



# Atlanta SPIN

Software & Systems Process Improvement Network

# Newsletter



Barbara Carkenord

## Developing Requirements for Purchased Software

**Presentation by Barbara Carkenord**  
**Article Written by Abi Salimi**

Barbara started the talk by asking the participants to put the names of the COTS (Commercial Off the Shelf) packages which their organizations were using on the flipchart papers posted on the walls. Many attendees participated in this activity and some of them shared the functionality of their packages with the audience.

An interesting observation was that there were varieties of packages which organizations were using. Many of these packages were not stand alone and had an interface with other systems. Also, very few of such packages were used as they were purchased and needed to be customized.

The importance of COTS packages is that about 99% of the U.S. companies in own them, and lots of money is spend in purchasing and maintaining such packages.

*Article continues on page 2*

Inside This Issue	
Developing Requirements for Purchased Software	1-2
Next Meeting	1
A History of the Capability Maturity Model for Software— Part 1	3-5
Sponsor Information	6
About Atlanta SPIN	7

### Next Meeting: October 20<sup>th</sup> 2010

Pragmatic Lean-Agile for Complex Systems Development

Time: 6PM  
 Location: LaQuinta Inn & Suites  
 6260 Peachtree Dunwoody  
 Atlanta, GA 30328



There are usually many gaps between the business needs and what the COTS packages deliver. Some reasons are that requirements are not identified well ahead of time and nobody evaluates these packages appropriately to make sure they address the requirements.

To deal with these challenges, Barbara talked about the following phases of COTS projects:

1. Determine the project objectives, scope and elicit business requirements
2. Evaluate and select a package
3. Assess the package for fit and determine if customization is necessary (write requirements for customization)
4. Oversee and validate customizations (with project management and quality assurance)
5. Develop the implementation plan
6. Post implementation assessment of success
7. Maintenance

COTS packages are different from the software built in-house. COTS are built to address the needs of a large number of users, not necessarily the specific needs of a particular user. COTS architecture may not work in the organization's environment (e.g., a software package which a company had purchased did not run on the company's network. A simple question, if were asked earlier, could have saved the company lots of money.) The other thing different is that it is not much fun for the developers to work on such packages, because most of the design is already done.

From a Business Analyst (BA) perspective, determine the COTS project objectives and scope, and elicit business requirements. Understand what the problem is and what is to be accomplished before a solution is selected. Come up with measurable objectives from business point of view (e.g., increase revenue, decrease cost, increase customer service, etc.)

Use the context diagram to figure out all the people, organizations and systems impacted by the package.

Create an objective evaluation process and rank the packages based on such objectives. Remember that vendors are good at making their packages look much better than they are. Assess the vendor through the Better Business Bureau, calling the vendor's customers, checking vendor's financial stability, etc. Select the best package, asses it, and customize it, if necessary.

Oversee and validate customizations and develop the plan for the implementation of the COTS package. Before customizing, consider cost, functionality and future package upgrades. See, also, who will do the customization-The vendor, 3<sup>rd</sup> party, or you. If there is too much customization, future upgrades could be expensive. Make sure knowledgeable and experienced developers are involved in customization.

Do Post-implementation assessment of success in about 3-6 months after implementation. Talk with business people to find out how it is going. The first few days of implementation is not the best time to ask the question. The last phase is maintenance. This phase is the most expensive and the longest phase. Must talk about this right at the beginning of the project.

Toward the end of the talk, there was a discussion of different types of gap analyses such as the ones identifying the differences between AS IS situation with potential TO BE; business needs vs. application solution features; manual vs. automated interfaces; organization's terminology vs. the ones used in the package; data in the old system vs. the ones used in the package; old and new workflows; etc.

The talk was concluded by a lively Q&A period.

# A History of the Capability Maturity Model for Software—Part 1

Article Written by Mark C. Paulk, Carnegie Mellon University

*Editor's Note: This is the first in a multi-part article taken with permission from ASQ's Software Quality Profession Vol. 12, No.1, Dec. 2009*

## **INTRODUCTION**

In August 1986, the Software Engineering Institute (SEI) at Carnegie Mellon University, with assistance from the MITRE Corporation, began developing a process maturity framework that would help organizations improve their software processes (Humphrey 2002). This effort was initiated in response to a request to provide the federal government with a method for assessing the capability of their software contractors. In June 1987, the SEI released a brief description of the software process maturity framework (Humphrey 1987a) and, in September 1987, a preliminary maturity questionnaire (Humphrey 1987b). Based on experience in using the software process maturity framework and the maturity questionnaire for diagnosing problems and improving processes, the SEI formalized the concepts as the Capability Maturity Model for Software (Software CMM). Version 1.0 of the model was published in 1991 (Paulk et al. 1991; Weber et al. 1991). Version 1.1 was released in 1993 (Paulk et al. 1993a; 1993b) and published as a book in 1995 (Paulk et al. 1995a). The Software CMM has now been retired in favor of the CMM Integration (CMMI) model, but it inspired many other standards and frameworks, including the People CMM (Curtis, Hefley, and Miller 1995), the Systems Engineering CMM (Bate et al. 1994), and the Systems Security Engineering CMM (Hefner 1997).

It was also one of the drivers in the development of ISO/IEC 15504 (Process Assessment), notably Part 7 on the assessment of organizational maturity (ISO 2008). The Software CMM had an enormous impact on the software community. It is estimated that billions of dollars were spent on CMM-based improvement (Emam and Goldenson 1999). Many case studies and research reports have been published on its impact on productivity, cycle time, and quality (Krasner 2001; Clark 2000; Harter, Krishnan, and Slaughter 2000), and its maturity levels have been embedded in parametric cost models such as COCOMO II (Boehm 2000). This article discusses the development of the Software CMM and the critical decisions made as it evolved; it is an update of a previous description of the evolution of the model (Paulk 1995c) that bridges to its successor, the CMMI for Development (Chrissis, Konrad, and Shrum 2006).

## **PRECURSORS TO THE SOFTWARE CMM**

The basic premise underlying the SEI's work on software process maturity is that the quality of a software product is largely determined by the quality of the software development and maintenance processes used to build it. The staged structure of the software process maturity framework is based on total quality management (TQM) principles that have existed for nearly a century. The work of Frederick Taylor and Frank Gilbreth on "scientific management" and time-and-motion studies in the early 1900s eventually led to the new discipline of industrial engineering (Hays 1994).

In the 1930s, Walter Shewhart, a physicist at AT&T Bell Laboratories, established the principles of statistical quality control (Shewhart 1931). These principles were further developed and successfully demonstrated in the work of such authorities as W. Edwards Deming (1986) and Joseph M. Juran (1988).

In recent years, the TQM concepts have been extended from manufacturing processes to service and engineering design processes. The software process can be defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products. As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization. This, in turn, leads to higher-quality software, increased productivity, less rework, and improved software project plans and management. This is an adaptation of the Deming chain reaction shown in Figure 1. Perhaps the most important point to note in this chain reaction is that process improvement and quality management should affect business objectives. The Software CMM adapted TQM principles for software organizations. The model establishes a project management and engineering foundation for organizational learning and evidence-based management. Watts Humphrey adapted Philip Crosby's quality management maturity grid, described in *Quality Is Free* (Crosby 1979), for his process work at IBM (Humphrey 2002), and the maturity structure inspired the software process maturity framework that became the Software CMM.

#### Crosby's Quality Management Maturity Grid

Crosby describes five evolutionary stages in adopting quality practices. The quality management maturity grid applies five stages to six measurement categories in subjectively rating an organization's quality operation. The five stages are:

Uncertainty: Management is confused and uncommitted regarding quality as a management tool.

Awakening: Management is beginning to recognize that quality management can help.

Enlightenment: The decision is made to really conduct a formal quality improvement program.

Wisdom: The company has the chance to make changes permanent (things are basically quiet and people wonder why they used to have problems).

Certainty: Quality management is considered an absolutely vital part of company management.

The six measurement categories are:

1. Management understanding and attitude: Characterized as "no comprehension of quality as a management tool" at uncertainty and "an essential part of the company system" at certainty.
2. Quality organization status. Characterized as hidden at uncertainty and a thought leader/main concern at certainty.
3. Problem handling. Fought when they occur at uncertainty and prevented at certainty.
4. Cost of quality as percent of sales. Characterized as 20 percent at uncertainty and 2.5 percent at certainty.

5. Quality improvement actions. Characterized as no organized activities at uncertainty and a normal and continued activity at certainty.
6. Summation of company quality posture. Summarized as “we don’t know why we have problems with quality” at uncertainty and “we know why we do not have problems with quality” at certainty.

#### Radice’s IBM Work

Ron Radice and his colleagues at IBM, working under the direction of Watts Humphrey, identified 12 process stages, which were characterized by 11 attributes measured on a five-point scale (Radice et al. 1985). The process stages are stages in the life cycle: requirements, product-level design, component-level design, module-level design, code, unit test, functional

verification test, product verification test, system verification test, package and release, early support program, and general availability. The 11 attributes include process, methods, adherence to practices, tools, change control, data gathering, data communication and use, goal setting, quality focus, customer focus, and technical awareness. The five-point scale consists of traditional, awareness, knowledge, skill and wisdom, and integrated management system. Humphrey brought these concepts to the SEI in 1986 and used them in defining the software process maturity framework.

*Part 2 of this article will be in the next Atlanta SPIN Newsletter.*

<b>Atlanta SPIN Board of Directors</b>	
Name	Role
Stephen Burlingame	President Sponsorship Chair
Bill Reister	Vice-President Programs Chair
Vivian Viverito	Secretary Technology Chair
Mike Sweeney	Treasurer
Scott Banks	Director Membership Chair
Fred Haigh	Editor-in-Chief Newsletter
Abi Salimi	Director At Large

To contact the Atlanta SPIN Newsletter Editor email:  
newsletter@atlantaspin.org

### Atlanta SPIN Sponsors

A special thanks to all of our sponsors, they make our efforts possible!

#### **Platinum Sponsors**



#### Monthly Raffle Sponsors

Atlanta SPIN would like to thank the following companies for donating to our monthly book raffle:



**SEI Webinar Series. Register Today! Free to all!**

 **Software Engineering Institute** | Carnegie Mellon

 **SPIN** Software and Systems Process Improvement Network

Go to <http://www.sei.cmu.edu/library/webinars.cfm> for more information about upcoming SEI webinars

Atlanta SPIN is proud to work closely with the following organizations and companies:



---

## **Process Improvement Websites**

### **Software Engineering Information Repository**

<https://seir.sei.cmu.edu/seir/>

This site has over 30,000 registered users and is a forum used to contribute and exchange information about software engineering improvement activities, including CMMI.

### **The CMMI Process Improvement Yahoo! discussion group**

[http://groups.yahoo.com/group/cmml\\_process\\_improvement/](http://groups.yahoo.com/group/cmml_process_improvement/)

A forum used to contribute and exchange ideas about CMMI-based improvement.

### **BSCW Shared Workspace**

<https://bscw.sei.cmu.edu/pub/bscw.cgi/0/79783>

A Forum used to contribute and exchange CMMI related materials

---

## **About Atlanta SPIN, Inc.**

[www.atlantaspin.org](http://www.atlantaspin.org)

Atlanta SPIN is a 501(c)(3) non-profit group that is a forum for the open exchange of ideas and experiences related to software and systems process improvement. Founded in 1994, Atlanta SPIN is a local SPIN chapter with the backing of the Software Engineering Institute at Carnegie Mellon University. Our monthly meetings highlight a speaker with expertise in a topic related to software and systems engineering process improvement. Atlanta SPIN provides a valuable educational resource for Atlanta's business community. For more information, please visit our website.

