**Presentation Abstracts – Mark C. Paulk**

*Abstracts* - Presentations can range from a one-hour talk to a full-day tutorial. Note that tutorials can be offered in shorter forms but not necessarily vice versa. The first section of this list of abstracts describes shorter presentations; the second tutorials and one-day seminars.

Dr. Mark Paulk  
University of Texas at Dallas  
ECSS 3.610, EC-31  
800 W. Campbell Road  
Richardson, TX  75080  
Email: Mark.Paulk@ieee.org  –or–  Mark.Paulk@utdallas.edu  
Website: [http://utdallas.edu/~mcp130030/](http://utdallas.edu/~mcp130030/)

The presentations in this section are recommended for one-hour talks

**Investing in Software Process Improvement: An Executive Perspective**

This presentation discusses some of the empirical evidence on the value obtained from investing in software process improvement using the Software Capability Maturity Model (Software CMM) or CMM Integration (CMMI) for Development developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. Business drivers for process improvement and some challenges in organization change are described, along with data on the impact on cost, schedule, and quality of achieving the five maturity levels in CMM(I). From an executive perspective, the crucial point is that continual improvement depends on systematically addressing the problems facing the organization -- regardless of the improvement framework selected. This "constancy of purpose" depends on management sponsorship, support, and investment.

**Software Process Improvement in Small Organizations and Small Projects**

Process improvement frameworks, such as the Software CMM and CMMI for Development, have been criticized for being applicable only to large organizations and projects, yet many small organizations and projects have used these models successfully in their process improvement efforts. Some of the common problems with interpreting process frameworks for the small project/organization include:

- What does “small” mean? In terms of people? Time? Size of project? Criticality of product?
- What are the CMM “requirements”?
- Are there key process areas or goals that are “not applicable” to small projects/organizations?
- Are there “invariants” of good processes?
The conclusion of this presentation is that the issues associated with interpreting process frameworks for the small project/organization may be different in degree, but they are not different in kind, from those for any organization interested in improving its software processes. Using these frameworks effectively and correctly requires professional judgment and an understanding of how they are structured to be used for different purposes.

**Agile Methods and Process Discipline**

Agile methods have been touted as the programming methodologies of choice for the high-speed, volatile world of Internet and Web software development. They have also been criticized as just another disguise for undisciplined hacking. The reality depends on the fidelity to the agile philosophy with which these methodologies are implemented and the appropriateness of the implementation for the application environment. Extreme Programming, Scrum, and similar agile methods are disciplined processes that incorporate good engineer and management practices, albeit with extreme implementations tailored to a specific kind of environment. Many of the challenges to agile methods arise from considering how they would fit in different environments; the degree to which agile methods can be adapted without losing their emergent properties is passionately debated. The compatibility of agile methodologies with process improvement frameworks, such as the Capability Maturity Model for Software and CMMI for Development, is summarized and critiqued, with the conclusion that appropriately implemented agile methods can be useful additions to an organization’s set of standard software processes.

**People Issues: The “Soft Side” of Software Engineering**

“Our greatest asset is our people.” This platitude is frequently followed by announcements of layoffs, downsizing, and similar Dilbertesque decisions. If people are the most important single factor in success, how do we incorporate that fact into our software projects and organizations? This tutorial provides an overview of “people issues” for software engineering, management, and process improvement from an individual, team, and organizational perspective. People issues for the individual include order-of-magnitude differences in individual performance and temperament differences such as those indicated by the Meyers Briggs Type Indicator. Team issues include those associated with establishing effective and high-performance teams. Organizational issues include decision making styles, organizational culture differences, and varying national cultures. For human-centric, creative work such as software organizations perform, the foundation for performance excellence is the competence of the people doing the work as enabled – or hindered – by the environment they work in.

**The Rational Planning of (Software) Projects**

The software crisis has persisted for decades. Our difficulties in planning and managing software projects may be rooted in fundamental human nature, as suggested by research in rational decision making, more than in the inherent difficulty of building software. Models such as the Capability Maturity Model for Software and CMMI for Development, which apply the concepts of Total Quality Management to software development and maintenance, embody one approach to
improving the software process. The problems addressed by both CMMI and TQM seem to lie in the basic ways that human beings think and organize themselves. In many circumstances, normal human decision making can be characterized as “irrational” because of systematic biases and fallacies in the way people make decisions. Mechanisms such as those suggested by the various software process frameworks support rational and informed decision making.

**Practices of High Maturity Organizations**

The Capability Maturity Model (CMM) for Software, developed by the Software Engineering Institute at Carnegie Mellon University, was a five-level model for improving the process capability of organizations, which has been widely adopted in the software community and beyond. It has now been superseded by CMM Integration (CMMI) for Development, which integrates systems and software Engineering. Although CMM(I) has been widely used, comparatively few organizations have achieved the higher levels of organizational maturity – Levels 4 and 5.

Conceptually the focus of Levels 4 and 5 is to use measurement – and an understanding of variation – to control and improve the process. This is usually viewed as applying the concepts of statistical process control (SPC) to the software process, but software processes are quite different from the manufacturing processes where SPC was developed. The lack of wide-spread experience with high maturity practices has been a challenge for assessors who wish to appraise Level 4 and 5 organizations and for those working to improve their organization’s processes. The purpose of this presentation is to discuss the engineering and management practices typically found in high maturity organizations that adequately address the statistical thinking in CMM(I).

Alternatives to measurement-driven decision making include broadening the scope of the improvement effort, e.g., systems engineering as exemplified in CMMI, systematically addressing the people issues, e.g., as exemplified in the People CMM, and addressing other business priorities, such as creativity and innovation.

**Mature Software Project Management**

Delivering software products with the promised functionality on budget and on schedule is a challenge many organizations have difficulty meeting. Over the past several years, a growing number of organizations have adopted the ideas of Total Quality Management for managing their software projects and systematizing organizational learning. Perhaps the best known model for applying TQM concepts to software organizations is the Capability Maturity Model for Software, which was superseded by CMM Integration for Development (CMMI). CMMI describes “what” engineering and management practices an organization should use, and it prescribes an improvement path that leads to mature processes and a mature organizational capability.

CMMI and similar process frameworks describe “what” without prescribing “how”. A variety of specific implementations can address the “how” of those “whats”. For management, traditional plan-driven project management versus critical chain project management versus agile project management may be considered. Selecting, hiring, and retaining competent professionals is always the foundation for successful projects. Risk management is critical for software projects. None of
these practices need be particularly innovative; for the most part they capture common sense practices that are unfortunately sacrificed all too often under pressure.

**Maturing Organizations, Evolving Project Management, and Increasing Capability**

This presentation begins with a brief discussion of the Capability Maturity Model for Software (the Software CMM) and its successor CMM for Integration (CMMI) for Development, but its primary focus is on the evolution of project management as an organization matures. Software project management changes to take advantage of the organizational learning implied by the organization’s moving up the maturity levels. The presentation closes with a discussion of the approach, deployment, and results associated with maturing organizations and a brief warning against abuses of the various process frameworks that may be used to guide improvement efforts.

**Lessons Learned in Process Modeling**

One of the powerful tools used by high maturity organizations is process modeling. Process models can provide insight into planning, strategic management issues, process control, operational management, process improvement, technology adoption, and training. Building useful and usable models is not, however, a trivial exercise. This paper summarizes some of the lessons learned in selecting a modeling technique, identifying the important measures to parameterize the model, collecting valid data to calibrate it, and applying the insights from the model effectively.

**Working with High Maturity Organizations**

As organizations attain the higher maturity levels according to the Software Engineering Institute’s Capability Maturity Model for Software, and its successor CMM Integration (CMMI) for Development, one of the challenges they face is to establish effective and efficient customer-supplier relationships. This presentation discusses the issues that for maturity organizations in working with low maturity customers and vice versa. For custom software development, both the customer and the development must have complementary degrees of maturity to enable the full power of high maturity practices.

**The eSourcing Capability Model for Service Providers**

Organizations are increasingly delegating their information technology (IT) intensive business activities to service providers, taking advantage of the rapid evolution of the global telecommunications infrastructure. The business processes being outsourced range from routine and non-critical tasks, which are resource intensive and operational, to strategic processes that directly impact revenues. Managing and meeting client expectations is a major challenge in IT-enabled outsourcing services and examples of failure abound. The eSourcing Capability Model for Service Providers (eSCM-SP) contains a set of 84 best practices that address the entire outsourcing (or insourcing) process. The eSCM-SP is structured along three dimensions: Sourcing Life Cycle,
Capability Areas, and Capability Levels. The Sourcing Life Cycle dimension addresses engagement initiation and completion activities as well as delivery and ongoing practices. Ten Capability Areas are used to logically group the Practices. Five Capability Levels address providing services, performing to meet client requirements, controlling through measurement, enhancing through innovation, and sustained excellence. The eSCM-SP aids IT-enabled outsourcing service providers in forming, managing and improving outsourcing relationships. In addition to describing critical sourcing issues and eSCM-SP version 2, this tutorial also describes some of the lessons learned by organizations adopting the Model.
The presentations in this section may be 1-hour presentations, or all the way up to full-day tutorials. Some of these are variants/expansions of the shorter presentations, e.g., “Practices of High Maturity Organizations” is almost completely included in “Understanding High Maturity,” and “Software Process Improvement With Good Judgment” includes the majority of the material in “Using the Software CMM in Small Organizations and Small Projects.”

**Software Process Improvement With Good Judgment**

Common concerns in interpreting the Software CMM, and now CMM Integration (CMMI) for Development, in different environments are discussed. General principles for good process definition and management are described and then illustrated with issues in interpreting process frameworks such as CMMI-DEV for small projects and organizations. Interpretation issues include: What does “small” mean? How do I define “project?” “Organization?” What are the model/standard “requirements”? Are there process areas or goals that should not be applied to small projects/organizations?

Professional judgment is necessary to use process frameworks correctly since the practices in frameworks such as CMMI-DEV are informative rather than normative, but some practices should always be judged necessary for adequate processes, while others are context sensitive. Using the a process framework without judgment leads to dysfunctional behavior; some of the drivers and motivations for abuse of CMMI-DEV are described.

The conclusion of this presentation is that the issues associated with interpreting process framework in various contexts may differ in degree but not in kind for any organization interested in improving its software processes. Using any process framework effectively and correctly requires professional judgment and an understanding of how the framework is structured to be used for different purposes.

**Trends in Software Process and Quality**

This presentation provides an overview of the current world-wide trends in the arena of software process and quality improvement. It begins with a discussion of the Capability Maturity Model for Software developed by the Software Engineering Institute and the state of the practice in software engineering. The shift to CMM Integration (CMMI) is described. Other approaches to software process improvement are briefly discussed, including the ISO 9000 family of quality management system standards, the ISO/IEC/IEEE 12207 standard for software life cycle processes, the ISO/IEC/IEEE 15288 standard for systems life cycle processes, and the ISO/IEC 15504 / 330xx standards for process assessment. These trends for software process and quality frameworks are described in the framework of performance excellence initiatives, such as Total Quality Management and Six Sigma.
Understanding High Maturity Practices

“High maturity” implies a superior process capability, with the expectation of continual, measured improvement, for a software organization, yet comparatively few organizations have achieved the higher levels of maturity against the Capability Maturity Model for Software (CMM) or CMM Integration (CMMI) for Development – Maturity Levels 4 and 5. The lack of wide-spread experience with high maturity practices is a challenge for both assessors and process engineers. The purpose of this tutorial is to discuss the fundamental principles of process management at the higher maturity levels and some of the effective engineering and management practices typically found in high maturity organizations. At the end of this tutorial, attendees should have a better understanding of how to interpret the Software CMM or CMMI-DEV at maturity levels 4 and 5 and insight into effective implementations of high maturity practices.

Basic Statistics for Software Engineers

As organizations improve their processes and use quantitative and statistical techniques to control and improve the way software is built, there is a growing need for software engineers to have a basic understanding of statistics. This tutorial provides an overview of software measurement, statistical concepts and terminology, statistical thinking, basic statistical tools, regression analysis, test of hypotheses, and control charts. It provides an awareness of statistical issues, including a discussion of assumptions that are made, pitfalls to be wary of, and myths and misconceptions to dismiss. This awareness should help the software professional understand the use and limitations of methods and tools that use statistics, such as cost models, statistical process control, reliability models, and rational decision making.

Considering Statistical Process Control for Software

The Capability Maturity Model for Software, developed by the Software Engineering Institute at Carnegie Mellon University, was a model for building organizational capability that was widely adopted in the software community and beyond. The Software CMM had five maturity levels that prescribed process improvement priorities for software organizations. Maturity Level 4 in the Software CMM focused on applying quantitative techniques, particularly statistical techniques, for controlling the software process. These concepts carried forward into the CMM Integration (CMMI) models that succeeded the Software CMM. The maturity concepts were extended to the capability levels for individual processes in ISO/IEC 15504, which were also adopted by CMMI.

In statistical process control, this means eliminating special (or assignable) causes of variation and addressing common causes of variation. Because the software process is not a repetitive manufacturing or service process, the application of statistical process control, specifically control charts, has been challenged by many in the software community. This tutorial discusses the issues in applying statistical thinking to the software process, prerequisites for applying statistical control, and the specific techniques that should be considered.

Quantitative management implies an understanding of variation, so that data can be properly understood when doing measurement-driven decision making. Decision making based on data
implies that the data is collected and analyzed frequently enough that “real-time control” of the process is possible and that the comparability of the process and product context for the data is understood. An understanding of variation is grounded in the principles of statistical thinking, and the appropriate use of statistical tools is an integral part of quantitative management. A common error in “quantitative management” is analyzing data without doing the statistical thinking that supports real insight. For software projects, real-time process control means that the data being used for control is collected, analyzed, and decisions made on the spot. Measurement and analysis on a monthly or longer basis is inadequate for real-time control, but many organizations fail to understand the impact on efficiency of immediate, informed reaction. Those who do no understand the comparability of their data may aggregate dissimilar data together. Statistical control depends on a profound knowledge of both process and product. Failing to set the product context via product lines, systematic reuse, domain-specific architectures, and similar approaches is a mistake that results in an unacceptable amount of variation in the data. It is important to know when statistical control is feasible and when approaches such as risk management are more appropriate; not every process should be statistically controlled. Examples from real-world software projects illustrate the challenges in stabilizing the software process and applying different statistical techniques.

**Mature Software Project Management**

This seminar addresses the challenges, strategies, and tools for managing software projects. The seminar is structured according to the knowledge areas of the Project Management Body of Knowledge. Customer relationship management, decision making, earned value, critical path, critical chain, and agile methods will be discussed.

Effective project management is a prerequisite for meeting commitments, yet all too many software projects fail to meet customer expectations for budget, schedule, functionality, and quality. Many customers now require their suppliers to demonstrate their commitment to process improvement and quality management by using, or being certified against, various models and standards. Project management is a fundamental requirement to achieve Maturity Level 2 against the Capability Maturity Integration (CMMI) for Development or to be certified against ISO 9001.

Effective project management requires more than implementing a set of basic functions, however. It implies selecting qualified people who can work together effectively, structuring decision making processes both internally and externally, managing customer expectations, monitoring progress, managing risks, and taking corrective action as appropriate. None of these are easy, although there are a number of tools that can help.

This seminar is therefore a broad survey of a variant of project management styles and techniques, as used in diverse environments. The perspective of the discussion is that of “mature” processes, i.e., processes that are well-defined, managed, measured, controlled, and effective.
The eSourcing Capability Model (eSCM)

Organizations are increasingly delegating their information technology (IT) intensive business activities to external service providers, taking advantage of the rapid evolution of the global telecommunications infrastructure. The business processes being outsourced range from routine and non-critical tasks, which are resource intensive and operational, to strategic processes that directly impact revenues. Managing and meeting client expectations is a major challenge in IT-enabled outsourcing services and examples of failure abound. The eSourcing Capability Model (eSCM) contains a set of 93 best practices that address the entire outsourcing (or insourcing) process. The eSCM is structured along three dimensions: phases, organizational elements, and capability levels. The phases dimension addresses pre-contract and post-contract activities as well as contract execution and overall practices. Organization elements include organizational management, people, business operations, technology, and knowledge management. The eSCM has five capability levels that address performing to meet client requirements, controlling through measurement, enhancing through innovation, and sustained excellence. The eSCM aids IT-enabled outsourcing service providers in forming, managing and improving outsourcing relationships. This tutorial provides an overview of outsourcing issues and the eSCM model.