2003

SPSM and SPC for software process and project management

Omar F. El-Gayar
Dakota State University

SPSM and SPC for Software Process and Project Management.

Conference Paper · January 2003

Source: DBLP

CITATIONS
0

READS
126

1 author:

Omar F. El-Gayar
Dakota State University
156 PUBLICATIONS **1,505** CITATIONS

SEE PROFILE

SPSM AND SPC FOR SOFTWARE PROCESS AND PROJECT MANAGEMENT

Omar F. El-Gayar[1]
Dakota State University
omar.el-gayar@dsu.edu

Abstract

Software process simulation modeling (SPSM) provides a platform for process and project management. On the other hand, statistical process control (SPC) provides the means to assess a process for stability and capability through visualization.

This paper provides a framework for integrating SPSM and SPC. The framework realizes their synergy by leveraging the investment in supporting software measurement processes, ensuring the stability of the underlying development process, and enhancing simulation output using visualization techniques made possible through SPC. The paper illustrates the framework by integrating SPC to an existing SPSM. The integrated model is then used to evaluate the underlying process with respect to capability.

Keywords: Software project management, software process management, software process simulation modeling, statistical process control, visualization

Introduction

Software process and project management are tightly coupled. A project may fail due to poor project management despite adequate technical competence and the execution a quality software development process. Similarly, a project may fail due to deficient development process despite competent project management. Moreover, the success of both process and project management relies heavily on the successful development and implementation of effective measurement processes and supporting quantitative techniques such as simulation. Finally, process and project management may share the same people. Process management personnel may have project management responsibilities, and project managers may assume process management responsibility.

This paper focuses on two techniques of particular importance to software process and project management, namely, software process simulation modeling (SPSM) and statistical process control (SPC). While initial effort to model software processes relied on informal description of software processes, it was not until the late 1980's when focus turned to more formal description resulting in a number of modeling paradigms and numerous modeling techniques. Accordingly, software process modeling emerged as a technique for defining and analyzing significant aspects of software processes. The aim ranged from facilitating human understanding, process management, process improvement, process guidance, to project management (Krasner et al. 1992). Of these techniques, SPSM provides a 'realistic' platform for understanding, managing, and improving software processes. From a project management perspective, SPSM is used for project planning, in particular, cost and schedule estimation, and project quality, risk, and human resource management. Simulation provides the means to estimate the effect of changes and study the effect of various 'what if' scenarios. However, the process quality dimensions of the software process are often overlooked in such models.

[1]This material is based upon work supported by the National Science Foundation/EPSCoR Grant #EPS-0091948 and by the State of South Dakota. The author also thanks Peter Lakey of Cognitive Systems for graciously providing the model used in this paper. The usual disclaimer applies.

On the other hand, SPC is a process monitoring technique that has been widely used in manufacturing industries and, since the mid-eighties; several attempts have been made to apply SPC to software processes (Cho 1987; Lanphar 1990; Putnam 1991; Card 1994; Card 1999). Since then, a number of applications are reported (Weller 2000; Florac et al. 2000; Lewis 1999). The importance of SPC to software project management cannot be undermined. Specifically, project management processes are responsible for seeing that a software product is developed according to schedule, and budget, and that the resulting software product meets functional and quality requirements. However, without the underlying process management activities, project managers assume a significant risk in setting and meeting project objectives.

Inherent in existing applications of SPC to the software process is the view of software as a manufactured product. Accordingly, techniques of SPC can be applied to software processes just as they have been applied to manufacturing processes (Florac et al. 1997). Nevertheless, as noted by Lantzy (1992), there are fundamental differences between software processes and manufacturing processes. Software development processes are very different from manufacturing processes in terms of many key process dimensions such as product volume, level of automation, standardization of processes, among others. Most notably, is that in software processes, the transformation of user requirement into software are dominated by cognitive activities, this contrasts with manufacturing activities where cognition is minimized. Moreover, in software processes, the inputs and outputs for a software process are different for each instance of the process (Lantzy 1992). In other words, each software product generated by a particular process is different (as determined by user requirements). The interaction between cognitive dominated activities and different inputs and outputs create challenges to the application of SPC for software. In particular,

1. The data regarding input and outputs experience great variability, which make it particularly difficult to develop meaningful control limits.
2. It is difficult to assign causes of variability in the data.
3. It is difficult to visualize the performance of the process using the traditional control limits as they typically contain too much variability to provide any useful guidance in managing the process.

These weaknesses call into question the applicability and most important the usefulness of SPC to the software development domain because these weaknesses are common to many software development projects.

Contribution of the Paper

This paper targets software project managers, particularly those with process management responsibilities. In this paper, the author proposes a framework for integrating SPSM with SPC for directing software process improvement effort in a software project management context. In essence, simulation provides a 'controlled' environment (thereby mitigating the aforementioned shortfalls of SPC) for evaluating a process for stability and capability, while SPC allows for assessing the process quality dimension through visualization. Specifically:

- Several of the activities involved in SPC, such as identifying objectives and measures, and collecting data are also required for SPSM. The proposed framework makes better utilization of such activities by supporting both SPSM and SPC.
- In developing SPSM, a step that is often overlooked is to assess the underlying process for stability. This is important for accurately capturing the stochastic properties of model input as well as validating and verifying the model. The integrated model utilizes SPC to assess process stability prior to model development.
- Analysis of simulation output is greatly enhanced by the different visualization techniques made possible through SPC.

Outline of the Paper

The paper is organized as follows; the next section defines software process modeling and its objectives as well as a brief review of software modeling paradigms and techniques. Next, is a brief description of statistical process control and its application to software processes, followed by a presentation of the framework for integrating SPSM and SPC. The case study illustrates the applicability of the proposed framework. The final section concludes the paper.

Software Process Simulation Modeling

A software process is comprised of a set of policies, organizational structure, technologies, procedures, and artifacts needed to conceive, develop, deploy, and maintain a software product (Fuggetta 2000). As such, software process modeling seeks to define

and analyze significant aspects of these processes (Fuggetta 2000). While there is not a wide-ranged consensus on the essential constructs of a process model, most frequently mentioned constructs include agents, roles and artifacts (Curtis et al. 1992) as well as activities and tools (Fuggetta 2000) as shown in Table 1.

There are a number of objectives driving software process modeling, most notably (Curtis et al. 1992; Kellner et al. 1999):

- Facilitating human understanding and communication
- Supporting process improvement
- Supporting process management
- Automating process guidance and execution
- Supporting project management
- Facilitating training and learning

To that effect, and since the late 1980's, a number of process modeling paradigms and numerous techniques evolved for modeling software processes. Examples of which include programming language, Petri-net, object-oriented, and quantitative, e.g., systems dynamics and discrete event simulation

System Dynamics

In project management, Abdel-Hamid and Madnick (1991) pioneered the application of system dynamics to software process modeling in general, and to software project management in particular. They present a generic model of software development and use the model to demonstrate various phenomena encountered in software projects, e.g., the 90% syndrome and Brooks' Law. The model also illustrates the potential for cost and schedule estimation under different management scenarios as well as the economics of quality assurance.

Table 1. Software Process Modeling Constructs

Modeling Construct	Description
Agents	An actor who performs an activity (process element)
Roles	Set of responsibilities assigned to an agent
Artifacts	A product created or maintained by the activities
Activities	Steps that need to achieve process objectives
Tools	Are to be utilizes by the process

The use of system dynamics in software project management has also been applied to quality, risk and human resource management. Madachy (1996) used system dynamics to evaluate the impact of performing formal inspections on project cost, schedule and quality. The model captured interrelated flows of tasks, errors inspection activities and personnel through the development process and was calibrated to industrial data. Rus and Collofello (1998) also used system dynamics to quality planning by evaluating various software reliability engineering strategies. In contrast to static reliability models, the model presented can be used to track the quality and reliability of the software throughout the development process by tracking project metrics and adjusts the model accordingly. Another example for the quality planning is a model developed by Tvedt and Collofello (1995) for an incremental software development process. The objective of the model is to evaluate the impact of various process improvement initiatives on development cycle time.

With respect to human resource project management, Collofello et al. (1998) uses a system dynamics model to assess the effect of managerial staffing decision on project's budget, schedule and quality. Particular attention is paid to evaluating process to integrate the effects of staff attrition. Abdel-Hamid (1989) also used system dynamics to answer "what-if" questions regarding various staffing practices with particular emphasis on the interchangeability of persons and months in a software project.

On project risk management, SPSM is particularly valuable for quantitative risk analysis particularly in evaluating the impact of various management policies on project cost, schedule and quality.

Discrete Event Simulation Models

In contrast to system dynamics models, discrete event simulation models (DESM) refers to models which evolve in time in response to the occurrence of discrete events. Such models allows for modeling systems that cannot be easily described by ordinary differential equations (i.e., system dynamics). Examples of such systems can be found in manufacturing, computer communication networks, and traffic control (Ho and Cao 1991)

In SPSM, DESM allows for capturing a software development process as a sequence of discrete activities where items (e.g., code modules) move from one activity to the next (e.g., from coding to testing). Items have attributes such as size, and quality, and activities change the attributes of items as these items are processes by the activities. Moreover, changes to the values for attributes may be deterministic or stochastic thereby capturing the uncertainty inherent in some of the attributes.

In software project management, Lakey (2000) of Cognitive Concepts developed a DESM to be used as a project management tool for estimating schedule, effort and quality during the detailed design for a computer software configuration item (CSCI). The underlying development process is an iterative and incremental process in which the CSCI is developed incrementally over a number of iterations. Each iteration includes a development phase, a review phase, and a rework phase. For each of the phases, the model tracks the effort involved as well as a number of other metrics such as the number of defects generated, found, and escaped during the process.

An important feature of this model is the incorporation of system dynamics modeling concept of feed back loops. Specifically, when evaluating these metrics the model takes into account a variety of process and product factors. Examples of product factors include the size quality and complexity of the input artifacts, while examples of process factors include schedule pressure, communication overhead and tools support. For more information about this model, please refer to (Lakey 2000).

Statistical Process Control

The increasing demand for software in terms of quantity and quality, places tremendous pressure on the software engineering community to adopt techniques for continuous improvement of their software development processes (Lantzy 1992). However, process improvement demands measurement of product and process attributes. In this vein, and over the past decade, concepts and methods associated with process improvement have gained wide acceptance in the software community (Florac et al. 1997). Of particular importance is the application of SPC to the software process. The strengths of SPC are that they provide easy to use graphical tools that enable the manager to visualize the performance of the manufacturing process (system) along multiple dimensions of performance. Moreover, the Quantitative Process Management Key Process Area of the Capability Maturity Model requires that projects manage their processes quantitatively and recommend that SPC be applied at this level.

Underlying SPC is the notion that almost all the characteristics of processes and products display variation when measured over time. This variation has two sources (Florac et al. 1997):

Total variation = common cause variation
+ assignable cause variation

where common cause variation: variation that stems from natural phenomena that are inherent to the process. The results of common variation are random and are within predictable bounds.

assignable cause variation: variation that have assignable causes that could have been prevented. Examples of assignable causes include changes to work environments, hardware failures, programmers' burnout, inadequate training, and so forth.

Process stability is at the core of process management. It refers to a predictable 'in control' process where the causes for all variations are attributed to common (natural) causes. On the other hand, process capability refers to the capability of a process to satisfy the requirements of the customer "voice of the customer"

Control charts are at the center of statistical process control. Control charts allow a software manager to visually assess a software process for stability and capability. A typical control chart plots the measured averages of a quality attribute from the process over time. A center line on the chart represents the process average of the quality attribute that correspond to the stable state. An upper

control (UCL) and a lower control (LCL) limits on the quality attribute are also plotted (Figure 1). The quality attribute must fall within this range if the process is to be in control. In depth discussion of various control charts can be found in (Montgomery 1997; Quesenberry 1997).

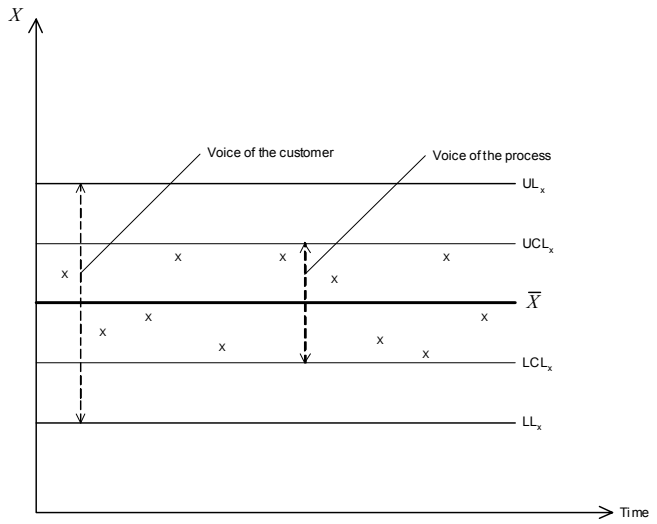

Figure 1a. X Control Chart for a Stable and Capable Process

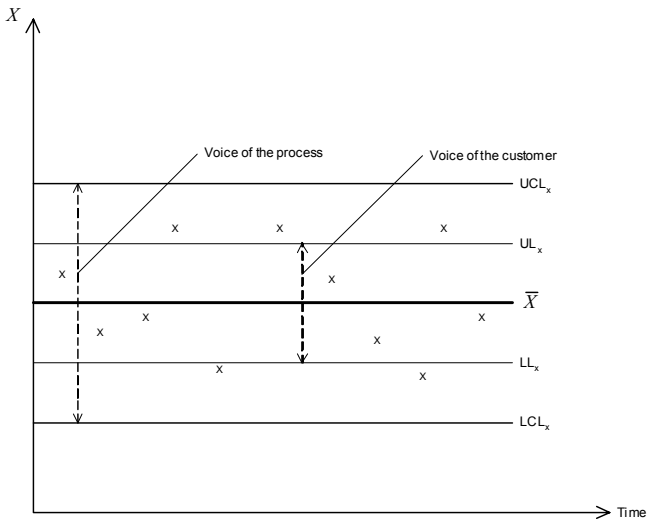

Figure 1b. X Control Chart for a Stable but Not Capable Process

For a software process, software entities are one of a kind, and as such we employed the control chart methods for individual measurements, the X and MR charts to gain an understanding of the process status and variability. The centerline and control limits for the X and MR charts are determined as follows:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$MR_i = \left| x_i - x_{i-1} \right|, i = 1,2,...., n-1$$

$$\overline{MR} \frac{\sum_{i=1}^{n-1} MR_i}{n-1}$$

$$UCL_X = \overline{x} + 3 \cdot \frac{\overline{MR}}{d_2}$$

$$LCL_X = \overline{x} - 3 \cdot \frac{\overline{MR}}{d_2}$$

$$UCL_{MR} = \overline{MR} \cdot D_4$$

$$LCL_{MR} = \overline{MR} \cdot D_3$$

where x_i : Denotes measurement *i* for some process attribute for *n* measurements.

$\bar{x}$: Center line for the X chart

MR_i : Moving range for two successive observations

$\overline{MR}$: Center line for the MR chart

UCL_X, LCL_X : Upper and lower control limits for the X chart, respectively.

UCL_{MR}, LCL_{MR} : Upper and lower control limits for the MR chart, respectively.

d_2, D_3, D_4 : 1.128, 0, and 3.27, respectively.

The Framework

The proposed framework integrates SPSM and SPC. The underlying premise is that these two approaches complements each other and that through integration, their synergy is realized as noted at the beginning of the paper.

As shown in Figure 2, the first step is it to establish goals and objectives. As indicated by (Florac et al. 1997), process measures are driven by business goals such as improving product quality, reducing time to market, and reducing cost. Moreover, business goals also determine the focus of the model. In particular, by identifying business goals one can answer questions such as why is one simulating, what is the scope of the model, what questions is one trying to answer, and so forth (Nordgren 1995).

Next, is the need to conduct a review of the software process currently in place. The review should include surveying existing documentation as well as interviewing software developers to ensure that the implemented process is consistent with the supporting process documentation. The purpose is to get a thorough understanding of the software process, thereby facilitating the development of a mental model of the process.

A mental model represents the picture one has in mind for the process under consideration. Specifically, it represents the tasks, resources, and products constituting the software process. As indicated by (Florac et al. 1997), forming and documenting a mental model is one technique for identifying key process issues which is a precursor to selecting and defining measures. Moreover, the flowcharting of the mental model is the first cut at the simulation model.

Using the mental model documented in the previous step, the next step is to select and identify process measures in accordance with the objectives and goals. Florac et al. (1997) provides additional information on selecting measures, while Park (1992) and Florac (1992) provide frameworks for defining operational definitions for frequently used measures such as size, effort, and schedule.

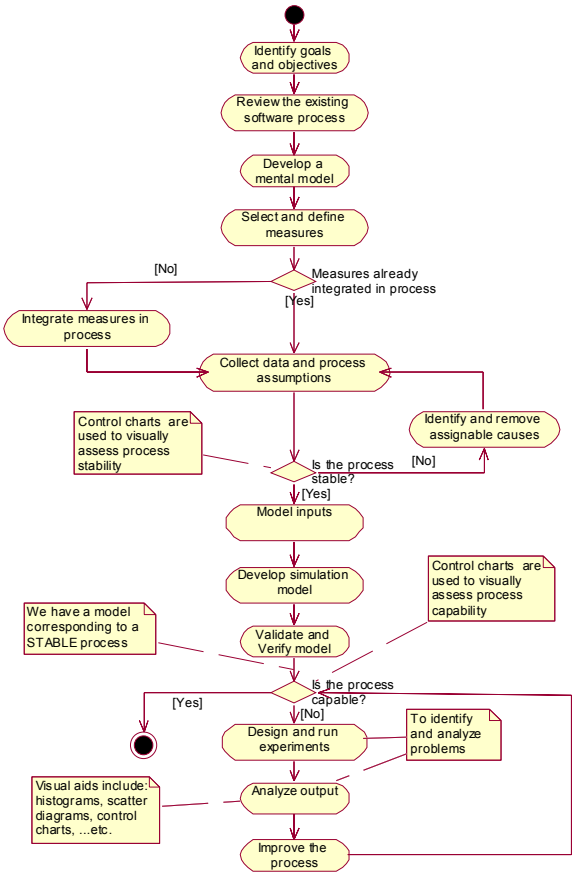

Figure 2. Framework for Integrating SPSM and SPC

After ensuring that all the identified measures are integrated in the current software process, and after collecting data on these measures, SPC, specifically, control charts can then be used to visually assess whether the process is stable (in control) or not. If the process is not stable, then one needs to identify and remove any assignable causes and retest the process for stability.

Once the process is stabilized, one proceeds with input modeling, the purpose of which is to model the stochastic properties of the inputs given the data sets collected on the various process attributes of interest (Leemis 1999). Input modeling is then followed by model development, validation, and verification.

At this point, there is a model that has been developed for a stable process. This model together with control charts can then be used to assess for process capability, and identify and trace elements for improving the software process. Moreover, using a simulation model allows for experimenting and testing of various process configurations in a manner that is either not possible or expensive to conduct in the field.

Case Study

The purpose of this study is to demonstrate the applicability of the proposed framework. Since the development and validation of a simulation model is beyond the scope of this article, this case study utilizes an existing simulation model and focuses on the use of SPC to assess the capability of the underlying software process, i.e., whether the process needs improvements in order to meet the requirements of the customer (project managers, users, …etc.).

Using the simulation model, the case study includes two scenarios. Both scenarios focus on the total effort per iteration as the measure under consideration. Other measures of interest include schedule and quality. Schedule was excluded due to the particular nature of the utilized model where effort and schedule are closely related. On the other hand, quality requires the use of a different type of control charts not described in this article. The primary rational for including the two scenarios is to demonstrate the use of SPC to assess project capability under normal operating conditions (base scenario or scenario #1) as well as assess the effect of changes to such conditions (scenario #2).

The Model

This study utilizes the model developed by Peter Lakey of Cognitive Concepts described earlier (Lakey 2000).

To demonstrate the proposed framework using this model, we added an SPC component for visualizing process performance using control charts as shown in Figure 3.

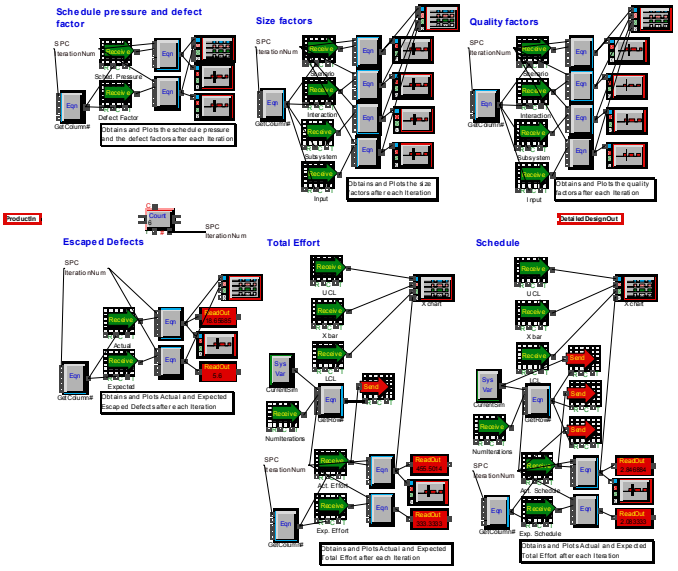

Figure 3. SPC Component for Visualizing Process Performance

Results and Discussion

In the first scenario, the SPC component of the model is used to evaluate a base scenario. This is the scenario representing the current state of the process and is the scenario used as a benchmark for the study. To calculate the control limits the model is run for 10 times, with 6 iterations in each run resulting in a total of 60 observations. Notice that in the absence of a simulation model, this may not be possible. In effect, in the field, there are no guarantees regarding the stochastic properties of the inputs from one iteration to the next, let alone one CSCI to the next. A simulation model allows for calculating more reliable control limits. Figure 4 depicts the X and MR control charts for the total effort per iteration.

In the second scenario the variability of the input scenarios is increased, thereby affecting the capability of the process. In effect, by increasing the variability of the input scenarios the "voice of the process" (wider control limits) is increased as shown in Figure 5.

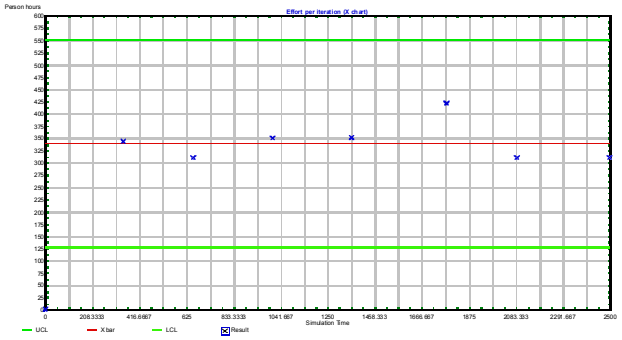

Figure 4a. Base Scenario X Control Chart for the Total Effort per Iteration

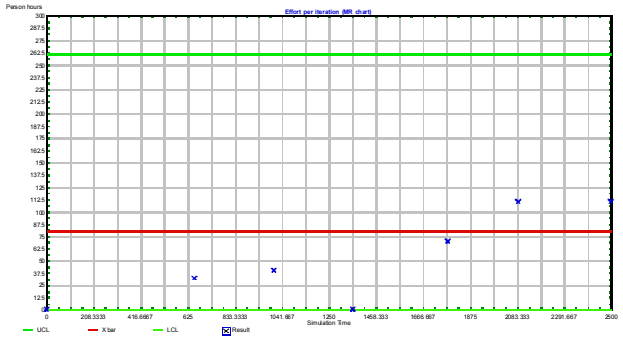

Figure 4b. Base Scenario MR Control Chart for the Total Effort per Iteration

Control charts allow for visualizing such change and comparing the resulting variability with the desired variability "voice of the customer" (variability limits imposed or desired by the project manager). For project managers, a control chart showing values outside the desired limit indicates an underlying process that is incapable of consistently meeting project requirements in term of effort (cost) and schedule and is thereby an indication of higher project risk.

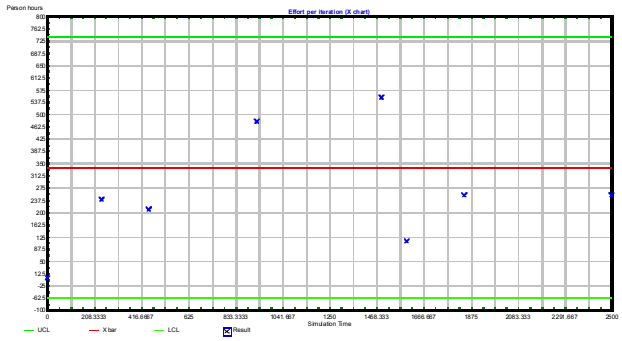

Figure 5a. X Control Chart for an Alternate Scenario

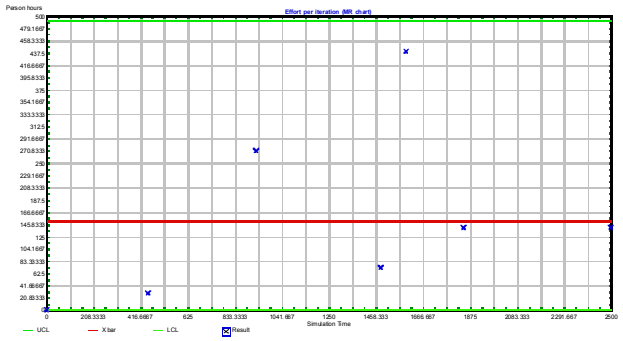

Figure 5b. MR Control Chart for an Alternate Scenario

Summary and Conclusions

While SPSM is used for software process and project management, SPC focuses on assessing the stability and capability of software processes through visualization. Nevertheless, both SPSM and SPC rely on the development of extensive software measurement systems and thereby support the Quantitative Process Management Key Process Area of the Capability Maturity Model.

This paper presents a framework for integrating SPSM and SPC, thereby capturing the synergy between the two approaches for software process improvement as well as maximizes the return on investment in the development and maintenance of extensive software measurement systems. The proposed framework is then used to evaluate the capability of a software process.

Suggestions for future research include investigating the use of SPSM to evaluate the stability (as opposed to the capability) of a software process. In this case, there is a need to be able to explicitly model assignable causes of variation, and the effect of such causes on process capability.

References

Abdel-Hamid, T. "The dynamics of software project staffing: A system dynamics based simulation approach," IEEE Transaction of Software Engineering (15:2), 1989.

Abdel-Hamid, T., and Madnick, S.E. Software Project Dynamics: An Integrated Approach, Englewood Cliffs, New Jersey: Prentice Hall, 1991.

Card, D. "Statistical process control for software," IEEE Software (11:3), May 1994, pp 95-98.

Card, D. "A statistical method for controlling software defect detection process," Computers & Industrial Engineering (37:2). Oct 1999, pp 137-141.

Cho, C. Quality Programming: Developing and Testing Software with Statistical Quality Control, New York, NY: John Wiley and Sons, Inc., 1987.

Collofello, J., Rus, I., Chauhan, A., Houston, D., Sycamore, D., and Smith-Daniels, D. "A System Dynamics Process Simulator for Staffing Policies Decision Support," Hawaii International Conference on System Sciences (HICSS), January 1998.

Curtis, B., Kellner, M.I., and Over, J. "Process Modeling," Communications of the ACM (35:9), September 1992, pp 73-90.

Florac, W. Software Quality Measurement: A Framework for Counting Problems and Defects. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992.

Florac, W., Park, R., and Carleton, "A. Practical Software Measurement: Measuring for Process Management and Improvement," Pittsburgh: PA, Software Engineering Institute, Carnegie Mellon University, 1997.

Florac, W., Carleton, A., Barnard, J. "Statistical process control: analyzing a Space Shuttle Onboard Software process," IEEE Software (17:4), July-August 2000, pp 97-107.

Fuggetta, A. "Software Process: A Roadmap," Proceedings of the conference on the future of software engineering, Limerick, Ireland, 2000.

Ho, Y. and Cao, X. Perturbation Analysis of Discrete Event Dynamic Systems, Boston, Massachusetts: Kluwer Academic Press, 1991.

Kellner, M., Madachy, R.J., and Raffo, D.M. "Software Process Simulation Modeling: Why? What? How?," Journal of Systems and Software (46:2/3), April 1999, pp 1-18.

Krasner, H., Terrel, J., Linehan, A., Arnold, P. and Ett, W.H. "Lessons learned from a software process modeling system," Communications of the ACM (35:9), September 1992, pp. 91- 100.

Lakey, P.B. Discrete Event Software Project Management Tool. Cognitive Concepts, 2000.

Lanphar, R. "Quantitative process management in software engineering, reconciliation between process and product views," The Journal of Systems and Software (12:3), July 1990, pp 243-249.

Lantzy, M. "Application of Statistical Process Control to the Software Process," Proceedings of the ninth Washington Ada symposium on Ada: Empowering software users and developers, McLean, Virginia. New York, NY: ACM Press, 1992, pp 113-123.

Leemis, L. "Simulation input modeling," Proceedings of the 31st Conference on Winter simulation: Simulation---a bridge to the future, Phoenix, Arizona, New York, NY: ACM Press, 1999.

Lewis, N. "Assessing the evidence from the use of SPC in monitoring, predicting & improving software quality," Computers and Industrial Engineering (37:1), 1999, pp 157-160.

Madachy, R. "System Dynamics Modeling of an Inspection-based Process," Proc. ICSE 96, Berlin, Germany, March 25 - 29, 1996, pp 376 – 386.

Montgomery, D.C. Introduction to Statistical Quality Control, Third Edition, John Wiley and Sons, New York, NY, 1997.

Nordgren, W.B. "Steps for proper simulation project management." Alexopoulos, C., Kang, K., Lilegdon, W., and Goldsman, D. Eds., Proceedings of the 1995 Winter Simulation Conference, 1995.

Park, R. Software Size Measurement: A Framework for Counting Source Statements, Pittsburgh: PA, Software Engineering Institute, Carnegie Mellon University, 1992.

Putnam, L. "Trends in measurement, estimation, and control. (making software engineering and management more quantitative)," IEEE Software(8:2), March 1991, pp 105-108.

Quesenberry, C.P. SPC Methods for Quality Improvement, New York, NY: John Wiley and Sons, 1997.

Rus, I., and Collofello, J. "A Decision Support System for Software Reliability Strategy Selection," submitted to the 13th International Conference on Automated Software Engineering, ASE98, 1998.

Tvedt, J.D., and Collofello, J.S. "Evaluating the Effectiveness of Process Improvements on Software Development Cycle Time via System Dynamics Modeling," Computer Software and Applications Conference (CompSAC'95), 1995.

Weller, E. "Practical applications of statistical process control," IEEE Software (17:4), May-June 2000, pp 97-107.