

DISS 725 – System Development: Research Paper 4  
Software Process Assessment and Improvement Models

by

Ronald G. Wolak  
wolakron@nova.edu

A paper submitted in fulfillment of the requirements  
for DISS 725 Spring 2001 – System Development: Research Paper 4

Graduate School of Computer and Information Sciences  
Nova Southeastern University

July 2001

An Abstract of a Paper Submitted to Nova Southeastern University in Fulfillment of the Requirements for DISS 725 Spring 2001 – System Development: Research Paper 4

DISS 725 – System Development: Research Paper 4  
Software Process Assessment and Improvement Models

by  
Ronald G. Wolak

July 2001

The paper that follows was submitted to satisfy the requirements of DISS 725 Spring 2001 – System Development: Research Paper 4. Software process assessment and improvement is recognized as an important part of the software development life cycle. Several contemporary models have been developed to assist organizations evaluate and improve their software development processes and capabilities. In the following pages, the paper began with an introduction to software process assessment and improvement. In this discussion, several process models were described: BOOTSTRAP, Capability Maturity Model (CMM), ISO 9001, Personal Software Process (PSP), SPICE, Team Software Process (TSP), and Trillium. This was followed by a look into the economics of software process improvements along with techniques to increase the success rate of assessment and improvement projects. The paper concluded with a summary and recommendations.

## Chapter 1 Introduction

Software development is a challenging undertaking that is often critical to the safety of humans and the welfare of businesses (Zahran, 1998). Problems caused by low quality, unreliable software applications include widespread inconvenience and loss of life. For example, the June 1996 explosion of a European Space Agency rocket carrying a number of satellites was attributed to software failure. Improved software quality is critical to ensure reliable products and services and to increase customer confidence. The following introductory sections describe the problem to be investigated and the goal to be achieved. In addition, the introduction provides an analysis of the relevance of the research and discusses the paper's five-chapter format.

### **Problem Statement and Goal**

The U.S. Government Accounting Office (GAO) reported that poor quality software systems are responsible for millions of dollars in cost overruns, schedule delays, and multi-billion dollar systems with substandard performance (Zahran, 1998). In response to this problem, the software industry has embraced software process assessment and improvement. Large sums are invested by software developers to improve application quality and reliability through the use of assessment and improvement models.

The goal of this paper is to provide an overview of contemporary software process assessment and improvement models, to discuss the economics of these models, and to investigate techniques to increase the success rates of assessment and improvement projects.

### **Relevance**

This research paper is relevant to the topic of software process assessment and improvement. The paper begins with an introduction to software process assessment and improvement. In this discussion, several contemporary models are explored: BOOTSTRAP, Capability Maturity Model (CMM), ISO 9001, Personal Software Process (PSP), SPICE, Team Software Process (TSP), and Trillium. This is followed by a look into the economics of software process improvement along with techniques to increase the success rate of assessment and improvement projects.

### **Format**

This research paper is a descriptive study formatted in five chapters. The first chapter covers the paper's problem statement and goal, relevance, and format. This is followed in the second chapter by a review of the literature relevant to the problem. In the third chapter, the research methods and online tools and resources employed during the completion of the paper are described. The fourth chapter presents the results of the research and provides an analysis of the economics and benefits of software process

assessment and improvement projects. The fifth chapter concludes the paper with a summary and recommendations.

### **Summary**

Software process assessment and improvement is an integral part of the software development life cycle. The improved software quality that results from these efforts increases product reliability and enhances customer satisfaction (Grady, 1997). In the following pages, this paper provides a review of literature relevant to software process assessment and improvement, a description of research methods employed, results of the research, recommendations, and an overall summary.

## Chapter 2 Review of Literature

The literature review that follows is organized by subject heading. Those subjects include the major software assessment and improvement models in use today. Included are BOOTSTRAP, Capability Maturity Model (CMM), ISO 9001, Personal Software Process (PSP), SPICE, Team Software Process (TSP), and Trillium.

### **BOOTSTRAP**

BOOTSTRAP is a European method for software process assessment and improvement that was developed to speed up the application of software engineering technology in the European software industry (Zahran, 1998). The BOOTSTRAP methodology is based on the CMM (discussed in the next section). However, it has been extended and adapted to include ISO 9000 guidelines and the European Space Agency software engineering standard (ESA-PSS-05).

Unlike the CMM, BOOTSTRAP does not assume strict adherence to a distinct key practice model and allows the use of alternative approaches (Zahran, 1998). This has been a key factor in its success. In addition, BOOTSTRAP has proven suitable for use by all kinds and sizes of software development organizations. The main features of BOOTSTRAP are:

- Questionnaires for both site and project evaluation
- Uniform procedure and mandatory assessor qualification/training
- Constructive instead of a normative approach
- Open questions
- Immediate feedback and action planning

In a related article, Stienen (1999) described the main characteristics of the BOOTSTRAP method. These included the reference framework, the assessment procedure, the structure of the questionnaires, and the rating and scoring mechanisms employed. The BOOTSTRAP method adopted a process model which addresses processes and practices for both the software producing unit and the project. Process areas were divided into organization, methodology, and technology.

### **CMM**

The Capability Maturity Model (CMM) plays an important role in the software improvement efforts (SPI) of organizations worldwide (Zahran, 1998). The process was developed by the Software Engineering Institute at Carnegie Mellon University in 1986. Its goal is to improve, over time, the application of an organization's software technologies. The model provides a guide for organizations to select software process improvement strategies by facilitating the determination of current capabilities and the identification of critical issues.

The CMM process is made up of five well-defined levels of sequential development: initial, repeatable, defined, managed, and optimizing (Freedman, 2000). These maturity levels provide a progressive scale for measuring the maturity of an organization and its ability to use software technologies. Organizations that depend on

formal rules, instead of individual performers, to manage software projects are considered to be more “mature.”

A recent journal article reported the results of several years of empirical study at organizations using the CMM (Herbsleb, Zubrow, Goldenson, Hayes, & Paulk, 1997). The studies consistently showed significant organizational performance improvements that were directly associated with process maturity. The data did not indicate any differences in the success in using the CMM for organizations of assorted sizes and types. However, there were hints that small companies found pieces of the CMM irrelevant and hard to apply.

A related article by Brodman and Johnson (1997) discussed a modified version of the CMM that was more suitable for small organizations and small projects. Problems typically reported with the CCM when used by these organizations were:

- Documentation overload
- Unrelated management structure
- Inapplicable scope of reviews
- High resource requirements
- High training costs
- Lack of need guidance
- Unrelated practices

Another article published by the SEI gave a comprehensive description of the CMM for software (Paulk, Weber, & Chrissis, 1999). The article stressed the need for a process maturity framework to prioritize improvement actions and to describe the five maturity levels. The IDEAL approach to software process improvement was also described. This approach consisted of five phases: initiating, diagnosing, establishing, acting, and leveraging. Finally, future directions for the CMM were discussed. Key aspects of these future directions include active voice, supplier management, risk management, product line engineering, and split quantitative process management.

## **ISO 9001**

ISO 9001 is an international standard for quality assurance in design, development, production, installation, and service (Weissfelner, 1999). It is broken down into twenty elements. ISO 9001-3 relates to the development, supply, and maintenance of software. Almost 90 percent of the companies that completed ISO 9001 implementation reported improved internal documentation as one of the most important benefits of registration. Other benefits included higher product quality, greater internal quality awareness, and increased competitive advantage.

ISO 9001 is similar to the CMM in the following areas: emphasis on process, documented processes, practiced processes, address the “what” and not the “how” (Zahran, 1998). Differences between the two approaches occur in the areas of focus, dimensions, assessment and certification, coverage, supplier’s role, and level of detail. For example, an ISO 9001-compliant software organization would not necessarily satisfy the requirements for a level 2 in the CMM. However, it would satisfy most of the level 2 and some of the level 3 goals.

A related article reported the status of implementing an ISO 9001 compliant quality system in a small software organization (Demirors, Demirors, Dikenelli, & Keskin, 1998). Among the challenges encountered during the installation were a lack of

guidance, action knowledge, maturity, and quality personnel. In response, ISO procedures were adapted in the following ways: role combination, shorter development cycles, enhanced early communication, simplified procedures, and minimized paperwork.

### **Personal Software Process (PSP)**

The Personal Software Process is a process-based method developed by the SEI for software engineers to use to apply process definition and measurement to their personal tasks (Humphrey & Over, 1997). Most important, the PSP shows developers how to manage product quality, meet commitments, and justify their plans with data. In addition, the PSP follows the concepts of the CMM. The key message of the PSP is that developers should use process management concepts to identify the methods most effective for them. A typical PSP course uses ten software development exercises, a structured sequence of defined processes, and five data analysis exercises to demonstrate the process.

An article by Silberberg (1998) reported on the successful application of the PSP to Ada software development. Examples of improvement made by the process included improved size and time estimating accuracy, reduced project time in the compile and test phases, and improved defect removal yield. The paper also provided a brief introduction to the PSP along with a comparison of the PSP with the CMM.

### **SPICE**

ISO/IEC 15504 was the result of the SPICE (Software Process Improvement and Capability dEtermination) project (Zahran, 1998). It provides a reference model for focused self assessments and includes a capability scale that is simple to understand.

SPICE defines a two-dimensional model used in a process assessment to describe processes and process capability (Drouin, 1999). The first dimension details the processes an organization should use to supply, develop, operate, evolve, and support software. The second dimension is made up of nine generic attributes used to characterize the capability of a process. These attributes are grouped into six capability levels (0-5).

### **Team Software Process (TSP)**

TSP is a defined method for a group of software developers to create quality software in an efficient manner (Hilburn, 2000). It provides process scripts, guidelines, tools, and techniques for a team to develop software applications. The process is based on an incremental model that divides effort into “development cycles.” Each cycle involves producing software that satisfies a subset of the total software requirements.

In another article, Hilburn and Towhidnejad (2000) discussed how TSP was used during a team software project in a junior level university course. The process provided students with clear, precise guidance and support, good data collection and analysis techniques, and an environment for building a successful software development team. The report concluded that the TSP was an excellent mechanism to emphasize software quality.

### **Trillium**

The Trillium model was initially designed for use with embedded software systems (e.g. telecommunications) and is based on the CMM (Coallier, Mayrand, & Lague, 1999). Its architecture differs from the CMM in the following ways:

- Architecture is based on roadmaps instead of key process areas
- A product rather than a software perspective
- Wider coverage of capability impacting issues
- Customer focus and a telecommunications orientation

Trillium is comprised of five levels (1-5). These are unstructured, repeatable and project oriented, defined and process oriented, managed and integrated, and fully integrated (Zahran, 1998). Trillium can be used in a number of ways. For example it can be used to benchmark an organization's product development process against industry best practices or to self-assess and identify opportunities for improvement. In addition, it is useful in pre-contractual negotiations to select a supplier.

### **Summary**

The literature review presented above was organized by subject heading. The subjects included the major software assessment and improvement models in use today: BOOTSTRAP, Capability Maturity Model (CMM), ISO 9001, Personal Software Process (PSP), SPICE, Team Software Process (TSP), and Trillium.

## Chapter 3 Methodology

### **Research Type**

This paper was a research-based descriptive study. The key outcome of the investigation was the exploration contemporary software process assessment and improvement models along with a look into the economics and success rates of assessment and improvement projects.

### **Research Methods Employed**

The primary research method employed throughout the course of writing this paper was browser-based Internet searches. The literature reviewed included textbooks, journal articles, and magazine articles referenced by a select set of online resources. Relevant texts were located, ordered, and delivered using the Fatbrain.com Internet site. The full text articles from journals and magazines were located and subsequently downloaded.

### **Online Tools and Resources**

A variety of online resources were used to locate and download literature relevant to the goal of the paper. These resources included ACM Search ([www.acm.org/dl/search.html](http://www.acm.org/dl/search.html)), IEEE Digital Library (<http://computer.org/search.htm>), and ProQuest Direct (<http://proquest.umi.com/>). Perhaps the most powerful search tools to be employed were the intelligent search agents Copernic 2001 and LexiBot.

Copernic 2001 is a well-documented freeware search agent (Copernic, 2001). It uses predefined channel sets, which allows researchers to target inquiries to all major Web search engines and also search for relevant text in newsgroups. Copernic conducts fast, multithreaded, full Boolean searches with progress displays and customizable search depth. Once results are compiled, Copernic displays returns (including name, location, and introductory text) in a right-click-enhanced list box sorted by relevance.

Another search technology utilized to gather literature was LexiBot from BrightPlanet (LexiBot, 2001). The LexiBot desktop search client acts as a universal translator for all dialects of search engines and searchable databases. LexiBot is able to search 150 services at one time using a standard query format. In addition, LexiBot's search technology is capable of identifying, retrieving, qualifying, and organizing "deep" and "surface" content from the Internet.

### **Summary**

In summary, this paper was a research-based descriptive study. Browser-based Internet searches were the primary research method employed. These searches queried databases that included the ACM, IEEE, and ProQuest Direct. Specialized client-based search technologies (i.e. Copernic 2001 and LexiBot) also aided in locating relevant literature.

## Chapter 4 Results

Competition among software developers continues to intensify as customers demand shorter cycle times, added functionality, and improved quality (Harter, Krishnan, & Slaughter, 1998). As they strive to deliver improved software at an ever increasing rate, software companies are often reluctant to sacrifice quality and incur increased development costs in order to decrease the time to market. Many are investing in software process assessment and improvement projects as one way to reduce costs, shorten cycle time, and increase quality. The following sections begin with a discussion of the economics (i.e. costs) associated with software assessment and improvement efforts. This is followed by an investigation into the benefits of these programs along with a discussion of techniques to effectively conduct software process assessment and improvement programs.

### **Economics**

The cost per capita of major software process improvements can exceed \$25,000 and the implementation time last as long as five years in large software development organizations (Jones, 1999). These per employee costs include training, consulting fees, software licenses, capital equipment, and office improvements. A recent study found that the “smallest” amount that can be spent and still achieve tangible benefits is around \$800. These are incurred through the use of methodological improvements that do not require capital investments: Joint Application Design (JAD), usage of function point metrics, and the usage of inspections. However, improvements in schedule, cost, and quality do not exceed 10 percent for this low an investment.

At the other extreme, several companies and government agencies have reported spending in excess of \$10,000 per employee with little or no benefit (Jones, 1999). Studies have also shown that smaller software companies are able to implement software process improvements methodologies at a lower cost and more rapidly than large corporations and government agencies.

Finally, the return on investment (ROI) for software process improvements typically ranges from \$3 to \$30 for every dollar invested (Jones, 1999). Maximum return occurs in the final stages of improvement. This is when software reusability programs take full effect.

### **Benefits**

The benefits of software process assessment and improvement include lowered costs, increased quality, and reduced time to market (Krasner, 1999). For example, the SEI reported the following:

- Annual productivity gain of 37 percent
- Annual reduction in time to market of 19 percent
- Annual reduction in post-delivery defects of 45 percent
- Average ROI for SPI of 5.7:1

Often the biggest payoffs of software process assessment and improvement projects are in human terms and not in dollars (Krasner, 1999). These include factors

such as pride in work, increased job satisfaction, and improved ability to attract and retain software experts.

### **Improvement Techniques**

Implementing an SPI program is not as easy as teaching a new design method or purchasing a new software system (Sakamoto, Nakakoji, Takagi, & Niihara, 1998). Software organizations must experience a “paradigm shift” to be successful. This includes effectively dealing with an organization’s resistance to change. One effective way of dealing with this resistance is to convince stakeholders with concrete and tangible examples of both problems and improvements.

In addition, software development efforts succeed because of carefully selected processes (McGuire & Randall, 1998). The successful implementation of SPI programs requires focusing on many people, organizational, process, quality, and methodological issues. Examples include (Wieggers, 1999):

- Consistent management leadership with a focus on quality
- Time to identify and implement improved processes
- Realization that SPI is based on common sense and commitment

Another technique to increase the chances of success is to watch for and eliminate the following factors that undermine SPI deployment (Wieggers, 1999):

- Lack of management commitment
- Unrealistic management expectations
- Time-stingy project leaders
- Stalling on action plan implementation
- Achieving a CMM level becomes the primary goal
- Inadequate training
- Expecting defined procedures to allow staff interchangeability
- Ineffective process assessments

### **Summary**

The results chapter presented above began with a discussion of the economics associated with software assessment and improvement efforts. This was followed by an investigation into the benefits of these programs along with a discussion of techniques to successfully conduct software process assessment and improvement programs.

## Chapter 5 Conclusion

Software process assessment and improvement initiatives by the software industry are in response to low quality, high cost software applications. Developers have committed large sums to these programs. However, SPI programs must be embraced at all levels of an organization in order to be successful. Business managers should understand their business benefits. Project managers must welcome the increased visibility into the software process that they provide, and end users should be interested because of the higher probability of meeting cost, quality, and timing goals provided.

### **Summary**

The research paper presented above was a descriptive study formatted in five chapters. The first chapter covered the paper's problem statement and goal, relevance, and format. This was followed in the second chapter by a review of the literature relevant to the problem. In the third chapter, the research methods and online tools and resources employed during the completion of the paper were described. The fourth chapter presented the results of the research and provided an analysis of the economics and benefits of software process assessment and improvement projects. Finally, the last chapter concluded the paper with a recommendations and summary.

## References

- Brodman, J., & Johnson, D. (1997). *A software process improvement approach for small organizations and small projects*. Paper presented at the International Conference on Software Engineering, Boston, MA.
- Coallier, F., Mayrand, J., & Lague, B. (1999). Risk Management in Software Product Procurement. In K. Emam & N. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement* (pp. 23-44). Washington, DC: IEEE Computer Society Press.
- Copernic (2001). Copernic 2001. Retrieved June 6, 2001, from the World Wide Web: <http://www.copernic.com>.
- Demirors, E., Demirors, O., Dikenelli, O., & Keskin, B. (1998). *Process improvement towards ISO 9001 certification in a small software organization*. Paper presented at the International Conference on Software Engineering, Dunedin, New Zealand.
- Drouin, J. (1999). The SPICE Project. In K. Emam & N. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement* (pp. 45-55). Washington, DC: IEEE Computer Society Press.
- Freedman, A. (2000). *Computer Desktop Encyclopedia* (Vol. 13.4). Point Pleasant, PA: The Computer Language Company.
- Grady, R. (1997). *Successful Software Process Improvement*. New York: Prentice Hall.
- Harter, D., Krishnan, M., & Slaughter, S. (1998). *The life cycle effects of software process improvement*. Paper presented at the International Conference on Information Systems.
- Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997). Software quality and the capability maturity model. *Communications of the ACM*, 40(6), 30-40.
- Hilburn, T. (2000). *Teams need a process!* Paper presented at the Annual Joint Conference Integrating Technology into Computer Science Education, Helsinki, Finland.
- Hilburn, T., & Townhidnejad, M. (2000). Software quality: A curriculum postscript? *ACM SIGCSE Bulletin*, 167-171.
- Humphrey, W., & Over, J. (1997). *The Personal Software Process (PSP)*. Paper presented at the International Conference on Software Engineering, Boston, MA.

- Jones, C. (1999). The Economics of Software Process Improvements. In K. Emam & N. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement* (pp. 133-149). Washington, DC: IEEE Computer Society Press.
- Krasner, H. (1999). The Payoff for Software Process Improvement: What it is and How to Get it. In K. Emam & N. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement* (pp. 151-176). Washington, DC: IEEE Computer Society Press.
- LexiBot (2001). Our Technology - Results: The LexiBot Expression. *BrightPlanet*. Retrieved June 6, 2001, from the World Wide Web: <http://www.brightplanet.com/technology/results2.asp>.
- McGuire, E., & Randall, K. (1998). *Process improvement competencies for IS professionals*. Paper presented at the Conference on Computer Personnel Research, Boston, MA.
- Paulk, M., Weber, C., & Chrissis, M. (1999). The Capability Maturity Model for Software. In K. Emam & N. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement* (pp. 3-22). Washington, DC: IEEE Computer Society Press.
- Sakamoto, K., Nakakoji, K., Takagi, Y., & Niihara, N. (1998). *Toward computational support for software process improvement activities*. Paper presented at the International Conference on Software Engineering.
- Silberberg, D. (1998). *Applying the personal software process (PSP) with Ada*. Paper presented at the Annual International Conference on Ada.
- Stienen, H. (1999). Software Process Assessment and Improvement: Five Years of Experiences with BOOTSTRAP. In K. Emam & N. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement* (pp. 57-75). Washington, DC: IEEE Computer Society Press.
- Weissfelner, S. (1999). ISO 9001 for Software Organizations. In K. Emam & N. Madhavji (Eds.), *Elements of Software Process Assessment and Improvement* (pp. 77-100). Washington, DC: IEEE Computer Society Press.
- Wieggers, K. (1999). Software process improvement in Web time. *IEEE Software*, 16(4), 78-76.
- Zahran. (1998). *Software Process Improvement: Practical Guidelines for Business Success*. Reading, MA: Addison-Wesley.