

Toward a Process Monitoring Automation: a Proposal

Nicola Boffoli, Danilo Caivano

Department of Informatics - University of Bari - Via E. Orabona 4 - 70126 - Bari – Italy, RCOST – Bari

email: {boffoli, caivano}@di.uniba.it

ABSTRACT

Software process monitoring is a complex activity. The predominant human factors that characterize these processes make automated monitoring a difficult task. This work presents and faces the main issues concerning process monitoring automation and suggests some solutions. Also, the paper presents the use of tools that support the automation of monitoring tasks in accordance to the solution proposed. Finally, the appendix presents an overview of the SPC-Toolkit, a kit of tools available via web that implement the proposed approaches.

KEY WORDS

Statistical Process Control, Six-Sigma, Decision Tables.

1. Introduction

A software process can be considered as a synergic blend of man, machine and methods in working activities whose execution leads to the production of desired outputs, starting from the available inputs [1, 2], let these be products or services. Product quality is tightly related to the quality of the processes used to produce them. Also, within the Software Engineering community a manner for “improving quality” is to improve software processes in order to improve software products. This implies the need for monitoring process execution, highlight process anomalies and quickly react to them.

Unfortunately, software processes are mainly human intensive and dominated by cognitive activities, this makes each process execution a creative and unique activity. The predominant human factor implies differences in process performances and thus multiple outputs. The phenomena known as “Process Diversity” [3, 4], determines difficulty in predicting, monitoring and improving a software process. This makes monitoring activities difficult to automate. For successful process monitoring, among others, there are mainly three problems to address that make the monitoring activity difficult to automate:

- *Problem1: Process Limits.* It is difficult to characterize the behaviour of the monitored process through baselines useful for evaluating its performances. Given the heterogeneity of the operative contexts, the different maturity levels of the processes in use, and the differences of previous knowledge on the monitored process, it is difficult to define an upper and lower limit for process performance variability.

- *Problem2: Process Anomalies.* Difficulty in attributing a meaning to “process anomalies” and in finding conceptual and operative tools for identifying them (what lays behind the concept of anomaly?)
- *Problem3: Sensibility.* Need to adapt the sensibility of monitoring activities to the continuous changing process performances.

The aim of this work is to generalize and put together the experiences collected by the authors in the previous studies [5, 6, 7, 8, 9, 10, 11], to contribute to the discussion on these issues and to propose potential solutions.

In order to do this, this paper presents the following proposal (structured in Figure1):

- *a monitoring model* (section2) to overcome the problem presented above.
- *a monitoring automation* (section3) that automates the execution of process monitoring activities according to the proposed monitoring model

Finally conclusions are drawn.

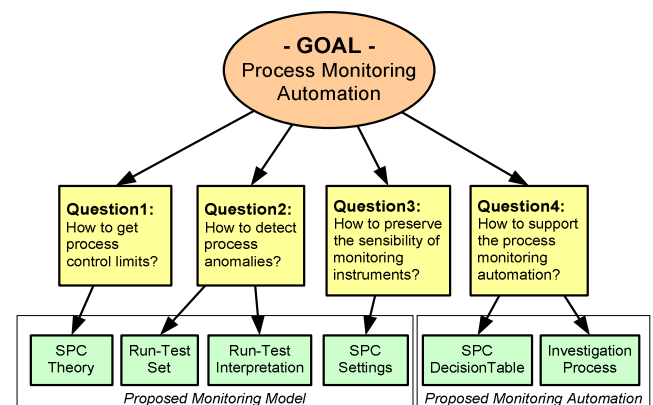


Figure 1: GQM Schema of the proposal

2. Proposed Monitoring Model

In this section a pattern based approach will be followed, where the word “pattern” means a problem-solution couple. According to this, each problem will be presented and discussed and a potential solution will be proposed.

2.1. Problem 1: Process Limits

Software process monitoring is a crucial activity in the context of software process improvement initiatives. Monitoring involves measuring a quantifiable process characteristic (i.e. productivity, defect density, execution time...) over time and pointing out anomalies such as reduction in productivity, an exceptional defect density or an unattended execution time (too high or too slow).

A process must be characterized before being monitored, for example by using a couple of reasonable threshold values, one for the upper and one for the lower process performance limits. When the observed performance falls outside these limits, someone can argue that there is something wrong in the process.

Moreover, process characterization requires past knowledge but often this is not trivial. Many processes are based on paradigms, of a non standard nature, involving the use of COTS, Open Sources, Web Services or other resources that affect the process performances in a non predictable way. The process maturity, the project dimension, the number of people involved in a project and their experience together with thousands of other factors can affect process performances [6, 9, 12]. Thus it is challenging to define an upper and lower control limit able to characterize process behavior. The only way is often to refer to expert judgment and use expert experience for estimating performances. Unfortunately, expert experience not necessarily includes the knowledge about the process in use. This implies that process monitoring seldom occurs.

Proposed Solution. There is a technique for time series analysis known as Statistical Process Control (SPC) [13, 14] that has shown to be effective in manufacturing and recently also used in software contexts. It was developed by Shewhart in the 1920s and then used in many other contexts. It uses several “control charts” together with their indicators to establish operational limits for acceptable process variation. By using few data points, it is able to dynamically determine an upper and lower control limit of acceptable process performance variability. Such peculiarity makes SPC a suitable instrument to face problem 1. Process performance variations are mainly due to: common cause variations (the result of normal interactions of people, machines, environment, techniques used and so on); assignable cause variations (arise from events that are not part of the process and make it unstable). A process can be described by measurable characteristics that vary in time due to common or assignable cause variations. If the variation in process performances is only due to common causes, the process is said to be stable and its behaviour is predictable within a certain error range; otherwise an assignable cause (external to the process) is assumed to be present and the process is considered unstable. A control chart usually adopts an indicator of the process performances central tendency (CL) and upper and lower control limits (UCLs and LCLs). Process performances are tracked overtime on a control chart, and if one or more of the values fall

outside these limits, or exhibit a “non random” behaviour an assignable cause is assumed to be present. Many control charts exist, but in software processes, due to the scarceness of data and since measurements often occur only as individual values, the most used ones are the XmR i.e. individual and moving range charts (Figure 2) [15, 16, 17, 18]. They will be briefly introduced.

In the X chart: each point represents a single value of the measurable process characteristic under observation; CL_X is calculated as the average of the all available values; UCL_X and LCL_X are set at $3\sigma_X$ around the CL_X ; σ_X is the estimated standard deviation of the observed sample of values. Formally, given a set of observations,

$$X = \{x_1, \dots, x_m\}; \overline{mR} = \frac{1}{m-1} \times \sum_{i=1, \dots, m-1} |x_{i+1} - x_i| \quad \text{and}$$

$$3\sigma_X = 2,660 * \overline{mR}. \quad UCL_X = \overline{X} + 2,660 * \overline{mR}; \quad CL_X = \overline{X};$$

$$LCL_X = \overline{X} - 2,660 * \overline{mR}. \quad \text{In the mR chart: each point}$$

represents a moving range (i.e. the absolute difference between a successive pair of observations); CL_{mR} , is the average of the moving ranges; $UCL_{mR} = CL_{mR} + 3\sigma_{mR}$ and $LCL_{mR} = 0$; σ_{mR} is the estimated standard deviation of the moving ranges sample. Formally:

$$UCL_{mR} = 3,268 * \overline{mR}; \quad LCL_{mR} = 0 \quad \text{and} \quad CL_{mR} = \overline{mR}.$$

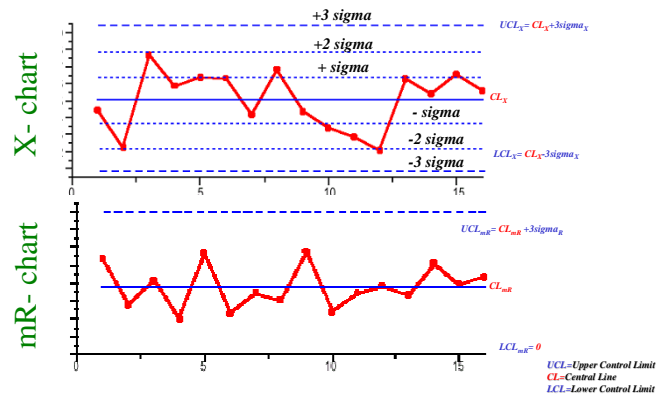


Figure 2: Example of Individual and moving ranges charts (XmR charts)

Sigma is calculated by using a set of factors tabulated by statisticians (for more details refer to [19]) and it is based on statistical reasoning, simulations carried out and upon the heuristic experience that: “it works”. A good theoretical model for a control chart is the normal distribution showed in Figure 3 where: the percentage values reported express the percentage of observations that fall in the correspondent area; μ is the theoretical mean; σ is the theoretical standard deviation. In the $[\mu - 3\sigma, \mu + 3\sigma]$ interval, fall 99.73% (i.e. $2.14 + 13.59 + 34.13 + 34.13 + 13.59 + 2.14$) of the total observations. Thus only the 0,27 % of the observations is admissible to fall outside the $[\mu - 3\sigma, \mu + 3\sigma]$ interval.

If we consider sigma in place of σ , the meaning and rationale behind a control chart results clear. For completeness it is necessary to say that the normal distribution is only a good theoretical model but, simulations carried out have shown that independently

from the data distribution, the following rules of thumb work:

- *Rule1*: from 60% to 75% of the observations fall in the [CL-sigma, CL+sigma]
- *Rule2*: from 90% to 98% of the observations fall in the [CL-2sigma, CL+2sigma]
- *Rule3*: from 99% to 100% of the observations fall in the [CL-3sigma, CL+3sigma]

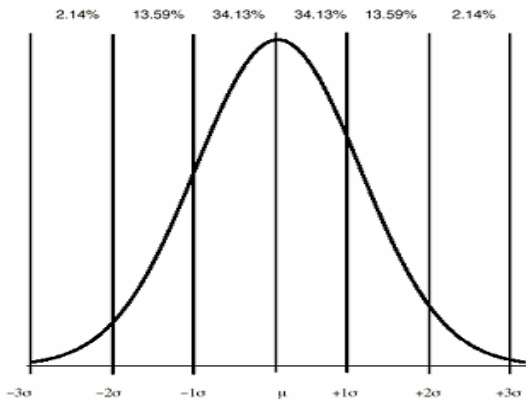


Figure 3: Normal Distribution

The control limits carried out using SPC are based on a process observation and they are expression of it. They are not the results of expert judgement and, furthermore, there is a clear way to obtain them.

2.2. Problem 2: Process Anomalies

In the monitoring activity a fundamental task is to point out process anomalies in order to quickly react to them. But, what are process anomalies and how can we find them? The anomalies are in general some kind of noise in the process performances, the results of an unknown cause in action that implies unattended variation (in better or worse) and thus a lower process predictability. In the software context, relevant examples in this sense are: the introduction of a new case tool that speeds up the development; the use of a new testing or inspection technique that reduces the post release defects; the degradation of system maintainability due to the lack of documentation; the presence of an unknown bug in the hardware platform or in the operative system that leads to a crash of the application and unattended disservices; the involvement of low profile programmers in the project team that determine lower productivity; an unexpected absence of a key person that implies project failure, and so on. Anomalies are consequences of the presence of causes in the process that determine unattended performance variations (in better or worse). The aim of monitoring activity is to point out the anomalies and stimulate the search of the possible causes. The aim of Software Process Improvement is to find the causes, eliminate them if detrimental or, otherwise, make them part of the process. The possible causes of variation can be many such as their effect on process performances and,

consequently, the observable anomalies. A standard mechanism is needed able to characterize an anomaly and, at the same time, to point out it.

Proposed Solution. The proposed solution is based on previous research of the authors who have proposed as solution a set of Run-Tests for selecting SPC indicators. According to the set,, an appropriate Run-Test Interpretation concerning “what is going on” in the process, is also provided.

Run-Test Set. It is a selection of a set of indicators, among those presented in SPC literature, along with their arrangement in logical classes: *sigma*, *limit* and *trend* (Table1).

Table 1: Run-Test Set details

Class	Run-Test	Description	Zone Based
Sigma	RT1: Three Sigma	1 point beyond a control limit ($\pm 3\sigma$)	No
	RT2: Two Sigma	2 out of 3 points in a row beyond $\pm 2\sigma$	Yes
	RT3: One Sigma	4 out of 5 points in a row beyond $\pm \sigma$	Yes
Limit	RT4: Run above or below the CL	7, 8 or 9 consecutive points above or below the centerline	Yes
	RT5: Mixing/ Overcontrol	8 points in a row on both sides of the centerline avoiding $\pm \sigma$ area	Yes
	RT6: Stratification	15 points in a row within $\pm \sigma$ area	Yes
Trend	RT7: Oscillatory Trend	14 alternating up and down points in a row	No
	RT8: Linear Trend	6 points in a row steady increasing or decreasing	No

Note that the sigma tests, whose unreliability is estimated of less than 1% have been supported by other tests (in agreement to software process characteristics) which are not based on sigma, and that represent the probability of a “rare event”.

For example, leaving out rigorous discussions, test4 indicates the probability that 7 to 9 points fall on the same side of the central line. So, considering that the probability that a point falls on one side or another of the central line is of 0.5, the probability that at least 7 points fall on the same side is of 0.0078 (0.5^7). As so, the probability of a “rare event” is approximately that of a sigma test.

Finally, the indicators have been classified as zone based or not. This because the zone based indicators are no longer significant on Moving Range (mR) charts and their use is therefore not advised on such charts.

The next figure shows some patterns of observations detected by each Run-Test of the set:

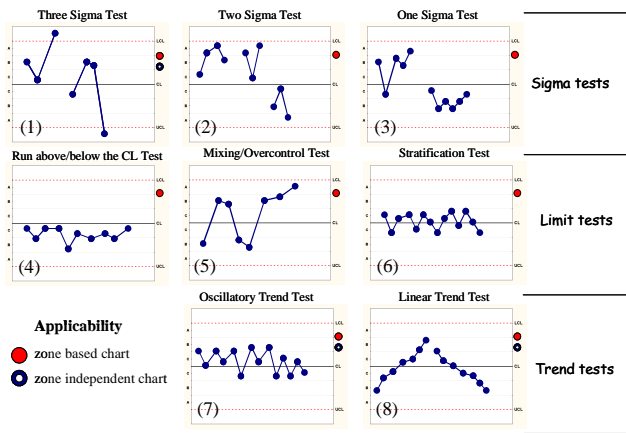


Figure 4 : samples of anomalous patterns detected by SPC

Run-Test Interpretation. It considers the interpretation from the software process point of view, of every test in each class, in particular a class-level description of the run test interpretation is here presented (more details in [5]):

- **Sigma Tests.** They provide an “early” alarm indicator that must stimulate searching possible assignable causes and, if the case, their identification and further elimination. One, Two and Three sigma tests point out a potential anomalous “trend” that “may” undertake assignable causes. In general, due to the high variance in software process especially when we manage individual rather than sample data, the faults highlighted by these tests could be numerous but less meaningful than in manufacturing contexts. For example, in a manufacturing process a party of poor quality raw material may be a potential assignable cause that must be investigated and removed. In software processes a possible assignable cause may be an excessive computer crash due to a malfunctioning peripheral but also to a headache of the developer. Therefore the signals that Sigma tests detect may express a general behaviour determined by an assignable cause or passing phenomena.
- **Limit Tests.** This class of tests point out an occurred shift in process performance. They highlight the need to recalculate the control limits when the actual ones are inadequate, because they are too tiny or larger than required. In software process monitoring and improvement we represent a measurable characteristic that expresses a human related activity outcome (time spent, productivity, defect found during inspection etc.) on a control chart. Thus a “sequence” of points that Limit Tests detect means that something has changed within the process (i.e. performance mean or variability).
- **Trend Tests.** While the previous tests class points out the presence of an occurred shift, this one highlights an ongoing or just occurred phenomena that is resulting in an ongoing shift that needs to be investigated. Typically a failure in this test class can

be the result of both spontaneous or induced process improvement initiatives.

More interpretation details about each run-test are summarized in Table2.

Table 2: Run-Test Interpretation details

Run-Test signals detected	Run-Test Interpretation what happens in the process
No RT	“controlled” Process
RT1	just early alarms
RT2	just early alarms
RT3	just early alarms
RT4	performance mean changed
RT5	performance variability increased
RT6	performance variability decreased
RT7	source of performance variability changed
RT8	ongoing phenomena

Finally, according to the interpretations here described (and more detailed in [5]) we are able to define the following function:

$$\Phi: \{ \text{Run-Test Set} \} \rightarrow \{ \text{Run-Test Interpretation} \}$$

“the signals” “what happens”

such function Φ for each failed run-test assigns the appropriate interpretation and thus it is able to relate the statistical “signal”, detected during monitoring activities, to “what happens” within the process.

2.3. Problem 3: Sensibility

The concept of Process Diversity means that a process varies between different organizations, different projects and also during the execution of a project [3]. This implies estimation model diversity [20]. The need for recalibrating a model is well known in the software estimation community. Process changes impact on the parameters and drivers predicted by the estimation models in use and, as a consequence, they determine estimations inaccuracy. Thus, even when the predictors and corrective parameters are adequately estimated at the beginning of the project, their values tend to change during execution as the context changes. Typical in this sense is the so called “maturity effect”, i.e. an improvement of human performances due to the experience collected during process execution: a better knowledge on the techniques in use, a better confidence with the development tool etc.. Hence, even though a good initial estimation of the process performance is done, it will not prevent estimation errors during project execution.

A further confirmation in this sense comes from COCOMOII [21] that implies the use of different cost drivers for each development process stage (Application Composition, Early Design, Post Architecture). All these considerations imply the difficulty in correctly characterizing process behaviour from the start to the end

- update such relations whenever deep changes in the monitored process lead to new interpretations
- solve conflicts that arise among the “advice” suggested by SPC due to multiple and simultaneous events identified by the approach
- allow the interpretation of two control charts, X-Chart and mR-Chart, which is highly recommended in literature.

Consultation of each SPC-DecisionTable, starting from the Run-Tests (the “signals”) failed in the adopted control charts, extracts a list of advice for the process manager, in particular each consultation extracts:

- according to the Φ function, the information about “what happens” within the monitored process
- according to the ρ function, the actions to carry out (in figure6 in ITALIC character) by the SPC-manager for updating the SPC settings and preserve its sensibility (i.e. recalculating control limits, selecting a different control chart, choosing a new measurement object, continuing the monitoring with the same control limits, ...)

In order to well understand the implemented table, in the following sections the authors firstly introduce the basic concepts of decision tables and then describe the implemented SPC-DecisionTable.

Decision Tables “pills”. A decision table is a tabular representation of a procedural decision situation, where

the state of a number of conditions determines the execution of a set of actions [22, 23, 24]. In general, a decision table is a table divided by a double line, horizontal and vertical, into four quadrants (Figure 5). The horizontal line divides the table in a conditional part (the top part) and an action part (the bottom part). Moreover, the vertical line divides the input values (left side) from the rules and combination of conditional states (right side). The table is defined so that each combination of conditions (conditional states) corresponds to a set of actions to carry out (rule). A tabular representation of a decision situation is characterized by a separation between conditions and actions on one end, and between rules and conditional expressions on the other. Each column of the table (decision column) identifies which actions should (or shouldn't) be carried out for a specific combination of conditional states. The conditional oriented approach of a decision table allows to express all the knowledge related to the problem being considered.

Cond ₁	CS ₁₁						CS ₁₂					
Cond ₂	CS ₂₁	CS ₂₂	CS ₂₃	CS ₂₁	CS ₂₂	CS ₂₃	CS ₂₁	CS ₂₂	CS ₂₃	CS ₂₁	CS ₂₂	CS ₂₃
.....												
Cond _k	CS _{N1}	CS _{N2}	CS _{N1}	CS _{N2}	CS _{N1}	CS _{N2}	CS _{N1}	CS _{N2}	CS _{N1}	CS _{N2}	CS _{N1}	CS _{N2}
Act ₁	X	X	X	X	X	X	X
Act ₂	X	X	.	.	.	X	X	X
.....												
Act _M	.	X	.	X	.	X	.	X	.	X	X	X
	1	2	3	4	5	6	7	8	9	10	11	12

Figure 5: an example of decision-table

Individuals Chart	Null				T1 or T2 or T3			T4			T5			T6			T7		T8	
mR Chart	Null	T1	T7	T8	Null or T1	T7	T8	Null or T1	T7	T8	Null or T1	T7	T8	Null or T1	T7	T8	Null or T1 or T7	T8	Null or T1 or T7 or T8	
1."controlled" process - <i>no action</i>	X	
2.just early alarms - <i>no action</i>	.	X	.	.	X	
3.performance mean changed - <i>identify a new reference set</i>	.	.	.	X	.	.	.	X	
4.performance variability increased - <i>identify a new reference set</i>	X	
5.performance variability decreased - <i>identify a new reference set</i>	X	
6.source of performance variability changed - <i>identify a new measurement object</i>	.	.	X	.	.	X	.	.	X	.	.	X	.	.	X	.	X	.	.	
7.ongoing phenomena - <i>no action</i>	.	.	.	X	.	.	X	.	.	X	.	.	X	.	.	X	.	X	X	

Figure 6: Decision-Table supporting Run-Test Evaluation (compact version)

SPC Implementation. Following to the generic presentation of the decision tables, we now focus our attention on presenting, how they have been implemented in SPC context (Figure 6).

The CONDITION quadrant includes the two different control charts that must be interpreted jointly for identifying appropriate actions. The CONDITIONAL STATES quadrant contains, for each chart, the chart's applicable tests that may fail. So each conditional state means "tests failed" or "no tests failed" in the case of a "null" value. The ACTIONS quadrant lists the set "occurred events" + "actions to carry out". The RULES quadrant contains rules that associate each possible combination of conditional states (tests failed) to a set of appropriate actions.

3.2 SPC-InvestigationProcess

For successful software process monitoring a software process must be constantly monitored and evaluated in order to determine its stability. This allows, on one hand, to quickly react with improvement initiatives in case process performances slow down, on the other hand, to verify the validity of the improvements made. The investigation process here proposed is able to constantly monitor process performance and point out its variation either it be spontaneous or induced by improvement initiatives.

The investigation process (Figure 7) includes:

- *Determination of the Measurement Object:* i.e. selection of both processes to evaluate and measurable characteristics that describe process performances.
- *Determination of the Reference-Set:* i.e. a set of observations of the measurable characteristics that are representative of the process performances and the control limits calculated using such observations.
- *Process Monitoring and Stability Evaluation:* the control limits are fixed, data is tracked over time on the charts, and the tests are executed for each new plotted data point. Whenever a test fails, presence of an assignable cause is investigated.
- *Run-Test Evaluation:* when a run-test fails, it suggests the actions to carry out. Such activity is supported by the SPC-DecisionTable consultation (Figure 6).

4. Conclusion and Future Works

This paper has faced the three main issues that represent a barrier to software process monitoring automation. For each of these, the authors have suggested a solution that represent an important contribution for overcoming them. Also, authors propose two tools that support monitoring automation based on the 3 solutions suggested. In particular, a DecisionTable for formalizing the 3 solutions proposed; the Investigation Process for coordinating and executing the monitoring activities presented.

Nevertheless, a complete automation of software process monitoring still represents an open issue. As discussed in [25], there are many aspects related to software process measurement such as the difficulty of collecting metrics, their reliability, their selection as measurement objects that leave a lot of space for subjective management decisions that can influence the success/failure of monitoring activities. In this sense, the authors are carrying out studies on identifying the most appropriate metrics for the automated approach proposed.

Finally, in the appendix the authors illustrate an overview of a software toolkit, SPC-Toolkit, developed ad hoc to support the execution of the proposed process monitoring automation.

Appendix. Overview of SPC-Toolkit

SPC-Toolkit is a kit of software components able to support the execution of the proposed process monitoring automation (available at: <http://serlab01.di.uniba.it/SPC-Toolkit>).

The architecture follows a Model-View-Controller (MVC) pattern. It has been developed using J2EE technology. In particular:

- Java Server Pages for View area: presentation logic
- Java Servlet for Control area: control logic
- Java-Beans for Model area: business logic

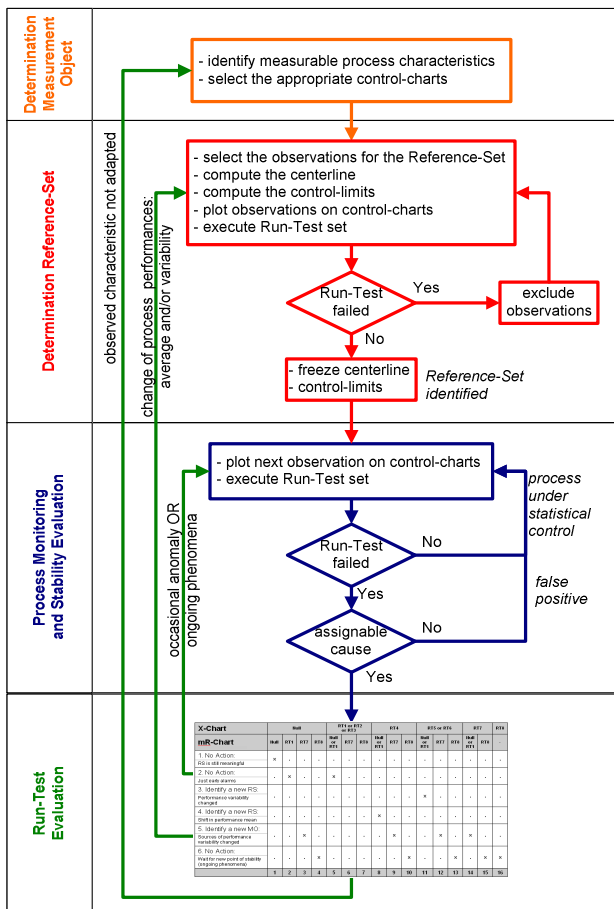


Figure 7: Investigation Process

The toolkit is made up of three different archives:

- SPC-Engine.jar
- SPC-DecisionTable.jar
- SPC-Wizard.war

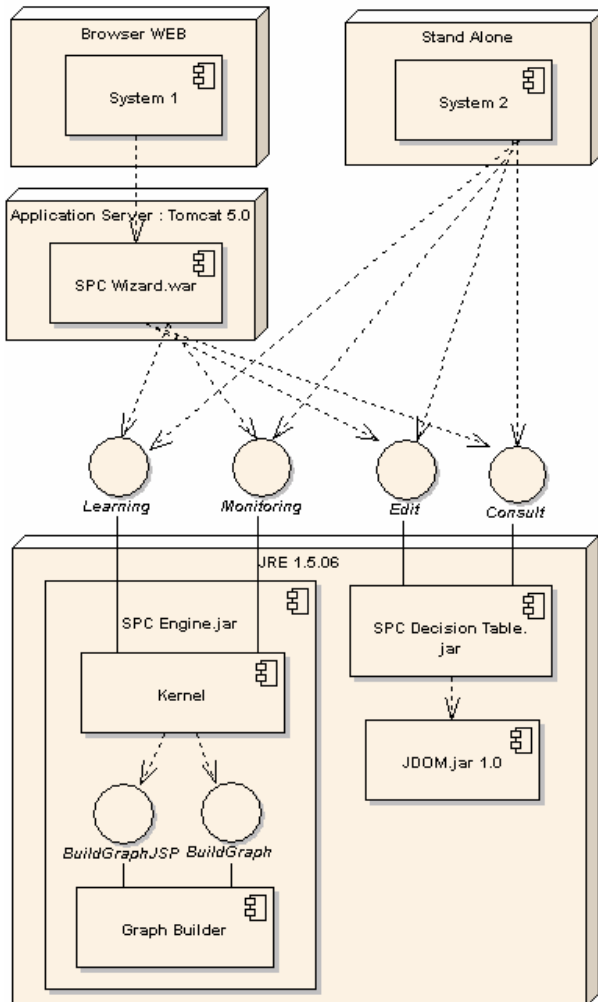


Figure 8: Architecture of SPC-Toolkit

SPC-Engine. jar

According to the first part of the proposed theoretical model, such archive automates the activities of our proposal. In particular the archive is made up of two different components: *Kernel* and *GraphBuilder*.

Kernel. According to our proposal such component coordinates how to use the control charts, the run-tests and their interpretations. Furthermore, it embeds the control flows described in the *Investigation Process*.

GraphBuilder. Such component builds the interactive graphs representing the SPC control charts and their failures. In particular, such graphs are implemented using the *Scalable Vector Graphics* (SVG) technology, that is a graphics file format and Web development language based on XML.

SPC-DecisionTable.jar

Such archive supports the consultation of the SPC-DecisionTable in order to interpret the SPC signals and relate them to “what really happens” within the process. Such activity implements the last step of the *Investigation Process*. Furthermore, there is an editor for updating the SPC-DecisionTable in order to collect and to formalize the experience acquired during its use.

SPC-Wizard.war

It is a web archive that implements the presentation logic for supporting the user interaction with the jar archives. In other words SPC-Wizard contains appropriate Java Server Pages and Java Servlets that, linked to the two jar files, make the whole SPC-Toolkit a web-application.

References

- [1] W.A. Florac and A.D. Carleton, “Measuring the Software Process: Statistical Process Control for Software Process Improvement”, Addison-Wesley, 1999.
- [2] W.A. Shewhart, “Statistical Method from the Viewpoint of Quality Control”, Dover Publications, Mineola, New York, 1939, republished 1986.
- [3] IEEE Software, “Process Diversity”, July – August 2000, 17(4).
- [4] IEEE Software, “The Global View”, March-April 2001.
- [5] M.T. Baldassarre, N. Boffoli, D. Caivano, G. Visaggio, “Managing Software Process Improvement (SPI) through Statistical Process Control (SPC)”, Proc. PROFES, pp30-46, Kansai Science City 5-8 April 2004.
- [6] M.T. Baldassarre, D. Caivano, G. Visaggio, “Software Renewal Projects Estimation Using Dynamic Calibration”, Proceedings of the International Conference on Software Maintenance - ICSM2003- IEEE Computer Society, Amsterdam Holland, Sept 2003.
- [7] Danilo Caivano, “Continuous Software Process Improvement through Statistical Process Control”, Proceedings of the European conference on Software Maintenance and Reengineering, Manchester UK, March 2005
- [8] M.T. Baldassarre, N. Boffoli, D. Caivano, G. Visaggio, “Improving Dynamic Calibration through Statistical Process Control”, Proceedings of the International Conference on Software Maintenance (ICSM), Budapest, September 2005
- [9] D.Caivano, F.Lanubile, G.Visaggio, "Software Renewal Process Comprehension using Dynamic Effort Estimation", Proceedings of International Conference on Software Maintenance, IEEE Computer Society, Florence, Italy, November 2001
- [10] Nicola Boffoli, “Non-Intrusive Monitoring of Software Quality”, Proceedings of the European conference on Software Maintenance and Reengineering, Bari Italy, March 2006
- [11] M.T. Baldassarre, N. Boffoli, D. Caivano, G. Visaggio, “SPEED: Software Project Effort Evaluator based on Dynamic-calibration”, Proceedings of the International Conference on Software Maintenance (ICSM), Philadelphia USA, September 2006
- [12] Clark B.K., “Quantifying the Effects on Effort of Process Improvement”, IEEE Software, 17 (6), 2000, pp.65-70

- [13] W.A. Shewhart, "The Economic Control of Quality of Manufactured Product", D. Van Nostrand Company, New York, 1931, reprinted by ASQC Quality Press, Milwaukee, Wisconsin, 1980.
- [14] W.A. Shewhart, "Statistical Method from the Viewpoint of Quality Control", Dover Publications, Mineola, New York, 1939, republished 1986.
- [15] E.F.Weller, "Practical Applications of Statistical Process Control", IEEE Software, May/June 2000
- [16] J.S. Gardiner and D.C. Montgomery, "Using Statistical Control Chart for Software Quality Control", Quality and Reliability Eng. Int'l, vol. 3, pp. 40-43, 1987.
- [17] R.E. Zultner, "What Do Our Metrics Mean?", Cuttler IT J., vol. 12. no. 4, pp. 11-19, Apr. 1999.
- [18] R. Radice, "Statistical Process Control in Level 4 an 5 Organization Worldwide", Proc. 12th Ann. Software Technology Conf., 2000.
- [19] D.J.Wheeler, D.S.Chambers, "Understanding Statistical Process Control", 2nd ed., SPC Press, 1992
- [20] IEEE Software, Estimation, November – December 2000, 17 (6)
- [21] Bohem B.W. Software Cost Estimation with COCOMO II, Prentice Hall, June 2000
- [22] Ho, T.B., Cheung, D., and Liu, H., "Advances in Knowledge Discovery and Data Mining", 9th Pacific-Asia Conference, Vietnam, 2005
- [23] Vanthienen, J., Mues, C., Wets, G. and Delaere, K., "A tool supported approach to inter-tabular verification", Expert Systems with Applications, 15, pp. 277-285, 1998.
- [24] Maes, R. and Van Dijk, J. E. M., "On the Role of Ambiguity and Incompleteness in the Design of Decision Tables and Rule-Based Systems", The Computer Journal, 31(6), 1988.
- [25] N. Eickelmann, A. Anant, "Statistical Process Control: What You Don't Measure Can Hurt You!", IEEE Software, pp. 49-51, Mar./Apr. 2003.