Software Process Monitoring using Statistical Process Control Integrated in Workflow Systems

Marília Aranha Freire, Daniel Alencar da Costa, Eduardo Aranha, Uirá Kulesza
Programa de Pós-Graduação em Sistemas e Computação
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Campus Universitário, Lagoa Nova – 59.078-970 – Natal, RN – Brazil
{marilia.freire, danielcosta}@ppgsc.ufrn.br,{uira, eduardo}@dimap.ufrn.br

***Abstract*— This paper presents an approach that integrates statistical process control techniques with workflow systems in order to achieve software process monitoring. Our approach allows: (i) software process monitoring through the automated metrics collection; and (ii) the statistical process control of software process aided transparently by statistical tools. The use of workflow systems to this integration adds the benefits of statistical process control without the additional effort to integrate and use statistical tools. Our proposal allows project managers to identify problems early during the process execution, enabling quickly reactions (process improvements, training, etc.) to reduce costs and ensure software quality.**

Keywords: Software Process Monitoring, Statistical Process Control, Workflow Systems

I. INTRODUCTION

The increasing complexity of modern software systems has required more well-defined software development processes utilization. Software Process Modeling Languages – SPML support the definition and modeling of software processes, providing functionalities to create and edit the activities flow of their various disciplines, addressing the elements that define a software process, such as tasks, steps, artifacts, and roles [1] [2] [3]. In addition to the benefits and advantages brought by process modeling languages, several recent studies have emphasized the importance of providing mechanisms and tools to support the execution of software processes in order to enable the tracking and monitoring of their activities. Monitoring software projects is important to assess productivity and detect problems that may be occurring and thus promote continuous process improvement.

One approach to support software processes execution is the use of workflow systems. These kinds of systems have been consolidated over the past few years on the business process management domain. The Business Process Execution Language (BPEL), for example, is one of the main industrial results developed by this community. Some recent studies have promoted the integration of approaches and languages for processes modeling and execution [4] [5] [6].

Process control and monitoring is a concept that have been explored and adopted in the industrial scenario. Some decades ago, emerged a statistical technique called Statistical Process Control (SPC) [7] that aims to monitor and quickly detect problems in process execution, allowing fast corrective responses, increasing the quality and productivity of the production processes. This technique has been widely used in industry in general, and its concepts are already being employed in the software industry over the past years [8] [9] [10]. When using this technique, upper and lower control bounds are established for relevant attributes of production processes (time, cost, output quality, etc.), usually based on historical data. Then, if attribute values collected during the process execution are out of the range, these values are called as outliers and they are highlighted for investigation, since they may be caused by problems occurred during the process execution.

This paper presents an approach for integrating statistical process control techniques and workflow systems for monitoring the execution of software processes. Our approach supports: (i) the monitoring of process execution through an automated support for metrics collection and (ii) the statistical process control deployed in workflow systems. As benefits, the approach promotes the monitoring of process stability – ability to be predictable, and process capability – ability to meet specifications, as well as quick responses to outliers, supporting the analysis and decision-making to continuous software process improvement.

The remainder of this paper is organized as follows. Section 2 presents the foundations of process monitoring and statistical process control. Section 3 presents an overview of our approach, which an implementation is presented in Section 4. Section 5 details the approach while illustrating its application and section 6 describes the related works. Finally, Section 7 concludes the paper and provides some directions for future work.

II. BACKGROUND

A. Software Process Monitoring

The automated support for the software development process definition is a concrete reality today. Several approaches have been proposed to facilitate not only the process definition, but also to provide better ways to specify software processes customizations [2] [1] [3]. They provide a set of tools, formalisms and mechanisms used for modeling processes together or even specialize them. Moreover, others research approaches have being proposed, such as DiNitto [11], PROMENADE [12], Chou [13] and UML4SPM [14].

While methodologies, tools and techniques for software processes definition are already consolidated, the environments supporting such software processes execution are still in the process of ripening. The integration of techniques for software processes definition, execution and monitoring has emerged as a way to support the automatic process monitoring, allowing the estimation of activities and

evaluation of team productivity, quality control and process management, which eventually contribute to continuous software process improvement. Software process monitoring is a complex activity that requires the definition of metrics to be collected during execution. The metrics collected at runtime can help the manager during the analysis of the project progress, facilitating the decision-making.

Freire et al [15] presents an approach for software processes execution and monitoring. In that approach, software processes are specified using the Eclipse Process Framework (EPF), which can be automatically transformed into specifications written in the jPDL workflow language [16]. These specifications in jPDL can then be instantiated and executed in the jBPM workflow engine [17]. In addition to supporting the automatic mapping of EPF process elements in workflow elements, the approach also: (i) supports the automatic weaving of metrics collection actions within the process model elements, which are subsequently refined to actions and events in the workflow; and (ii) refines the workflow specification to generate customized Java Server Faces (JSF) web pages, which are used during workflow execution to collect important information about the current state of the software process execution.

Such an approach has been implemented using existing model-driven technologies. QVTO and Acceleo languages were used to support model-to-model and model-to-text transformations, respectively. The approach proposed in this paper is developed based on the work presented in [15].

B. *Statistical Process Control*

The Statistical Process Control (SPC) is a set of strategies to monitor processes through statistical analysis of the variability of attributes that can be observed during the process execution [18] [10] [19] [20]. In terms of SPC, the sources of variations in the process are encompassed in two types: (i) common source of variation and (ii) special source of variation.

The difference between the common and special source of variation is that the former always arises, as a part of the process, while the later is a cause that arises due to special circumstances that are not linked to the process. For example, a common variation on development productivity could be caused by differences in programming experience between developers. On the other hand, a special variation could be caused by a lack of training in a new technology. As special sources of variation are usually unknown, their detection and elimination are important to keep the quality and productivity of the process.

Control charts, also known as *Shewhart charts*, are the most common tools in SPC used to monitor the process and to detect variations (outliers) that may occur due to a special source of variation. The use of control charts can classify variations due to common or to special causes, allowing the manager to focus on variations from special causes. The control chart usually has thresholds at which a metric of the process is considered as an outlier. Those thresholds are called *Upper Control Limit* (UCL) and *Lower Control Limit* (LCL). One pair of UCL and LCL are defined based on the statistical analysis of historical data or based on expert opinion, being used to identify the outliers. However, other pairs can be defined to highlight, for instance, limits that should be respected due to client requirements, such as process productivity (function points implemented per week, etc.) or quality (number of escaped defects, etc.).

Despite the known benefits of using SPC to monitor software processes in order to detect problem during process execution, this task in practice is still very arduous. The software project manager needs to know not only the statistical foundation, but also understand and manipulate statistical tools for the generation of graphics and information necessary for monitoring the processes. To reduce these problems in using SPC, the approach suggested here minimizes the work of the project manager by transparently integrating the use of statistical tools for monitoring the process execution in a workflow system. Furthermore, this automatic control enables continuous recalibration of the control limits, according to the changes occurring in the process performance, and ensures the correct use of statistical techniques.

III. SOFTWARE PROCESSES MONITORING USING SPC INTEGRATED IN WORKFLOW SYSTEMS

A. *Approach Overview*

The approach proposed in this paper is organized in six steps, as shown in Figure 1 and detailed next.

1) *Process Modelling and Definition*

The first approach step is directly related to the software process definition. At this stage one should use a process modeling language (SPL) to specify the process to be monitored. As described in Section IV, the current implementation of our approach provides support to the process definition using the EPF framework. EPF offers features and functionalities for the process definition and modeling through the use of the UMA process modeling language (Unified Method Architecture) [21], which is a variant of the SPEM (Software Process Engineering Meta-Model) [22]. Existing process frameworks such as OpenUP [23] (available in the EPF repository) can be reused and customized to define new software process, reducing the costs of this activity.

2) *Metrics Modelling and Definition*

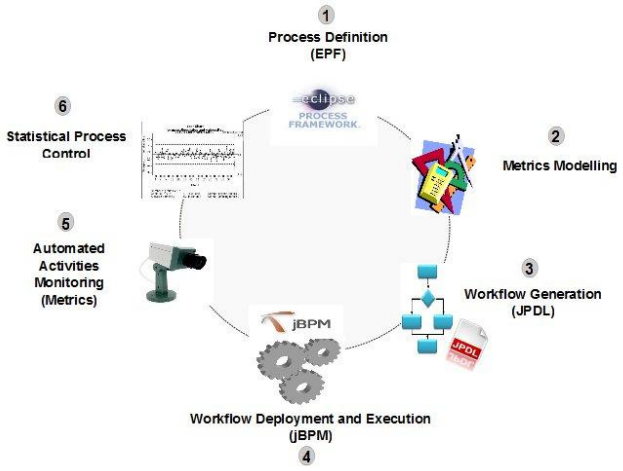


Figure 1 Approach Overview

After the process modeling and definition, it is necessary that process engineers specify the metrics to be collected and monitored during the process execution. Each metric must be defined and associated with one or more activities of the monitored process. The definition of this association is accomplished by specifying the activities that produces each metric, which are modeled using the following meta-model [15].

3) Workflow Generation

To enable the process execution in a workflow system, supporting the automatic collection of defined metrics, a model-to-model (M2M) transformation is performed to generate the JPDL workflow elements from EPF process elements. This transformation is responsible for the generation of actions that allows the automated collection of metrics during the execution of the process activities in the workflow system. In addition, the transformation also generates web pages that will be used for interaction with the process users.

4) Workflow.Deployment and Execution

After generating the workflow and other configuration files, the jBPM workflow engine is used to support the software process execution. It allows project managers to visualize the process execution in real-time and be aware of what is happening during the project development in order to take decisions. They can know, for instance, what activity each member of the project is performing, as well as the status of project activities (performance, quality, etc.) based on the collected metrics.

5) Automatic Activities Monitoring

Monitoring software processes in an automated manner allows greater control of the process by the project manager. This approach, as presented in [15], allows the project manager to automatically monitor the project's progress by viewing web pages and exploring information about the previously defined metrics. These pages provide status information of the process and also the values of the metrics collected dynamically. During the workflow execution, at the end of each task defined in the metrics model, its duration is calculated by performing an action fired after an *end-task* event, and this value is stored and displayed to the project manager.

The project manager can use the collected data to support continuous process improvement and contingency actions, avoiding the occurrence of future problems. Examples of information that can be provided by such metrics are: the execution time of each task step; which task step has a longer duration in the timeline; what is the estimation accuracy; quality or productivity benchmarks, such as function point or use case point per man-hour.

6) Statistic Process Control in Workflow Systems

Our approach promotes the integration of statistical process control into workflow systems. To enable automatic monitoring using SPC, at the end of each monitored task in the workflow, the new value obtained is compared to the last ones in order to determine if it is within the expected range defined for the statistical control. In other words, the observed value for the metric is compared to the LCL and UCL values. If the observed value is lower than the LCL or greater than the UCL, a warning message is issued for the project manager to analyze the cause of this outlier (values significantly different from the expected).

To calculate and implement the control limits, one possible way is the following (other procedures can also be implemented):

(a) If there are not historical data, it can be used expertise to set limits on changes expected for each metric;

(b) If there are historical data, calculates the range as follows:

(i) If the data distribution follow the normal distribution (as verified by statistical tool integrated with monitoring), uses a number of standard deviations, which by default is 3 (includes approximately 99.7% of the population data) and that can be changed by the user to increase or decrease the range. Increasing the range is meant to include more extreme values that could be the problem and will not generate warning. On the other hand, increasing the range reduces the amount of false-positive (indicating problems that are not a problem);

(ii) if the system does not identify the normal distribution, uses the Chebyshev's theorem to calculate the number of standard deviations to cover the same 99.7% of the population data;

In both cases (i) and (ii), the user can make adjustments to the number of standard deviation to be considered. Based on that, the tool indicates the percentage of data encompassed (expected/normal values) according to data distribution observed. The user can also indicate a percentage of interest and the number of standard deviations that should be used will be calculated by the tool.

To facilitate the monitoring and visualization of attributes being monitored, an *X chart* is updated on the screen of the project manager after each new collected value. The outliers are shown in red color in the graphic, representing a possible anomaly.

After the collection of new values, they become part of the historical basis of the process, contributing to the

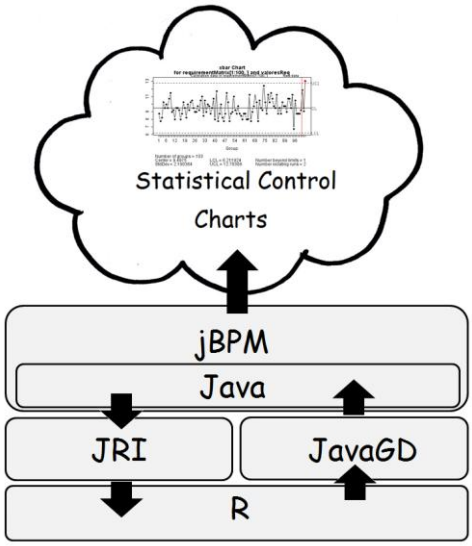


Figure 2 Approach Implementation

adjustment of LCL and UCL values, the known dynamically tuned monitoring sensibility promoted by SPC. Outliers representing problems occurred in the process are not considered for this adjustment, since other occurrences should also be detected.

IV. APPROACH IMPLEMENTATION

The implementation of our approach was accomplished through the integration of the JBPM workflow engine with the computational statistics tool R [24]. This integration is implemented with the *API Java/R Interface* (JRI) [25] that enables *R* function calls within Java code, which is the language used by JBPM. Figure 2 illustrates the approach implementation.

During the M2M and M2T transformations, events and actions handlers are generated and they are responsible for collecting data during workflow execution according to the metrics defined in the model. These action handlers call the R statistical functions to build process control charts (Shewhart, Cusum etc.). However, these functions return specific R charts implementations that need to be treated in the Java code in order to be displayed by jBPM. To enable the Java interpretation of these graphics, an R library called *Java Graphics Device* (JavaGD) [26] is used. This library provides Java canvas objects equivalent to the graphics produced by *R*. Once the canvas objects are obtained, they can be treated and transferred to a view framework such as the *JavaServer Faces* (JSF) [27] used by jBPM.

V. APPROACH IN ACTION

To illustrate the approach proposed in this paper, we present the modeling of a software process and its metrics according to the approach depicted in [15]. The following subsections will describe the approach in action following its respective steps presented in Figure 1.

1) Process and Metrics Modelling (Steps 1 and 2)

The process modeled to illustrate the approach is an OpenUP based process and it is presented in Figure 3. The

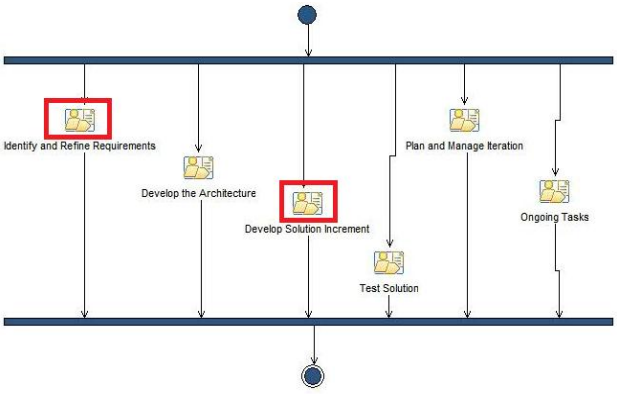


Figure 3: Process Fragment Example

metrics were modeled to monitor the highlighted activities *identify and refine requirements* and *develop solution increment*, aiming collecting the time spent in each activity.

2) Workflow Generation and Execution (Steps 3 and 4)

Once the two model transformations were held and the workflow was deployed in the jBPM engine, the workflow may be calibrated with historical organizational information regarding the metrics before starts the process execution. For example, if the metric is about implementation, then the calibration information would be the time developers take to implement simple or complex functionalities. Also, the limits must be specified and may attend the project requirements of quality or productivity. This is an important step as the approach intends to alert deviations along the process execution and needs to know if a collected metric value is a

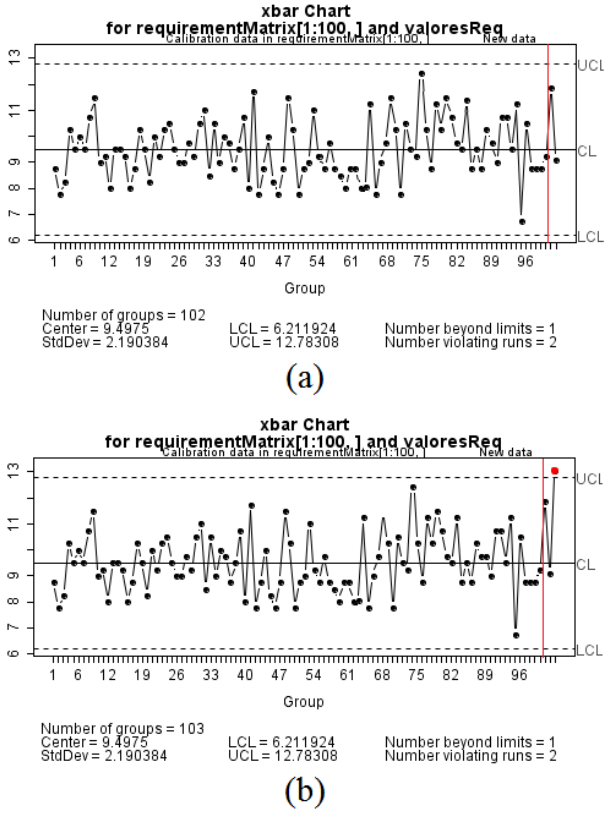
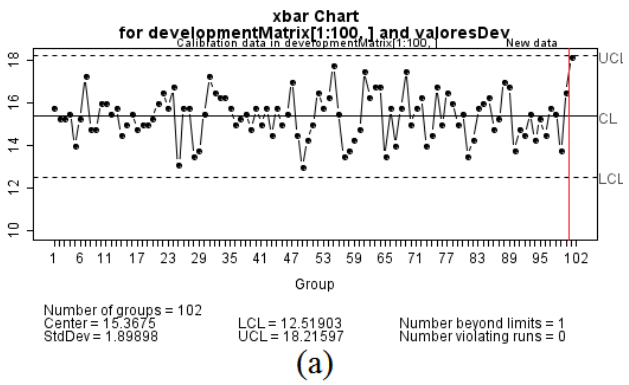
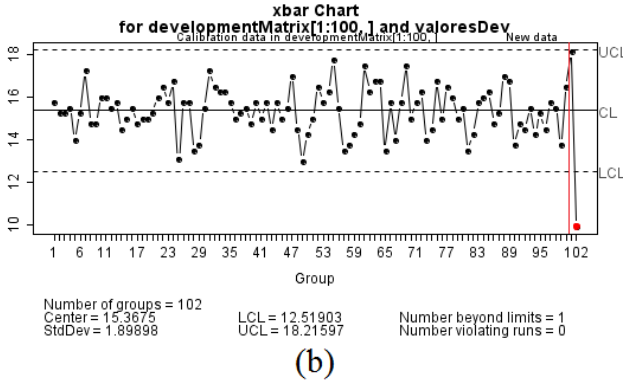


Figure 4: Requirement Elicitation Metric Collection



(a)



(b)

Figure 5: Development Time Metric Collection

deviation indeed.

3) Automatic Monitoring and Statistical Process Control (Steps 5 and 6)

At this stage, the *X chart* is generated and the value of the attribute is graphed on the project manager screen at the end of each monitored activities instances. In the graphic, the x-axis represents the activities instances and the y-axis represents the attribute values collected. Figure 4(a) depicts the first collection of the *time spent per use case* metric after the calibration step. Note that the new collected value fits in the *Upper Control Limit* (about 6 days) and the *Lower Control Limit* (about 13 days). One can also include new limits to represent specific user quality requirement. During the process execution, the limits (UCL and LCL) can be recalculated including the value of the last execution to reflect adjustments made possible in the process at runtime. Figure 4(b) illustrates a case where the collected value supersedes the *upper control limit*. This fact could be explained, for example, as a case when the development company is eliciting requirements to a new business that was not explored in previous projects. In that case, the workflow can trigger a warning notification to interested stakeholders (e.g. an e-mail to a project manager) in order to help planning scope, resources or deadline changes and avoid unwanted situations such as iteration or deployment delays.

Figure 5(a) and Figure 5(b) depict the collect values for the *implementation time per use case* metric. In contrast to Figure 4(b), Figure 5(b) shows one case that the value is beyond the *lower control limit*. This could happen, for example, because of the development of a new functionality

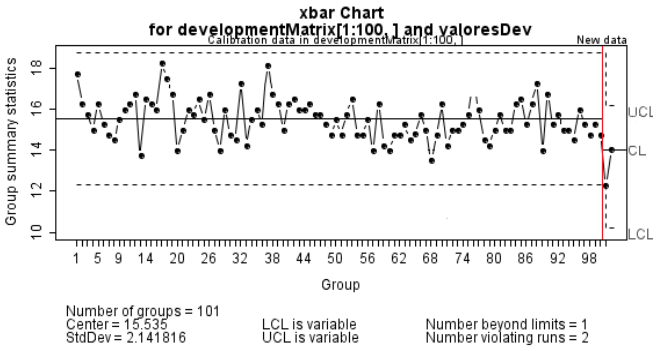


Figure 6: Development production shift

that is very simple compared to the functionalities previously developed (e.g. the implementation of a simple CRUD or a simple login functionality), or a case when the developer did not perform other related fundamental activities like testing or documentation. In some cases, the productivity value is just suffering a natural change and for this reason the control limits must be adjusted accordingly. Figure 6 shows the limits adjustment in which the LCL and UCL were updated to handle the new values of the productivity shift that may be caused by the process improvement or maturation.

The current implementation presented in this work supports only the *X charts*, but the integration between other control charts (e.g. CUSUM) and workflows is also possible.

VI. Related Work

Several studies have been proposing and discussing the use of SPC in software process management to promote continuous improvement. Baldassarre et al [18] discuss the use of SPC from the results found after empirically use this technique in the industry. The paper discusses four synthesized major problems encountered in the software process monitoring showing how SPC can answer each one. It contributes for guiding practitioners towards a more systematic adoption of SPC. Komuro [20] describes experiences of applying SPC techniques to software development processes showing several real examples. The paper points out issues that need to be addressed in order to apply SPC and shows that the key for the successful process improvement is the alignment with business goal.

However, these related works mainly emphasize how to adapt SPC to control software projects and also point out its advantages and disadvantages. None of them focuses on the automated support for monitoring of software processes. Our approach provides support to the automatic and statistical monitoring of software processes in workflow systems through the generation and customization of software processes in workflow systems. The automatic monitoring using SPC transparently during the execution of the process in workflow systems contributes directly to minimizing the complexity issues traditionally involved in work with statistical tools in software projects.

VII. Conclusion

In this paper, we have proposed an approach that integrates statistical process control with workflow systems to monitor software processes. The use of workflow systems

to our integration promotes the collection and analysis of metrics quickly and automatically, adds the benefits of statistical process control without the additional effort to integrate and use statistical tools, and enables the automatic recalibration of the control limits used. Our proposal allows project managers to identify problems early during the process execution, enabling quickly reactions (process improvements, training, etc) to reduce costs and ensure software quality, in other words, allows the fast monitoring. In addition, it also reduces the effort to use automatic monitoring and SPC in an integrated manner.

Currently, our model-driven framework is being increased and adapted to also support process monitoring of software engineering experimental studies. The process monitoring in this domain is fundamental to identify problems that could invalidate all the collected data and study conclusions. If the problem is identified early, actions can be performed to correct the problem, avoiding the loss of all data and giving to the software researcher a chance to better understand the software engineering technique, method or process under investigation.

REFERENCES

[1] IBM. (2010) Rational Method Composer. [Online]. [Online]. http://www-01.ibm.com/software/awdtools/rmc

[2] Eclipse Foundation. (2009) Eclipse. [Online]. http://www.eclipse.org/epf/

[3] IBM. Rational solution for Collaborative Lifecycle Management. [Online]. https://jazz.net/projects/rational-team-concert/

[4] R. Bendraou, J.M. Jezequel, and F. Fleurey, "Achieving process modeling and execution through the combination of aspect and model-driven engineering approaches," in *J. of Softw. Maintenance and Evolution: Research & Practice Preprint.*, 2010.

[5] R. Bendraou, J.M. Jezequel, and F. Fleurey, "Combining Aspect and Model-Driven Engineering Approaches for Software Process Modeling and Execution," in *Proc.Intl. Conf. on Softw. Process*. Vancouver, Canada, 2009, pp. LNCS, vol. 5543, pp. 148-160.

[6] Rita Suzana Pitangueira Maciel, Bruno Carreiro da Silva, Ana Patrícia Fontes Magalhães, and Nelson Souto Rosa, "An Integrated Approach for Model Driven Process Modeling and Enactment," in *XXIII Simpósio Brasileiro de Engenharia de Software*, 2009.

[7] W.A. Shewhart, *Statistical Method from the Viewpoint of Quality Control*. Mineola, New York: Dover Publications, 1986.

[8] J C Benneyan, R C Lloyd, and P E Plsek. (2012, Feb.) Statistical process control as a tool for research and. [Online]. http://qualitysafity.bmj.com

[9] F ZORRIASSATINE and J. D. T. TANNOCK, "A review of neural networks for statistical," *Journal of Intelligent Manufacturing*, pp. 209-224, 1998.

[10] Monalessa Perini Barcellos, Ana Regina Rocha, and Ricardo de Almeida Falbo, "Evaluating the Suitability of a Measurement Repository for Statistical Process Control," *International Symposium on Empirical Software Engineering and Measurement* , 2010.

[11] E. Di Nitto, A. Fuggetta G. Cugola, "The JEDI event-based infrastructure and its application to the development of the OPSS WFMS," *IEEE Trans. Softw. Eng.*, vol. 27, pp. 827-850, 2001.

[12] X. Franch and J. Rib, *A Structured Approach to Software Process Modelling.*: in Proceedings of the 24th Conference on EUROMICRO - Volume 2, 1998, pp. 753-762.

[13] S.-C. Chou, *A process modeling language consisting of high level UML diagrams and low level process language.*: Journal of Object-Oriented Programming, vol. 1, no. 4, pp. 137-163, 2002.

[14] R. Bendraou, M-P. Gervais, and X. Blanc, "UML4SPM: An Executable Software Process Modeling Language Providing High-Level Abstractions," *10th IEEE International Enterprise Distributed Object Computing Conference*, pp. 297-306, 2006.

[15] Marília Freire, Fellipe Aleixo, Kulezsa Uira, Eduardo Aranha, and Roberta Coelho, "Automatic Deployment and Monitoring of Software Processes: A Model-Driven Approach," in *Conference on Software Engineering and Knowledge Engineering*, Miami/Florida, 2011.

[16] JBOSS. jBPM Process Definition Language (JPDL). [Online]. http://docs.jboss.org/jbpm/v3/userguide/jpdl.html

[17] JBOSS. JBoss jBPM. [Online]. http://www.jboss.org/jbossjbpm/

[18] Maria Baldassarre, Nicola Boffoli, Giovanni Bruno, and Danilo Caivano, "What Statistical Process Control can really do for Software Process Monitoring: lessons from the trench," in *Trustworthy Software Development Processes.*: Springer Berlin / Heidelberg, 2009, pp. 11-23.

[19] Nicola Boffolli, G. Bruno, D. Caivano, and G Mastelloni, "Statistical process control for software: a systematic approach.," in *ESEM*, 2008, pp. 327-329.

[20] Mutsumi Komuro, "Experiences of applying SPC techniques to software development processes," in *Proceedings of the 28th international conference on Software engineering*, New York, NY, USA, 2006, pp. 577-584.

[21] Eclipse. Eclipse EPF Project. [Online]. http://epf.eclipse.org/wikis/openupsp/base_concepts/guidances/concepts/introduction_to_uma,_94_eoO8LEdmKSqa_gSYthg.html

[22] OMG. Software Process Engineering Meta-Model. [Online]. http://www.omg.org/technology/documents/formal/spem.htm

[23] IBM Corp. (2009) OpenUP Process Version 1.5.0.4. [Online]. http://epf.eclipse.org/wikis/openup/

[24] Wien, Institute for Statistics and Mathematics of the WU. (2012, Fevereiro) R project. [Online]. http://www.r-project.org/

[25] RForge.net. (2012, Fevereiro) JRI - Java/R Interface. [Online]. http://www.rforge.net/JRI/

[26] S. Urbanek. (2012, Fevereiro) JavaGD. [Online]. http://rosuda.org/R/JavaGD/

[27] Java Community. (2012, Fevereiro) JavaServer Faces. [Online]. http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[28] Weller, E.F.; Bull HN Inf. Syst., Phoenix, AZ , "Practical applications of statistical process control [in software development projects] ," *Software, IEEE*, vol. 17, pp. 48-55, May/Jun 2000.

[29] Eclipse. Acceleo. [Online]. http://wiki.eclipse.org/Acceleo

[30] Geppert A, Tombros D, "Event-based distributed workflow execution with EVE," *Middleware'98 Workshop*, 1998.