by repeating the review and assign step of the planning process.

3. Eliminate
Risks that can be **eliminated** should be. This judgment is generally based upon the cost of eliminating the risk versus the costs of potential impact and the likelihood that it will occur.

Risks that are eliminated are closed when the strategies to eliminate them are complete.

Continuous risk identification will identify the risk again as a new risk should conditions change such that the strategies taken are no longer effective. Early identification and elimination of unnecessarily risky areas is a means of preventing risks.

4. Reduce
Risk exposure is a measure of the impact of a risk on the program and the likelihood or probability that it will occur. In **reducing** the risk exposure one can reduce either one or both of these factors.

5. Hybrid
A combination of any of these strategies is also possible (e.g., perform research into the effectiveness of alternative designs while taking minor action to reduce the risk as well).

The Strategize Process

1. Identify Strategies.
Strategies can be identified through any number of existing problem-solving techniques. Each

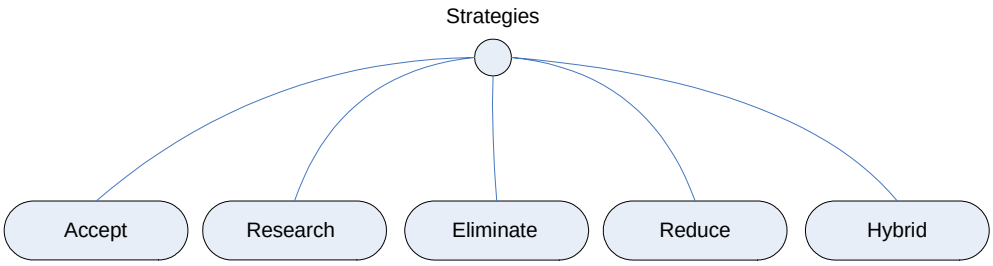


Figure 3: Strategies Types

viable alternative strategy should Include, description of actions to be taken, estimate of required resources, estimated schedule, estimated benefit or change in the state of the risk.

It's very important to identify the relationships or dependencies to other alternative strategies (e.g., strategy B enhances the results of strategy A if both are used)

2. Evaluate Strategies.
The alternative strategies can be evaluated to determine which one has the best potential for managing a particular risk with respect to the current environment, the available resources and the requirements.

Tables or matrices can be constructed to evaluate the strategies for a given risk, if the choices are complicated.

Two types of evaluation can be performed quantitative and qualitative.

Evaluation can be done on a qualitative basis, but there will be a high degree of subjectivity and dependence on the expertise of the evaluators.

Quantitative evaluation would provide more accuracy but the cost (and precision) of quantitative measurements of benefit and cost often require considerable resources.

A single strategy may resolve many risks, or may cause conflicts with other strategies. The best solution for any given risk is not necessarily the best answer for the program, but it's a set of strategies that best manages the program's risks.

Each program must identify a set of criteria by which strategies are evaluated. These criteria can include budget, personnel, schedule, product quality, specific components, return on investment, components dependency and others. Good strategies, then, are those which meet the selection criteria.

Then we can construct matrices or trees to view a set of strategies relative to the criteria, for example:

- **Dependency trees** show the dependencies between risks, and therefore, identify those risks or conditions leading to those risks, which, if eliminated or greatly reduced, can have the same beneficial effect on several risks.

- Strategies for these root causes should be emphasized.

- **Dependency / Interaction matrices** identify the interactions between strategies and highlight conflicts and synergy.

- **Cost/Benefit** matrices help correlate required resources and predicted gains.

- **Schedule and Dependency matrices** highlight which strategies need to betaken first.

- **Strategy vs. Program area matrices** identifys areas of the program which may need additional budget reallocation.

The key to any of these types of matrices or relation diagrams is to provide information to support informed decisions. For a large set of risks, or complicated, interrelated strategies, it is vital that the relationships between risks and their potential strategies be understood before resources are committed. As new risks are identified and planning begun, their relationships to existing risks and strategies must also be understood to avoid unnecessary duplication of effort or negation of actions already taken.

3. Decide on a strategy.
During the process of evaluation, alternatives are eliminated that don't meet the selection criteria. This is documented along with the other risk information to capture complete information on each risk (conditions and criteria do change with time).

The final decision is made based on the overall cost and benefit of the strategies to the program. These decisions are documented as any other program decision and this information is a part of the data retained for each risk and then strategies for risk management become tasks within the program.

4. Implement.
Implement is the connection with the next phase of the SEI paradigm, tracking, begins when action plans are implemented. Tracking is also used for those risks with contingency plans, to alert management to the need for implementing those contingency plans.

Risks are tracked according to plan to provide visibility into the current state of the risks. Strategies are tracked and status is reported as any other task within the program. Risk action plans, like the risks themselves, need to be managed to ensure the plans are implemented and effective.

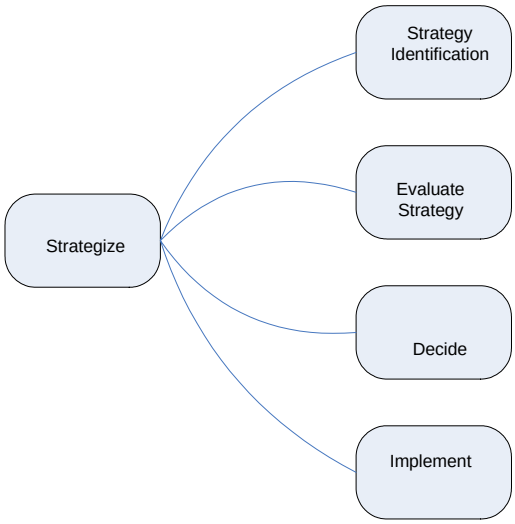


Figure 4: Strategizing Process

Biography

Yaqeen Hussam is experienced in Software Engineering. Working as a Software Engineer in QuickTel, Research and Development (R&D) department. He holds a bachelor's degree in Computer Engineering from Cairo University. He is an activist for IEEE actions in Egypt. Also he is interested in the Information and Communications Technology (ICT) interactions with society, social aspects and sociological studies and their interactions.

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:
yaqeen@ieee.org

Skilled Navigator: the one who can survive in the foggy environment. "*Agile Project Management*"

By: Ahmed S. El-Shikh

Just a new project appears in your company, some voices repeat familiar words such as: "W*e can not make a plan for that project, it will take a long time and we can not imaging what we will face*". Another one said: "W*e will make a perfect plan that predicts any thing and we will do our best to follow it*" may be. Other different situations generate other words, let us see: "*It is a repeated project and we know well what we will do, let us start. Do not waste time in making plans*". Finally, the most famous words: "*We have no time to plan, just start, every moment is essential. We know that planning is a very good and useful thing but not in our case, when we have time to do, sure we will do*" hope so. As I think, one day you had heard some or all of these words, especially if you work in one of the **SMEs**. In large companies, the situation can be the same in some projects, not all of course.

Some time we can find our self in the case that requires some flexibility and force us to agree to the opinion of the group, but really can we ignore planning to a reason or another? On the other hand, does the solution exist in the perfect detailed plan that must be followed whatever the signs those appear during the project running?

To some degree of confidence, I expect that you will agree with me that "**Zero-Planning**" projects will fail before the starting point. Also, the "**100%-Planning**" project can be theoretically accepted, but in the real life environment when you manage your project within fuzzy constraints it can be really hard to achieve.

We really need a **balanced approach** that can get benefits from planning and enable the avoidance of real life obstacles that can face the project and the project manager.

I can hear you say: "*I am an expert in my project management methodology and so familiar with it. I do not need to learn a new one*". The problem here is not what you know and what you do not, **the problem is**: "*What is the suitable methodology to your company and your project*".

You can make a fast **self assessment** about your business case and let the result answers you. You can use the following check list:

- My company has a stable strategic plan.
- All my stakeholders committed to my project plan.
- The scope and objectives are clear, complete and stable.
- My team is loyal to my project
- Our risk and quality plans are well defined and effective.
- My organization and peers understand the project management techniques.
- All required tools, technologies and techniques are available.
- The company and project environment are changing rapidly.
- My company future is not fuzzy.

You are a very lucky man if you are a member of an organization that enables you to be so bold to answer with "*Yes*" on all or the most of these statements. If the most of your answers are "*No*", I think these lines can help you to be the "**Skilled Navigator**" that we talk about.

In highly innovative businesses that focus on the edge of the most up-to-date technologies, such as software

industry, the project manager needs to be **agile**.

The **agility** here means that you have the ability to take smart, fast and right decisions in the rapidly changing environment. To be agile, you need to be able to maneuver around the obstacles that face you. You can face some problems from the type that can not be imagined or predicted in advance to occurrence.

Since the Second World War, the importance of formal and matured **project management methodology** is noticed. The start was in the governmental projects that contain a large number of sub-contractors that need high efforts to enable alignment of all these distributed sub-projects. A lot of large scale commercial companies followed this approach and do some modifications. Although these methodologies had shown successful achievements, but what can work well for large companies and governments can not fit for SMEs. Not just SMEs, also some large companies can conduct some projects that need to be managed in agile manner.

The need to develop a new **agile project management methodology** emerged from the dramatic economic shift occurred in the business all over the world. The economic was based on the elephant size companies that own the required assets and resources. Nowadays, economic is based manly on a large number of the small and medium size companies that can be faster than the large one in the adaptation to the new emerged technologies. This shift was a natural result to the new fields of industry that depends mainly on knowledge and intelligent. The software industry can be considered as the shiniest example to this type. It depends mainly on the human mind and the human factor is the most expensive assets inside it.

As we mentioned before, not only the SMEs can need the agile project management but also the large scale companies can need it in some cases. So, it is better to talk about the **agile environment not an agile company**.

The **agile environment** can be characterized according to two factors:

- *The spread of stakeholders.*
- *The type of the project.*

The spread of the stakeholders can be viewed from two points of view:

- Stakeholders are internally or externally.
- The number of engaged organizations.

The project type can be:

- Operational Project.
  This contains repeating of a well known operation steps.
- Technology Project.
  This contains conducting new challenges and some completely new issues.

The **wide spread of the project stakeholders** can force the use of the classical project management methodology. The clearest case appears when more than one organization is engaged in the project even if the project is a technology one; the agile methodology will not fit. More than one organization means distributed works and activates which need more clear conditions and defined boundaries.

In the real world cases, the urgency and speed of the conducted project can force the use of agility even if the project is done between two or more organizations. Just a higher degree of cooperation and matching in technical aspects are required in order to avoid misunderstand or conflicts.

	External Stakeholders	Internal Stakeholders	Single Organization
Operational Project	Classic	Classic	Classic
Technology Project	Classic/Agile	Agile	Agile

Characteristics of the agile environment

Now let us ask an **important question**: what can happen if we tried to apply the classical project management technique inside an agile environment?

The classical methodologies focus on building a **detailed plan** and fight to meet deadlines and budgets. So, the planning phase will consume a lot of time, effort and money in order to accurately predicate what can probably happen. In the agile environment, with rapid unpredictable changes, the concentrating on executing the plan will result in one of two things;

- We can force the project to the edge of failure since we do not let the plan to adapt to the business case.
- Or we will spend a lot of effort, approximately multiple times of the effort done to complete the detailed plan, to correct the plan and produce a number of new versions of the plan.

Conrad Brean had said before that: "*A good plan today is better than a perfect plan tomorrow*." This statement can be considered as a **golden rule** that fit in the agile environment.

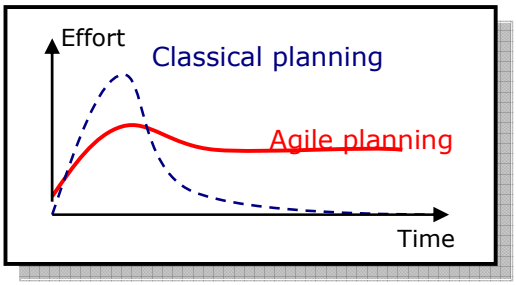
Just be careful; misunderstand of the word "**good**" can lead to a lot of disasters. The maturity of the company and the background level about project management techniques are two critical factors that can affect the definition of the word "*good*" inside your company. In a company with no project management technique at all, the work breakdown structure or even a task list with some estimated times can be considered as a good plan. But in a matured company the plan without risk management plan or value-added analysis can not be considered as good anyway.

We really need a **formal definition** of the word good. But first to be able to discover the word good we need to define the word perfect. The **perfect plan** is the one that had been done with the focus on identifying all the detailed aspects of small grain activates such as; time, resources and effort estimation. (I.e. an activity-based one that uses the bottom-up estimation). The **good plan** can be defined as the plan that holds an overview picture about the project from the start to end and focus on achievements rather than activates. Focuses on milestones not day-by-day tasks and contains detailed plan only up to the upcoming milestone, this is the **agile plan**.

In agile project management, the **planning activity is a continuous effort** during all phases of the project. It is not a concentrated effort in the planning phase only.

In other words; agile project management spend more effort in the **execution phase** than the planning phase; this does not mean that the

planning phase becomes less important.



Classical vs. agile planning

To illustrate this concept let us take an example. If you are the navigator of a small boat in a foggy environment, it is clear that your goal is to reach the beach in time and without crises. Imaging that you had spent a lot of time before the journey to expect what can happen, also you tried to estimate precise time to each event and activities, while you are in the boat, an unexpected thing appear in front of you, I think it is obvious that you will maneuver to avoid. Only the crazy one can execute his plan whatever the result. The unexpected obstacle does not surprise you because you failed to make a perfect plan, it appears because, in a foggy environment, you can not see or predict to a long distance from you. It is not your mistake, it is just a natural of the environment that you are expected to navigate in.

Be careful again, if you take the decision not to plan at all before the journey, the sudden event can freeze you or make you lose your control. You need to plan to traditional aspects and deal with the unexpected ones.

Now it is time to investigate the **rules of the agile project manager**. Is it the same as in the classical environment or not? Before conducting this point you must put in your mind that **the project management is an art not a science**, reading a lot of books or even holding a credible certificate is not the key. If you review the example of the navigator, you will trust these words; he really needs to use his creativity to maneuver not retrieve a word or two that he had read it before.

The agile project manager must:

- Be a facilitator more than a manger.
- Regularly scan the horizon to detect factors that can affect his project.
- Act as an information manifold.
- Build a strong honest relation with project stakeholders during calm period to grantee receiving help in the emergencies.
- Coach the team rather than play the taskmaster rule.
- Have a mile wide knowledge on a lot of fields and solid personal skills.

As you can see, the rules of the project manager in the agile environment are different for those in the classical one; also rules of your project's stakeholders encounter some changes. The most important key point in this aspect is that rules in agile environment are not characterized to be within a solid boundaries box.

Rules are designed to be permeable to the degree that enables others to cross those boundaries and become involved. Be calm, permeable rules do not mean that others will take your place or share you in your decisions, just it means that in some time, in emergency, hard times and crisis you can find a hand to help and support you. Again remember the navigator example and you will trust my words, when the bout face a sudden obstacle, some one can help you to maneuver to the left or to the right. His hands support you with the required force, he did not take your

position but you really were in need to an extra force to save your and his life.

Finally, agility does not mean eliminating the use of documentation and powerful project management computerized tool. Agility can, to some degree, increase the need to use professional tools to help the project manager, project team and other stakeholders in more efficient way. The combination of all these tools is called the **operational infrastructure** of the agile project management environment.

We still have a lot of points in the agile world to talk about. Hope we can do in the next articles if the God willing.

References

"*Agile Project Management: How to Succeed in the Face of Changing Project Requirements*" by Gary Chin, 2004.

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".

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:
el_shikh@sas-sys.com