Egypt–SPIN Newsletter

Issue 16, Jul. – Oct., 2007

Sponsored by SECC

From the Editor (Ahmed S. El-Shikh)

Welcome to our 16th issue of Egypt –SPIN newsletter. In each issue we try to put together relevant information in the form of articles and recaps covering the previous six months events hoping to provide our members of Egypt – SPIN with information to support their current interests.

We are used to announce the major events in the field of process improvement in Egypt. At the same time, it is a pleasure to announce many success stories and great achievements that happened during the previous period of time.

Please, refer to SECC website, www.secc.org.eg, for more information. Hereunder, you can find a summary list of the most important events:

- **3rd Quarter of 2007**
  - SEI course entitled "**Managing TSP Teams**" was conducted by two Egyptian SEI-Authorized PSP Instructors.
  - Two rounds of SEI courses entitled "**PSP for Engineers I & II**" were conducted by two Egyptian SEI-Authorized PSP Instructors.
  - **One Egyptian company** achieved **CMMI ML 3**; the SCAMPI A was lead by an Egyptian SEI-Authorized SCAMPI Lead Appraisal.
  - SECC participated in **GITEX Dubai 2007**.
- **4th Quarter of 2007**
  - SECC organized an **Egypt-SPIN Event** on "**SEI-Team Software Process (TSP^SM^) & Capability Maturity Model Integration (CMMI®) Product Suites Synergy**" at SECC premises. The speaker was **James McHale**, who is an SEI Senior Technical Staff Member in the SEI-TSP Initiative Team.
  - SECC organized a public event entitled "**Executive Awareness Workshop Six Sigma and DFSS for IT and Software**" at Grand Hyatt Cairo Hotel. The speaker was **Dr. Radouane Oudrhiri**, who is the CTO in Systonomy Consultation Company.

This issue introduces some hot topics in two series and two independent articles as follows: CMMI ML2 Extensions to ML3 (1st article), Software Reuse (2nd article), SW Process Improvement Traps and Code Analysis in VSTS (3rd and 4th articles respectively).

*Eng. Ahmed Abd El Aziz completes his series that raises a warring flag to software companies that are ambitious to achieve the **transition from CMMI ML2 to ML3 using traditional approaches in only one year**. In addition, he summarizes the minimum level of required **Extensions of ML2 Practices**.*

*Eng. Ahmed Hammad summarizes his experiences in the field of **Software Reuse** including the relation with business goals, requirements engineering and process support.*

***Eng. Sally Hassan** summarizes her experiences in the field of **SW Process Improvement**. Her article discusses the most famous traps that can face the process improvement efforts in any organizations, symptoms and possible solutions.*

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 aselshikh@mcit.gov.eg

Table of Contents

Egypt-SPIN Event on SEI-Team Software Process (TSP^{SM}) and Capability Maturity Model Integration (CMMI®) Product Suites Synergy

By the editor

On October 30, 2007 and with participation of more than 120 professional from 66 Egyptian software companies, **SECC** organized an **Egypt-SPIN** public event on **"SEI-Team Software Process (TSP^{SM}) and Capability Maturity Model Integration (CMMI®) Product Suites Synergy"**. The event was sponsored by the Egypt-SPIN Point of Contact (POC); **Dr. Gamal M. Aly** and was held at SECC premises in the Smart Village under supervision of Egypt-SPIN Coordinator- **Eng. Madiha Hassan**. The event was conducted as a part of SECC strategic objective to widely deploy the "**SEI-TSP Product Suite**" related services in the local and regional market. These services include the delivery of "**SEI-PSP Training**" courses and "**SEI-TSP Coaching**".

The speaker was **James McHale**, an SEI Senior Technical Staff Member in the "**SEI-TSP Initiative Team**". He has over twenty years of experience with industrial control systems as a programmer, designer, and project team leader. Since 1999, he has worked with the SEI's Team Software Process Initiative. He is an SEI-Authorized TSP coach, SEI-Authorized Instructor for Introduction to CMMI and SEI-Authorized PSP Instructor. In addition, he is an interviewer for the SCAMPI High Maturity Lead Appraiser oral examination, and is co-author of two SEI Technical Reports mapping the TSP to the software CMM and to CMMI.

The event included two sessions; the first one was on "**CMMI, TSP, and Software Quality**" and the second one covered "**TSP: Operational CMMI**". The speaker introduced many topics such as:

1. Different quality models and practices
2. Quality goals and plans
3. CMMI – a model of best practices
4. PSP – operational best practices for a developer
5. TSP – operational best practices for a team
6. TSP – implementation of CMMI practices on the project level
7. Benefits of "SEI-TSP Product Suite" adoption
8. TSP process elements, structure and flow
9. CMMI and TSP overlap and gaps summary
10. CMMI and TSP best practices
11. ROI of "SEI-TSP Product Suite" adoption
12. Accelerating the achievements of CMMI maturity levels by adopting "SEI-TSP Product Suite"

For more information, you can download the presentations in MS PowerPoint format from SECC website (http://www.secc.org.eg/SPIN Newsletter.asp).

SECC Public Event on “Six Sigma for IT and Software Engineering” for Executives

By the editor

On November 6, 2007, and with participation of more than 220 professional form Egyptian software companies, **SECC** organized a public event on **“Executive Six Sigma for IT and Software Engineering”**. The event was sponsored by SECC Chairman; **Dr. Gamal M. Aly** and was held at Grand Hyatt Cairo Hotel. The event was conducted as a part of SECC strategic objective to provide SPI services to the Egyptian community using new techniques and methodologies.

The speaker was **Dr. Radouane Oudrhiri**, who is the CTO of **Systonomy**, a consultancy company specialized in Software Quality Engineering and Six Sigma for ICT and Software. He has more than 20 years of teaching and consulting experience in software quality engineering, process improvement and Six Sigma/DFSS for software. Radouane has an M.S. in Operation Research from ENSAE Paris and MBA and Ph.D. in information systems and Decision from ESSEC Paris. Radouane is a Software Six Sigma Master Black Belt trained by GE. He is an expert in software architecture, design and quality engineering and sits on the board of multiple technology companies. He is a visiting lecturer in software engineering at Kingston University, Telecom Paris, University of La Sorbonne Paris. He is also an evaluator at the European Commission on the Software Process Improvement Experience and Mobile Technologies.

The event was a full-day event. The speaker introduced many topics such as:

1. Six Sigma – Defined
2. The Software Business
3. Six Sigma for Software
4. Six Sigma Project in software and IT
5. Defect Containment and/or Management
6. Inspection and Review
7. Requirements Engineering and Management
8. Relationships with other Frameworks

The tutorial presentation, examples and open discussions handled many aspects such as:

1. How and why Six Sigma is different methodology for business process improvement
2. The special features of the software industry that should be taken into consideration when applying Six Sigma in software companies
3. Using Six Sigma as an umbrella for CMMI, TSP/PSP and ITIL through providing the required link between technical and business layers in the IT and SW organization

From CMMI ML2 to ML3 in One Year! Part2: Extension of ML2 Practices

By: Ahmed Abd El Aziz

This is the second article that explains the difference between ML2 and ML3 to help management to decide if they can go from ML1 to ML3 directly in one year or they have to move to ML2 first then to ML 3. The previous article addressed the different expectations in SCAMPI A appraisal when appraising ML2 process areas in an appraisal to achieve ML2 or in another appraisal to achieve ML3.

This article will focus on another area. Some practices in ML3 process area could be considered as extension or elaboration to some practices in ML2. They give them more depth and add more value. To satisfy these ML3 practices, the company staff does not need to do additional activities or generate more documents. They actually need – in most of the cases – to change the way they were doing the old activities or extend them a little bit.

This article will explore some of these activities and how they are elaborated or extended in ML3.

Requirements Gathering

ML2 focuses on collecting the customer requirements. ML3 has more proactive approach. Customer requirements must also be elicited. Elicitation means identifying additional requirements not explicitly provided by the customer.

What if the functions the customer asked could be done in many ways; which way he needs? What if this line of business has many standards; which standard to follow?

Let's have some examples to explore the idea. In the first example the customer needs a financial application where it has a tax calculation function. One of the customer requirements could be "Calculate tax". The requirement is collected – ML2 satisfied. But many questions need to be answered. According to which law should the tax be calculated? If the customer is targeting to sell or use his product in some specific country, this country must be identified. The requirement may change to "Calculate Tax according to the Egyptian law" for example.

If the customer is targeting to sell or use it in many countries, these countries must be identified and a new requirement will appear; "The system must enable changing the law according to which the tax is calculated". Even this requirement needs more elicitation. Is this change required at the installation time or at the run time? If the customer is some governmental organization where it installs the system in one country and calculates tax according to that country law only, then the requirement may change to "The system must enable changing the law according to which the tax is calculated at the installation time".

If the customer is an international accounting organization that has a central location and offers services to customers from different countries, then the requirement may change to "The system must enable changing the law according to which the tax is calculated at the run time".

Is there more? Yes. What if the tax calculation law changes; should the customer be able to change the tax calculation mechanism in the system, or he has to go back to the development company? If he has needs to be able to change, a new requirement appears; "The administrator must be able to change the tax calculation mechanism".

One more example, if the customer is asking for a Learning Management System; should it follow some standard like SCORM, or not? If yes, there is one more requirement; "The system must be compatible with SCORM standard" which may be was not explicitly required by the customer.

Requirements Validation

ML2 requires that commitment to the requirements is obtained from project participants. In ML3, requirements must be validated to ensure that the resulting product will perform as intended in the user's environment. So in addition to obtaining commitment from the project team on the requirements (commitment from developers means they can implement the requirements; commitment from testers means they can test the requirements; …) a prototype may be prepared and tested on the customer environment and the requirements must be approved from the customer.

Work product Reviews

When it comes to reviews we find that ML2 mentioned two things; requirements and plans. Inconsistencies between the requirements and work products must be identified and all the project plans must be reviewed. Reviewing any other work products is not required. Even the review process itself is not expected to be formal. An email with the review defects with another email showing solving the defects is enough to prove the review. No need to analyze the review results; just fix the defects.

In ML3, the organization must identify which work products it will review and how they will be reviewed. It is expected that all the major work products will be reviewed like analysis, design, code, and test cases. It is also expected that this review is a formal one – at least for some work products – using a checklist and one or more meetings. The results of the reviews must also be analyzed. This could be a Pareto analysis to identify which causes make the most defects, phase injection analysis to identify which phase causes the most defects, actual number of defects compared to expected number of defects, average time to solve defects …

Training

In ML2, training is required for all process areas. Even if the organization staff members have a good technical experience, they need at least training on the processes they will use.

In ML3 the organization has to go far beyond this. The training must be planned. The relationship between the training and the organization's business objective and needs must be clear. This applies for all process training and technical training.

For example if the management decided to use some content management system in the new projects expected during the next three years and the existing staff has no enough experience with this system, this need (e.g. Establish experience in the content management system) is expected to be well documented and the training that will be offered that satisfy this need (e.g.

Train all developers and testers on the content management system during the first quarter of 2008 in a professional training center to the advanced level) is planned and the relation between the training and the need is well documented.

Project Management

In ML2, every project must have a plan that consists of estimation, schedule, risks, data management … However, neither use of historical data or building historical data from projects is not mandatory. In the previous article in the Issue 15 we referred to using the historical data in estimation and how it differs in ML2 and ML3.

In ML3 there is a major change. Projects must use experience of other performed projects in the organization in their planning. To achieve this; two major activities must be done.

- First at the planning phase of any new project, the organizational process assets (estimation repository, the organizational risk database, the lessons learnt of other projects …) must be reviewed and used appropriately to prepare the project's estimation, risks, and other planning outputs.

- Second at the end of the project or at the project milestones, the organizational process asset library must be updated with the actual results of the project.

This is the most critical point in ML3. Every thing else needs additional effort only to be achieved. This point needs – in addition to extra effort like others – a relatively long time to be achieved. Time is needed for some projects to work, contribute to the organizational process asset library, and finally other projects show the usage of the organizational process assets in their planning.

Risk management

In ML2 risks must be identified, analyzed, and monitored. Is not that all about risk management?

Actually no, there is still more to be done in ML3. Before identifying the project risks, the existence of organizational risk management guidelines is expected which identifies the risk taxonomy (risk sources and categories), a list of the risk management parameters and their meaning and usage, and the risk management strategy to be used in the projects.

In addition to risk identification and analysis required in ML2, mitigation plans must also be prepared for the most important risks. These plans must be executed and monitored. Mitigation planning and their execution must comply with the defined risk management strategy mentioned above.

Conclusion

Many practices in ML3 could be seen as elaboration and extension to some other practices in ML2. These practices require change in the way the work is done more than requiring new activities to be performed.

The table in the beginning of the following page summarizes the activities mentioned in this article showing what is required in ML2 and what the additional requirements in ML3 are.

	ML2 Requirements	**Additional ML3 Requirements**
Requirements Gathering	Requirements are collected from the customer and are understood.	Requirements that are not explicitly mentioned by the customer are identified, discussed and documented.
Requirements Validation	Project team must give commitment to the requirements.	Requirements must be validated and approved from the customer. A prototype may be prepared to help in validation.
Work Product Reviews	Requirements and plans must be reviewed. These reviews do not need to be formal ones.	More work products are expected to be reviewed. Some of these reviews are expected to be formal.
Training	The organization team must be trained at least on the processes in place.	The training needs must be identified. Training must be planned.
Project Management	The project management plan must be prepared.	Historical data must be used in planning the projects. Projects must update the historical database with their actual results.
Risk Management	Risks must be identified, analyzed, and monitored.	Risk guidelines and risk taxonomy must exist. Most important risks must be mitigated.

Biography

Ahmed Abd El Aziz has over 11 years of experience in the software industry. During that period he worked as developer, project manager, department manager, and process improvement leader. He joined SECC in the last year as a Senior Software Process Improvement Engineer. During that period he offered consultation on CMMI to many software development and implementation companies to maturity levels two and three. He also participated in many SCAMPI A appraisals beside many pre-appraisals and gap analysis activities. Ahmed is certified PMP since May 2005 and is Candidate SCAMPI A Lead Appraiser and Candidate SCAMPI B & C Team Leader since April 2007 and is expected to be Authorized SCAMPI A Lead Appraiser by the first quarter of 2008.

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:
aaziz@mcit.gov.eg

Software Reuse, an Integrated Approach

By: Ahmed Hammad

Software reuse is a comprehensive issue; without taking into account all perspectives, it could fail. Here we are trying to look at different angles and highlight several issues.

Business View

First, senior management should take reusability into consideration. Loading technical team alone with the reusability efforts will not normally succeed. It should start from top management. As opposed to seeing the business as developing a certain product for some customer, senior management should look at the business as developing an application family to satisfy different client's needs. The business focus here will affect all efforts after it.

As example, HP shifted their thinking from developing a new printer driver for each new printer to developing a reusable family of drivers. The result was a huge saving in cost and improved quality.

Requirements

After focusing on a new business perspective regarding what is really required, reusing is shifted to create requirements with focus on reusability. Requirements even could be restated to match certain reusable components features. Requirements analysis phase will follow by creating use cases focused on variability points.

The following example will explain the concept of determining variability points:

Vacation Request:

```
1. Employee requests vacation from his direct manager
2. Direct Manager retrieves balance from Human Resources
If there is no balance
        3. The vacation request is rejected
If vacation is more than 3 days, Direct Manager forward to Senior Manager
        4. Senior Manager approves or rejects the vacation request
If vacation is more than 3 days
        5. SM needs to receive email notification
6. Direct manager approves or rejects the vacation
```

In some companies they don't escalate to senior manager the vacation for whatever reason. The Direct manager always has permission to approve or reject the vacation request. We will mark variability with curly brackets {}.

```
1. Employee requests vacation from his direct manager
2. Direct Manager retrieves balance from Human Resources
If there is no balance
        3. The vacation request is rejected
If {StrictSMFollowUp}
        If vacation is more than [ApproveLimit] days,
                4. Direct Manager forwards to Senior Manager
                5. Senior Manager approves or rejects the vacation request
```

If vacation is more than {NotificationLimit} days
6. SM needs to receive email notification
7. Direct manager approves or reject the vacation

The *StrictSMFollowUp* is variable; it could be true or false based on the way the organization work. In our example, we classified the businesses that will use our application, some have strict SM follow-up, and others have not.

ApproveLimit states the max number of days Direct Manager allowed to approve without consulting Senior Manager. *NotificationLimit* states the max count of days that need no notification.

The above is just an example to identify variability points in the requirements. We can use many ways to mark variability in the requirements that makes sense for our own applications.

Variability Implementation

- *Inheritance:* As an example, in a typical application we wanted to support different database servers. You can select the basic reference dbase and create classes to implement it. Each other new dbase could inherit from the base classes and override essential operations. As the basic SQL construct are the same between databases, you should centralize the variability in specific classes.

- *Parameter:* You can have pages or screens with customization parameters. Administrators have the permission to customize the system. As example, the timeout of waiting the response from the external bank server.

- *Configuration:* In many cases you can use extensive configuration files to adapt the behavior of the system. In some systems you can even specify a complete alternate component through the configuration file. As example, specifying another XML parser. In this case, your application should use standard XML parser interface and the new parser should implement the same interface.

Architecture and Design:

Building the architecture and design in a way that makes configurability and reusability easier is an important issue. Layered architecture and components based design make it easier for you to achieve better designs.

Focusing on using known architecture and design patterns will make it easier for you to build state of the art architecture and design. Explaining patterns is outside the scope of this article, but some references will be listed in the references section.

Reusable Asset Library:

Having a reusable assets library with processes that define how to reuse a certain component, how to add new component and how to update existing component is important of reuse success.

A traceability table that traces each component version with which products are using it is required to guide in updating the applications with new components updates if necessary. Asset library should be organized so that each one item should have:

- Version number. A unique version number to the library item.
- Status: is it ready for reuse or still in beta state.
- Bin reusable component (if exist) such as DLL files or OCX files.
- Documentation: A simple document that specify how to use. A sample code will help component users to use it effectively.
- Release notes: This can specify what bugs are solved, new features, or knows issues compared to the earlier version of the same component.
- Source code: if applicable.

Process Support

We will describe basic processes to help managing the reusable asset library. We will not state the details of each one.

- **Add new reusable code item:** The steps needed and standards that should be satisfied in the new reusable code item.
- **Delete reusable code item:** Remove a typical item as it is no longer used nor supported.
- **Update reusable code item:** A new revision is created and needs to be available in the repository.
- **Use typical reusable code item:** reusing should be organized to know the full dependencies between reusable code items and applications that use it.
- **Requirements development process** should have activities for requirements development to guide in stating techniques and methods to state the variability points.
- **Architecture and design process** should be stating using typical techniques to help focusing on reusability.
- **Testing processes** should take into account testing variability points comprehensively.

Before adding a new version to the library, an extensive test and peer review is done to make sure the component is meeting quality requirements.

A common challenge is what to do with a large set of reusable code, should we create a project to organize them and add them to the new empty asset library? Or just focus on the new reusable components?

Without first closing the sink that continuously diminishes reusable code, we will always have the case of old code that need cleaning to be added to the reusable code repository. So first establish a system to close all waste holes and in parallel clean up and restructure old code.

Conclusion

Whatever your company size and experience are, you can not simply order reuse or install it at once and expect it to happen. Rather, you should encourage reuse by following many steps as described in this article. Although reusability conceptually is a simple idea, many efforts should be done to realize its benefits. This article was an attempt to highlight many of these issues.

References

[1] Software Reuse, Architecture, Process, and Organization fir Business Success by Jacobson, Martin Griss, and Patrick Jonson

[2] Design Patterns, GOF, by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.

[3] Pattern Oriented Software Architecture, a system of Patterns, Volume 1 by Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sornmerlad, and Michael Stal.

[4] Patterns of Enterprise Application Architecture, By Martin Fowler, David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, Randy Stafford.

Biography

Ahmed M. Hammad is a Senior Software Process Improvement Engineer in SECC. Before joining SECC, he was working as a software manager in the Research & Development Dept. in the Egyptian Telephone Company "QuickTel". He has experience in software development and management for more than 13 years, in desktop, web based and embedded development.

Feedback Contacts

Feedback, comments and questions are appreciated by the author.

Email:
ahammad@mcit.gov.eg

SW Process Improvement Traps

By: Sally Hassan

Surviving in the increasingly competitive software business requires more than hiring smart, knowledgeable engineers and buying the latest development tools. You also need to use effective software development processes.

During software process improvement journey, organizations face some common traps that can undermine a software process improvement program. Organizations need to learn about these process improvement killers to be able to avoid them.

In this article, we are going to present ten common traps and the best practices to avoid them. These ten traps are:

- Lack of management commitment
- Unrealistic management expectations
- Time-Stingy project leaders
- Delay on action plan implementation
- Achieving a CMMI level becomes the primary goal
- Inadequate training is provided
- Expecting defined procedures to make people interchangeable
- Failing to scale formal processes to project size
- Process improvement becomes a game
- Process assessments are ineffective

A detailed description of these traps symptoms and solutions is provided in the following paragraphs.

Trap #1: Lack of Management Commitment

Symptoms: Senior managers (and/or middle managers) may not really be willing to make short-term sacrifices to free up the resources required for the long-term investment.

Solutions:

- Executives must be educated about the costs, benefits, and risks so they will have a common vision and understanding of this complex area of software development.

- Commitments need to be aligned along the organizational hierarchy, so that managers are not working at cross purposes.

Trap #2: Unrealistic Management Expectations

Symptoms: Excessive enthusiasm by ambitious managers can also pose risks to the improvement program. If the goals, target dates, and results expected by managers are not realistic, the software process improvement effort is ultimately set up for failure.

For example, a manager may hope to solve current staff shortages by driving the organization to reach CMMI® Level 2, which typically leads to higher software productivity and quality.

Solutions:

- Educate your managers to help them to understand the realities of what a serious process improvement initiative will cost and what benefits they might expect.

- Collect data from the software literature on results that have been achieved by other companies with effective improvement programs and the investments those companies made over a specified time period.

- Use data available from the software literature or from other areas of your own company to help your managers develop realistic expectations and set reasonable, even ambitious, goals.

Trap #3: Time-Stingy Project Leaders

Symptoms: Successful software process improvement initiatives require project leaders to adjust their project schedules to permit team members to devote some time to improvement activities. A project leader who claims to believe in software process improvement but in practice treats it as a burden added on top of the project activities is sending conflicting signals to team members.

Solutions:

- You need to have consistent, active commitment at all stages of management; a bottleneck anywhere in the organizational hierarchy can bring the software process program to a screeching halt. One way to achieve consistency is through an interlocking management commitment process as a corporate or organizational policy.
- Top managers publicly state their goals and priorities (including software process improvement), and people at the lower management levels write their goals and priorities to support those of their superiors.
- Senior management must make it clear that project leaders will be evaluated on the effectiveness of their process improvement activities, as well as on the success of the software projects themselves.
- Software project planning needs to account for the staff resources that are being devoted to design and implement the new software processes.

Trap #4: Delay on Action Plan Implementation

Symptoms: The lack of progress on improvement plans is frustrating to those who actually want to see progress made, and it devalues the investment of time and money made in the process assessment itself.

Solutions:

- A good way to turn action plans into actions is to treat improvement activities as mini-projects.

- Concentrating on just two or three improvement areas at a time avoids overwhelming the project team.

- You need to measure progress against the plans, and to measure the impact of each action plan on the business results achieved.

- Project leaders must report the status of their action plans every three months, to a multilevel management steering committee.

Trap #5: Achieving a CMMI Level Becomes the Primary Goal

Symptoms: Organizations that adopt the CMMI framework for process improvement risk viewing attainment of a specific CMMI maturity level as the goal of the process improvement, rather than as one mechanism to help achieve the organization's real business goals.

Solutions:

- In addition to aiming at the next CMMI level, make sure your software process improvement effort is aligned with corporate business and technical objectives.

- Engage the process improvement activities with any other improvement initiatives that are underway, such as ISO 9001 registration, or with an established software development framework already in use.

- Recognize that advancing to the next CMMI maturity level can take one to three years. It is not feasible to leap from an initial ad hoc development process to a super-sophisticated engineering environment in one fell swoop.

- Elaborate that your goal is not to be able to chant, "We're Level 5! We're Level 5!" Your goal is to develop improved software processes and more capable development engineers so that your company can prosper by offering higher quality products to your customers more efficiently than before.

- Use a combination of measurements to track progress toward the business goals as well as measure the progress of the software process improvement program.

Trap #6: Inadequate Training is provided

Symptoms: Inadequate knowledge can lead to false starts, well-intentioned but misdirected efforts, and a lack of apparent progress. Without training, the organization's members will not have a common vocabulary and understanding of how to assess the need for change or how to interpret specialized concepts of the improvement model being followed, such as the CMMI.

Solutions:

- Training to support established process improvement frameworks can be obtained from various commercial sources (such as process improvement consultants or training vendors), or you can develop such training yourself.

- Understand that different participants in the software process improvement activities will need different kinds of training.

- Use commercial sources of training wherever possible. This way you avoid having to create all of your own training materials.

Trap #7: Expecting Defined Procedures to Make People Interchangeable

Symptoms: Managers who have an incomplete understanding of the CMMI may expect that having repeatable processes available (CMMI Level 2) means that every project can expect to achieve the same results with any set of randomly assembled team members.

Solutions:

- Explain that individual programmers have been shown to have a 10-to-1, 20-to-1, or even higher range of performance (quality and productivity) on software projects. Process improvements alone can never equalize such a large range of individual capability.

- Never underestimate the importance of attracting, nurturing, and rewarding the best software engineers and managers you can find.

- Aim for software success by creating an environment in which all team members share a commitment to quality and are enabled—through superior processes, appropriate tools, and effective team interactions—to reach their peak performance.

Trap #8: Failing to Scale Formal Processes to Project Size

Symptoms: A small organization can lose the spirit of the CMMI (or any other process model) while attempting to apply the model to the letter, introducing excessive documentation and formality that can actually slow down project work. This undermines the credibility of software process improvement, as team members look for ways to bypass the official procedures in an attempt to get their work done efficiently.

Solutions:

- The process definitions your group develops should be no more complicated or elaborate than they need to be to satisfy CMMI goals.

- Strive for a practical balance between documenting procedures with enough formality to enable repeatable project successes, and having the flexibility to get project work done with the minimum amount of low-value overhead effort.

- Your process improvement action teams should provide a set of scaleable processes that can be applied to the various sizes and types of projects your group undertakes.

Trap #9: Process Improvement Becomes a Game

Symptoms: The focus is on making sure a process audit is passed, rather than on really changing the culture of the organization for the better.

Solutions:

- Focus on meeting organizational and company objectives with the help of improved software processes.

- Do not simply try to conform to the expectations of an established framework like the CMMI. It is not enough to simply create documented procedures to satisfy the letter of some improvement framework; you must also satisfy the spirit of the framework by actually following those procedures in your daily project work.

- As a process change leader, identify the behaviors you would expect to see throughout the organization in each improvement area if superior processes are successfully internalized and institutionalized.

- As a manager, your group members need to understand that you are serious about continually striving to improve the way they build software; the old methods are gone for good. Continuous improvement means just that, not a one-shot game we play so that someone's checklist can be filled in properly.

Trap #10: Process Assessments are Ineffective

Symptoms: If process capability assessments (often led by the EPG) are conducted without adequate participation by the software development staff, they turn into audits. This can lead to a lack of commitment, buy-in, and ownership of the assessment findings by the project team.

Solutions:

- Include a free-form discussion with a representative group of project team members as part of the assessment process whenever time permits.

- Use your EPG to actively facilitate the change efforts of your project teams, not just to audit their current process status and report a long, depressing list of findings.

- Identify process liaisons or champions in the project teams to augment the assessment activities of the process group.

Biography:

Sally Hassan Mostafa is a Senior Software process improvement consultant at SECC. She got her B.Sc. in Computer Engineering from the Faculty of Engineering, Ain Shams University, Cairo (Egypt). She has over 12 years of experience in software engineering, project management, and management. She provided consultation for process improvement initiative based on CMMI. Participating in the projects for process improvement in Egypt helping the companies in getting CMMI certification through gap analysis, consultation visits and formal SCAMPI appraisal class "A".

Feedback contacts

Feedback, comments and questions are appreciated by the author.

Email:
sallyh@mcit.gov.eg