A Software Measurement Pattern Language for Measurement Planning aiming at SPC

Research Papers Track

Daisy Ferreira Brito[1], Monalessa Perini Barcellos[1], Gleison Santos[2]

[1]Ontology and Conceptual Modeling Research Group (NEMO)
Computer Science Department, Federal University of Espírito Santo
Vitória – ES – Brazil

[2]Graduate Program on Information Systems –
Federal University of the State of Rio de Janeiro - UNIRIO – RJ – Brazil

{dfbrito, monalessa}@inf.ufes.br, gleison.santos@uniriotec.br

Abstract. *The growing interest of organizations in improving their software processes has led them to aim at achieving the high maturity, where statistical process control (SPC) is required. One of the challenges to perform SPC is selecting measures suitable for it. Measures used in SPC can be found in the literature and could be reused by organizations, but information is disperse and non-structured, not favoring reuse. This paper presents MePPLa (Measurement Planning Pattern Language), a pattern language developed based on the findings of a systematic mapping and a survey that investigated measures for SPC. An initial evaluation of MePPLa showed that it favors reuse, contributes to productivity in measurement planning and to the quality of the measurement plan.*

1. Introduction

Software organizations have increased their interest in software process improvement (SPI). There are several patterns and maturity models that support SPI implementation. Some of them, such as CMMI (Capability Maturity Model Integration) [SEI 2010] and MR-MPS-SW (Reference Model for Brazilian Software Process Improvement) [Montoni *et al.* 2009], propose a SPI implementation in levels. At the highest levels (CMMI levels 4 and 5 and MR-MPS-SW levels B and A) SPC is required.

For applying SPC, a good foundation is necessary, i.e., processes characterized by appropriate measures and data that can be used to analyze processes behavior and predictability. However, software organizations face many difficulties related to defining measures suitable for SPC [Barcellos *et al.* 2013; Schots *et al.* 2014].

In the literature there are several works presenting measures that can be used in SPC initiatives. These measures can be reused by organizations intending to carry out SPC. However, selecting which measures are useful in a certain context is not trivial, because information is disperse and non-structured, making access difficult, effort-demanding and sometimes inefficient.

From a set of measures used in SPC initiatives, it is possible to identify some patterns of measures used to monitor certain measurement goals and to perform statistical control of certain processes. A *pattern* can be understood as a successful solution to a recurrent problem [Deutsch *et al*. 2004]. Thus, patterns for measurement planning present solutions to problems related to measurement planning, such as the selection of measures to be included in a measurement plan.

Patterns can be organized into *pattern languages*, which represent patterns and their relationships and also define a process that guides patterns selection and use. The use of pattern languages favors reuse and, consequently, contributes to improve productivity. In addition, since a pattern language provides a mechanism for selecting patterns (for example, a flow that guides the user in patterns selection), even users without much knowledge of the problem domain can be guided to the solution [Falbo *et al.* 2013].

Considering the benefits provided by pattern languages, they can be useful in the software measurement planning context. For example, they can assist organizations in measurement plan elaboration through the reuse of measurement planning patterns. Therefore, we developed MePPLa (Measurement Planning Pattern Language), a pattern language that helps organizations to elaborate measurement plans suitable for SPC. In MePPLa, each pattern is a solution composed of a measurement goal, a process to be submitted to SPC, and measures to analyze the process behavior and monitor the referred goal. The patterns were identified based on the results of a systematic mapping study [Brito and Barcellos 2016] and a survey [Brito *et al.* 2017] that investigated measures used in SPC.

This paper introduces MePPLa and is organized as follows: Section 2 provides the background for the paper talking about software measurement, SPC and pattern languages; Section 3 concerns the research method; Section 4 presents MePPLa; Section 5 addresses MePPLa evaluation; and Section 6 presents final considerations.

2. Background

2.1. Software Measurement and Statistical Process Control

Software measurement is the continuous process of defining, collecting, and analyzing data regarding software processes and products to understand and control them, as well as supply meaningful information to their improvement [Solingen and Berghout 1999]. It is a primary support process for managing projects, and it is also a key discipline in evaluating the quality of software products and the performance and capability of organizational software processes [ISO/IEC 2007].

For performing software measurement, initially, an organization must plan it. Based on its goals, the organization must define which entities (processes, products and so on) are to be considered for software measurement and which of their properties (e. g., size, cost, time, etc.) are to be measured. The organization has also to define which measures are to be used to quantify those properties. For each measure, an operational definition should be specified, indicating, among others, how the measure must be collected and analyzed. Once planned, measurement can start. Measurement execution involves collecting data for the defined measures, storing and analyzing them. Data analysis provides information to decision making, supporting the identification of appropriate actions. Finally, the measurement process and its products should be evaluated to identify potential improvements [Barcellos *et al*. 2010].

There are some approaches in the literature to support measurement planning. One of the best known is GQM (Goal-Question-Metric) [Basili *et al.* 1994], which represents a systematic approach for tailoring and integrating goals to software processes, products and quality perspectives of interest, based upon project and organizational specific needs. To put it simply, GQM states that from goals it is possible to identify information needs that can be met by measures. By following this idea, from

their goals, organizations can derive information needs and define measures able to meet them.

Depending on the organization's maturity level, software measurement is performed in different ways. At initial levels (such as CMMI levels 2 and 3), measurement consists in collecting data from projects and comparing them with their corresponding planned values. At high maturity levels (such as CMMI levels 4 and 5), it is also necessary to carry out SPC to get to know processes behavior, determine their performance in previous executions, and predict their performance in current and future projects, verifying if they can achieve the established goals [Barcellos *et al.* 2013].

SPC uses a set of statistical techniques to determine if a process is under control, considering the statistical point of view. A process is under control if its behavior is stable, i.e., if its variations are within the expected limits, calculated from historical data [Florac and Carleton 1999]. The behavior of a process is described by data collected for measures related to it [Barcellos *et al*. 2013].

A process under control is a stable process and, as such, has repeatable behavior. So, it is possible to predict its performance in future executions and, thus, to prepare achievable plans and to improve the process continuously. On the other hand, a process that varies beyond the expected limits is an unstable process and the causes of these variations (said special causes) must be investigated and addressed by improvement actions, to stabilize the process. Once the processes are stable, their levels of variation can be established and sustained, being possible to predict their results. Thus, it is also possible to identify the processes that can achieve the established goals and the processes that are failing in meeting the goals. In this case, actions to change the process to make it capable should be carried out. Stabilizing their critical processes is a characteristic of high maturity organizations or organizations that are looking forward to achieving the highest maturity levels [Florac and Carleton 1999].

2.2. From Patterns to Pattern Languages

According to Deutsch *et al*. (2004), *patterns* are vehicles for encapsulating knowledge. They allow capturing what must be done to solve a given problem. In order to use a pattern it is necessary to recognize an opportunity to apply it, that is, it is necessary to understand and recognize the context in which the recurrent problem treated by the pattern occurs and to identify if the problem to be addressed is an instance of the recurrent problem.

Buschmann *et al.* (2007) point out that many patterns found in the literature are related to others, but most fail to explain how patterns can be combined to form solutions to larger problems than those treated by each pattern individually. The solution descriptions tend to concentrate on applying the patterns in an isolated way. Thus, they do not properly solve problems that arise when several patterns are applied in combination. This situation is problematic, since features introduced by one pattern may be required by others. Therefore, a broader context is needed to describe the larger problems that can be solved and those that can arise when patterns are used in combination. This context can be provided by what is known in Software Engineering as *Pattern Language* (PL).

A software pattern language is a network of interrelated patterns that defines a process for systematically solving problems related to software development [Deutsch 2004]. This process aims to provide global support for patterns use in order to solve problems related to a technical domain or some specific application. This holistic view

should provide explicit guidance on the problems that may arise in the domain, inform the possible means of solving them and suggest one or more patterns to solve each specific problem [Falbo *et al.* 2013].

Pattern languages reflect the fact that patterns tend to be strongly related and it is difficult, or even impossible, to use them in isolation [Falbo *et al.* 2013]. In this sense, in order for a PL to be effective in its goal of guiding the user on patterns application and to properly address the combined application of several patterns, it is necessary that relations between patterns are defined and made explicit in the PL.

Visual notations can be used to graphically represent PLs. The purpose of adopting visual notations is to give an overview of the patterns and their relationships, contributing to a holistic understanding of the PL and assisting in patterns selection [Quirino 2016]. In this sense, Quirino (2016) proposes a cognitively rich visual notation named OPL-ML (Ontology Pattern Language Modeling Language) for representing pattern languages in the ontology engineering domain. Although proposed to represent ontology pattern languages, OPL-ML can also be applied to represent PLs in other domains, such as Software Engineering.

Regarding PLs related to software measurement, there are some proposals in the literature. Andrade *et al.* (2010), for example, present a pattern language to support estimates for software projects in micro and small enterprises. The language is intended to assist the project manager in estimating a software project during the project planning phase. For this, the patterns give guidance on what managers must do to collect actual data and use them to support estimates and decision making. Based on the work by Andrade *et al.* (2010), Braga *et al.* (2012) propose a pattern language for estimates in agile projects. The language consists of eight patterns that can help agile teams to perform estimates for agile software projects. Figure 1 illustrates the pattern language proposed by Braga *et al*. (2012).

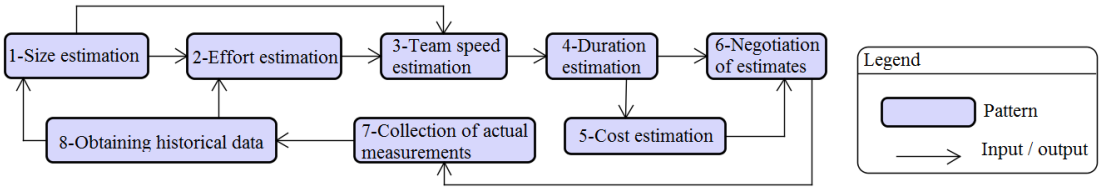

Figure 1 - Pattern Language for Software Estimates in Agile Projects [Braga *et al.* 2012].

3. Research Method

The research method adopted in this work followed the Design Science Research paradigm [Hevner 2004]. According to Hevner (2007), the Design Science paradigm considers three cycles of closely related activities: Relevance, Design and Rigor.

The *Relevance Cycle* starts the research and defines the problem to be addressed, the research requirements and the criteria to evaluate the results [Hevner 2007]. The problem addressed in this work refers to the difficulty faced by software organizations when planning measurement for SPC, especially when selecting the measures to be considered. In order to understand the state of the art about measures used in SPC initiatives or suggested for it, we investigated the literature through a systematic mapping study [Brito and Barcellos 2016]. From the results we got some perceptions: *(i)* prevalence of defect-related measures; *(ii)* lack of concern about relations between measures; *(iii*) lack of approaches to support measures selection; and *(iv)* lack of

operational definitions for the measures. Considering the problem, the gaps identified from the systematic mapping and the benefits of using pattern languages reported in the literature, we noticed that the use of a measurement planning pattern language could help organizations in developing measurement plans for SPC. Thus, we decided to propose a pattern language for this purpose. As the main requirements, we established that the pattern language should be able to guide users on selecting patterns to be included in a measurement plan and should also represent relationships between measures. In addition, the pattern language should be graphically represented. As criteria for results evaluation, it should be considered the viability of using the pattern language and its usefulness.

The *Design Cycle* refers to development and evaluation of artifacts or theories to solve the identified problem [Hevner 2007]. As previously mentioned, the problem that motivated this work is the difficulty of developing measurement plans for SPC. In order to address it, we propose the use of a measurement planning pattern language. The artifact proposed in this work is MePPLa (Measurement Planning Pattern Language), a pattern language that provides patterns related to processes and composed of measurement goal, information needs and measures suitable for SPC, which can be reused to help measurement plans elaboration aiming at SPC. The patterns of MePPLa were identified from the results of the systematic mapping [Brito and Barcellos 2016] and from a survey conducted with some professionals to obtain information about the state of the practice on measures used in SPC [Brito 2017]. To support the use of MePPLa, we developed a tool. For evaluating MePPLa, we performed an experimental study in which participants used the pattern language and provided their perceptions about it.

Finally, the *Rigor Cycle* refers to knowledge use and generation. Rigor is achieved through the adequate application of existing foundations and methodologies [Hevner 2004]. A knowledge base is used to support the research and the knowledge generated by the research contributes to the growth of the knowledge base [Hevner 2007]. In this work, the main theoretical foundations are knowledge related to secondary studies (particularly, systematic mappings), software measurement, statistical process control, pattern languages, and evaluation methods (particularly, experimental study and survey). The main contributions to the knowledge base are the following: (*i*) MePPLa, which can support organizations in measurement planning for SPC and can be evolved to incorporate new patterns; *(ii)* the systematic mapping that consolidates information about measures used in SPC initiatives or suggested for it, providing a panorama of the research topic and indicating possible future research [Brito and Barcellos 2016]; (*iii*) the survey conducted with professionals with experience in SPC, providing information about measures that have been used in SPC initiatives in Brazilian organizations [Brito 2017]; and *(iv)* the tool developed to support the use of MePPLa.

4. MePPLa: A Measurement Planning Pattern Language

MePPLa is a pattern language with the purpose of helping measurement planning for SPC. It currently comprises 26 patterns, of which 12 are related to the Project Management process, 6 related to the Codification process and 8 related to the Testing process. The MePPLa application context encompasses organizations that intend to submit these processes (or some of them) to SPC, including CMMI level 3 or MR-MPS-SW level C organizations that want to implement SPC practices for achieving high maturity. In order to use MePPLa, the organization must have established its

organizational (business) goals, and identified the processes related to them and that will be submitted to SPC. Thenceforth, the organization can use MePPLa to select, for each process, the patterns to be included in the measurement plan.

As previously mentioned, MePPLa patterns were identified from investigative studies into the literature [Brito and Barcellos 2016] and practice [Brito 2017]. These studies investigated measures used in SPC as well as measurement goals and processes related to the measures. As a result, 110 measures, 18 processes and 49 goals were identified. Based on this set of measures, processes and goals, and taking the purpose and context of use of MePPLa into account, we selected the Project Management, Coding and Testing processes as the ones to be treated in the first version of MePPLa. The Coding and Testing processes were both identified in the survey and in the literature, being the Testing process the most cited in both studies. The Project Management process, in turn, was identified only in the literature, but since it is considered a suitable process for SPC, because it is usually a critical process and executed in all projects, it was also selected to be treated in MePPLa. Once the processes were selected, the measurement goals and measures related to them were analyzed and patterns were extracted. The MePPLa patterns follow the GQM structure [Basili *et al.* 1994], thus a measurement planning pattern has a measurement goal, information needs derived from it and measures to meet the information needs (including the operational definition of the measures).

During patterns extraction, we took some actions aiming to define patterns suitable for the MePPLa scope. For instance, we disregarded very general measurement goals such as *gaining market competition*, which is more a business goal than a measurement goal, and *understanding the software processes performance*. We also unified some similar or complementary measurement goals. For example, the goals related to the Testing process *Reducing the number of escaped defects* and *Improving defect detection* were unified in *Improving test effectiveness*. As for the measures, we disregarded very specific measures with little potential for reuse, such as *Percentage of effort saved by process automation* and *Rate of effort spent on internal revision of tests development*.

Concerning the Project Management process, we extracted patterns related to project planning and project monitoring and control. For the first, we identified patterns to deal with estimates considering different granularities. We extracted patterns related to effort, duration and cost estimates for activity, phase and process. Thus, when the pattern language is applied, the user can choose which granularities s/he wants to consider in the measurement plan. For the Codification process, we identified patterns containing measures related to productivity, defect density and rework. As for the Testing process, we extracted patterns related to delivery defects, detected defects and effort spent on tests activities.

In order to refine the alignment between processes and related patterns, we decomposed some processes. The Project Management process was decomposed into Project Planning and Project Monitoring and Control, and the Testing process was decomposed into Test Preparation and Test Execution.

For each identified pattern, we established a detailed description. Table 1 shows, as an example, the description of the *Test Preparation Productivity* pattern. Due to space limitation, we omitted part of the operational definition of the base measures.

Table 1 - Description of the Tests Preparation Productivity Pattern.

Test Preparation Productivity

Name: Test Preparation Productivity
Process / Sub-process: Tests / Test Preparation
Measurement Goal: Improve productivity in test preparation
Information Needs: What is the productivity in tests preparation?
Measures: Test Preparation Productivity, Number of Elaborated Test Cases, Test Preparation Effort.
Operational Definition of Measures:

Derived measure	Test Preparation Productivity
Mnemonic	TPP
Description	Measure used to quantify productivity in the tests preparation, that is, the ratio between the number of test cases produced and the effort spent on test preparation.
Entity	Tests Preparation sub-process
Property	Productivity
Scale	Positive real numbers accurate to two decimal places
Measurement unit	Test Cases / Man-Hour
Formula	(NPTC/ TPE)
Measurement Procedure	Calculate productivity in test preparation using the formula for calculating the measure.
Measurement Periodicity	The measurement must be performed for each execution of the Test Preparation sub-process.
Measurement Responsible	*<<indicate the role responsible for collecting data for the measure. It is recommended that the measurement responsible is the data provider (i.e., a role involved in tests preparation)>>*
Measurement Moment	At the end of each execution of the Test Preparation sub-process.
Measurement Analysis Procedure	For process behavior analysis (organizational context): - Represent in a control chart values collected for the measure in several projects. - Obtain the process control limits and analyze the process behavior: (i) If the values pass in the stability tests, then the process is stable and a baseline can be established. Stability tests [WHEELER and CHAMBERS 2010]: Test 1: There is at least one point outside 3σ; Test 2: there are at least two out of three successive points at the same side and at more than 2σ from the central limit; Test 3: there are at least four out of five successive points at the same side and at more than 1σ from the central limit; Test 4: There are at least eight successive points at the same side. (ii) If the values do not pass in the stability tests, the process is unstable. It is necessary to investigate the special causes, identify corrective actions and execute them. For quantitative project management (project context): - Represent in a control chart values collected for the measure in the project. - Analyze the process behavior considering the organizational behavior expected for it (i.e., by using the process baseline as reference). (i) If the values pass in the stability tests considering the process baseline as reference, then the process behaved according to the behavior expected for it in the organization. (ii) If the values do not pass in the stability tests considering the process baseline as reference, then the process did not behave according to the behavior expected for it in the organization. It is necessary to investigate the causes, identify corrective actions and execute them.
Analysis Periodicity	In the project context, the analysis must be performed once to each execution of the Test Preparation sub-process. In the organizational context *<<indicate the periodicity based on a time period (e.g., fortnightly) or on an amount of new data collected (e.g., each 4 new values collected)>>*
Analysis Responsible	*<<indicate the role responsible for analyze data collected for the measure>>*
Analysis Moment	In the project context, the analysis must be performed after each execution of the Test Preparation sub-process. In the organizational context, analysis must be performed during the activity in which organizational process behavior analysis is done *<<indicate which is the activity in your organization >>.*
Base Measure 1	Number of Prepared Test Cases
Mnemonic	NPTC
Description	Measure that quantifies the number of test cases elaborated in the test preparation.
Entity	Tests Preparation sub-process
Property	Number of test cases
Scale	Positive real numbers accurate to two decimal places
Unit of measurement	-
Formula	-
Measurement Procedure	Obtain and record the number of test cases prepared in the test preparation.
(...)	(...)
Base Measure 2	Test Preparation Effort
Mnemonic	TPE
Description	Measure that quantifies the test preparation effort.
Entity	Tests Preparation sub-process
Property	Effort
Scale	Positive real numbers accurate to two decimal places
Unit of measurement	Man-hour
Formula	-
Measurement Procedure	Obtain and record the effort spent on the preparation of the tests, i.e., test cases preparation.
(...)	(...)

Related Patterns: Test Preparation Efficiency.

According to Barcellos *et al*. (2013), ideally, the operational definition of a measure used in SPC should include: name, description, mnemonic, measured property, measured entity, scale, measurement unit, formula, measurement procedure, measurement responsible, measurement periodicity, measurement moment, measurement analysis procedure, measurement analysis responsible, measurement analysis periodicity, and measurement analysis moment (for base measures that compose derived measures and are not directly analyzed, it is not necessary to provide information about measurement analysis). Considering that MePPLa contains patterns to be used by different organizations, there are pieces of information that cannot be predefined, and, thus, are not included in the patterns descriptions and must be completed when applying a pattern (i.e., when including it in a measurement plan). For these pieces of information, the pattern provides some guidelines (shown in Table 1 in *italics* and delimitated by << >>) aiming to help organizations to complete the operational definition of the selected measures.

Once the patterns were defined, we developed models for graphically representing the pattern language. We used OPL-ML [Quirino 2016] as modeling language for representing MePPLa. According to OPL-ML, a pattern language must have the structural and behavioral perspectives separately represented. Thus, MePPLa is composed of two types of models, the *structural model*, which represents the patterns and the structural relations between them, and the *behavioral model*, which describes the process for selecting and applying the patterns.

The *structural model* provides information about structural relations (e.g., dependence and composition), which are especially useful during measurement analysis, since they reveal related measures and goals that impact on others. This model can also be useful in measurement planning by helping identify which patterns can be selected for a better analysis of goals achievement and identification of causes that may be interfering with it. During MePPLa development, the structural model was useful for us to develop the behavioral model, since the structural model indicates the dependencies that must be considered in the flows that guide patterns selection. Figure 2 presents the structural model containing patterns related to the Testing process.

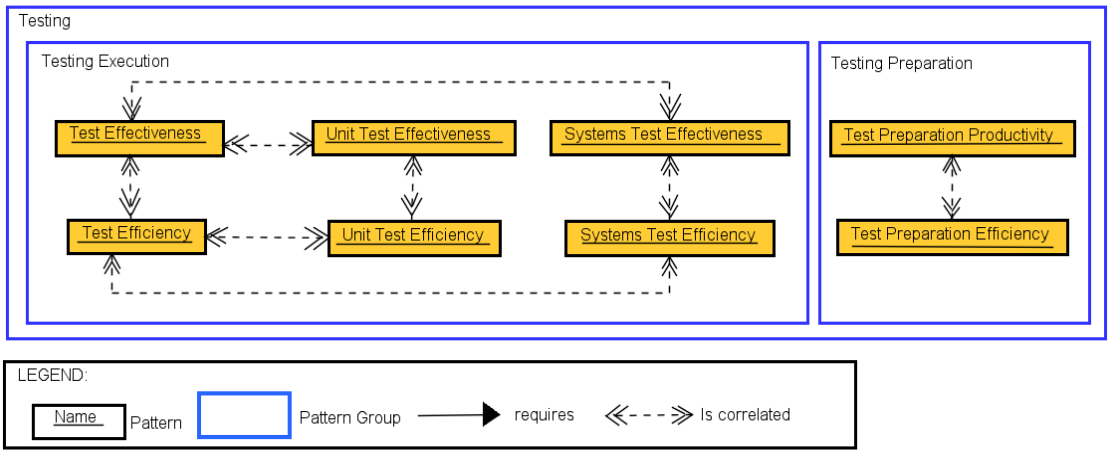

Figure 2 - Structural model of the Testing Pattern Group

In MePPLa two types of structural relations are used: dependence, when a pattern requires the application of another, and correlation, which was added to the notation provided by OPL-ML to indicate patterns whose measures are correlated, but the patterns are not dependent. For example, the measure *requirements change rate* is

correlated to *amount of rework*, since many changes in requirements may impact the amount of rework. However, a pattern including the measured *amount of rework* does not depend on the application of the pattern containing the measure *requirements change rate* to be applied (and vice-versa). As Quirino (2016) suggests, patterns can be organized into groups. In MePPLa, patterns were grouped considering the processes and sub-processes to which they relate.

As shown in Figure 2, there are only correlation relations between patterns from the Testing group. For example, *Test Efficiency* is correlated to *Test Effectiveness*; since improvements in test effectiveness may impact test efficiency (more effective tests may require more effort). The structural model for Testing has two subgroups, one related to the Test Execution sub-process and another related to the Tests Preparation sub-process.

The *behavioral* model (or process model) has two formats: the black box format, which provides a general view of the pattern language from the behavioral perspective; and the detailed format, which provides a detailed view of the pattern language process, containing the flows that guide patterns application. Both formats must be understood as a process to be followed step by step, from an entry point to an end point. Figure 3 shows the behavioral model of MePPLa in the black box format. In this format it is possible to see the processes treated in MePPLa, since the pattern groups refer to the processes addressed by MePPLa. As the name suggests, in the black box format it is not possible to visualize the patterns and flows within each group.

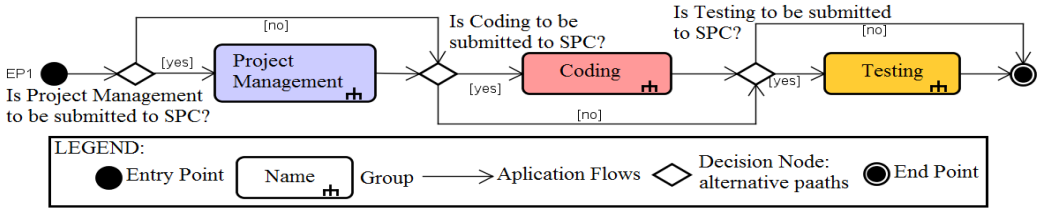

Figure 3 – Black-box format of the behavioral model

For each pattern group (i.e., for each process) we developed a detailed behavioral model presenting the group internal content, i.e., its patterns and flows between them. Figure 4 shows the detailed behavioral model of the Testing process. Due to space limitations, detailed behavioral models of other processes are not presented in this paper.

The detailed behavioral models are consistent with the structural model. Thus, the patterns are grouped according to the processes and sub-processes to which they relate. Patterns are connected by flows to guide users in patterns selection, respecting the relationships between the patterns defined in the structural model (for example, if in the structural model a pattern A depends on a pattern B, in the behavioral model the user is guided to apply B before A). To navigate the model, the user must choose an entry point according to the measurement goal to be considered in the measurement plan. For example, in Figure 4, if the measurement goal *Improve test effectiveness and efficiency* is relevant, the user should start navigation from EP1. If this measurement goal is not relevant and *Improve test preparation performance* is, then the user should start navigation from EP2. Note that if the user starts navigation from EP1, s/he will also be guided to the patterns related to *Improve test preparation performance* and, by following the flows, s/he can decide if they are relevant or not. From an entry point, the user must follow the flows and select the patterns according to the goals that s/he wants to treat in the measurement plan. When a pattern is applied, it means that its description is to be included in the measurement plan.

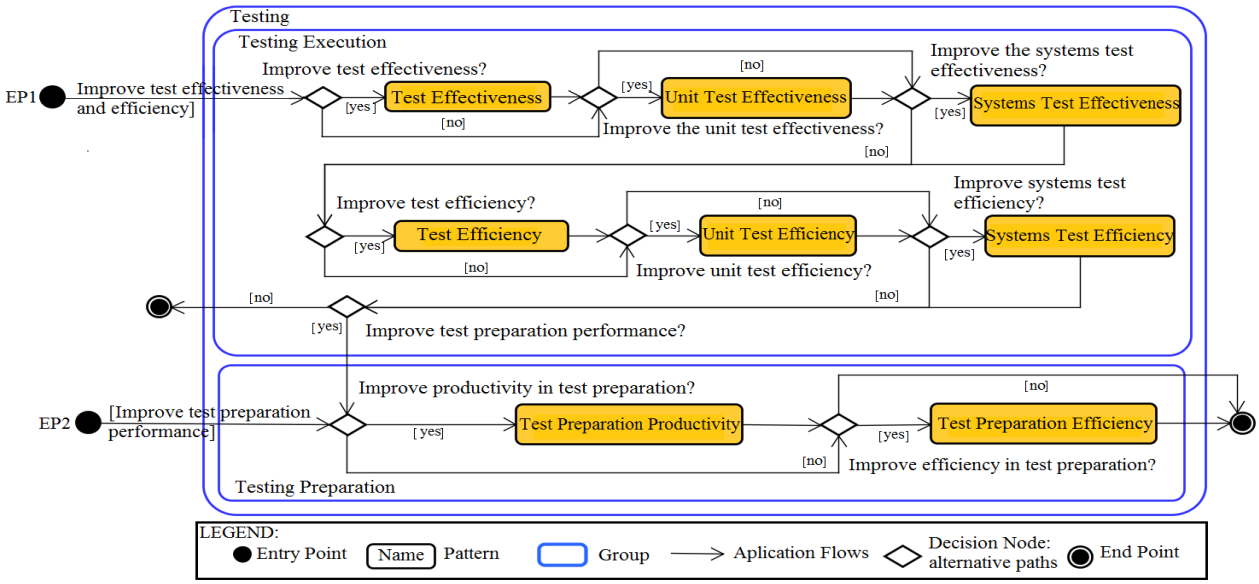

Figure 4 - Behavioral model of the Testing Pattern Group.

The full specification of MePPLa can be found in [Brito 2017]. To support the use of MePPLa, we developed a tool[1] that allows users to navigate the behavioral models and select the patterns they want to include in their measurement plans. As a result, a file containing the detailed descriptions of the selected patterns is produced. Information recorded in the file can be extracted and included in the organization's measurement plan or organizations can edit the file and turn it into a measurement plan.

5. MePPLa Evaluation

As a first evaluation of MePPLa, we conducted a study to evaluate whether MePPLa is useful to software measurement planning for SPC and whether its use is feasible. Using the GQM approach [Basili *et al*. 1994] this goal is formalized as follows: *Analyze* MePPLa, *with the purpose of* evaluating the pattern language, *with respect to* its usefulness in software measurement planning aiming at SPC and feasibility of use, *from the point of view of* software measurement professionals, *in the context of* software projects. To analyze the results, we considered the following indicators: (i) adequacy of the result obtained from the use of MePPLa; (ii) MePPLa usefulness; and (iii) benefits provided by MePPLa.

The *instrumentation* used in the study consists of three forms: (i) a consent form, in which participants declared to accept participating in the study; (ii) a form to characterize the participants profile, aiming to obtain information about the participants' knowledge and experience in software measurement, SPC and pattern languages; and (iii) a questionnaire to capture the participants' perceptions about MePPLa.

The *procedure* adopted in the study consisted in sending to the participants a document containing information about MePPLa and the supporting tool. Based on the instructions provided in the document, participants used the tool and, then, answered the questionnaire (iii).

The *questionnaire* included questions related to MePPLa usefulness, adequacy of results obtained from the use of MePPLa, quality of the produced result, and productivity of the measurement planning activity. The questionnaire contained

[1] The tool is available in *http://dev.nemo.inf.ufes.br:8180/MPPL/login.faces.*

objective questions and a discursive one. For the objective questions, we asked the participant to provide a justification for the given answer. The discursive question aimed at the general evaluation and improvement of MePPLa.

The study *participants* were three professionals with experience in SPC implementation or evaluation in software organizations (CMMI and MR-MPS-SW appraisers or implementers) and one graduate student. Among the participants, two declared *medium* knowledge of software measurement and SPC, and two declared *high* knowledge. Concerning practical experience in measurement planning and SPC, two participants declared *high* experience, one declared *medium* and one participant declared *low* experience. As for the experience with pattern languages, two participants declared *high* experience, one declared *low* experience and one reported that the use of MePPLa it would be the first experience with pattern languages. Next, we present the results and some discussions related to each of the three indicators used in the study.

(i) Adequacy of the result obtained from the use of MePPLa

The use of MePPLa results in a set of measurement goals, information needs and measures (plus operational definition) related to processes to be submitted to SPC. As previously said, this set can be included in a measurement plan or can be turned into one by adding complimentary information not provided by the patterns. Three participants reported that the result is adequate and one participant considered it inadequate. The participant who considered the result inadequate justified that it was due to lack of detailed information in the operational definition of measures and absence of business goals.

Regarding the operational definition of measures, since the patterns defined in MePPLa are to be used by several organizations, some pieces of information cannot be predefined because they depend on the organization for which the measurement planning is performed. For these pieces of information, the pattern provides guidelines to help organizations to complete the operational definition. Even so, after the study, considering the participant feedback, we reviewed the operational definitions of measures and made changes to improve them.

As for the business goals, as previously explained, they must be established before using MePPLa. Based on the business goals, it is necessary to identify the related processes to be submitted to SPC and, henceforward, using MePPLa to apply patterns related to the processes.

(ii) MePPLa usefulness

Three participants considered MePPLa useful or very useful. However, although considering the patterns useful, one participant found MePPLa useless and justified their perception in the difficulty to change information in the operational definition of measures and to include new measures into the resulting measurement plan.

It is possible to notice that the justification given by the participant is directly related to limitations of the supporting tool and not to MePPLa itself. We took the participant feedback into account and improved the tool accordingly.

(iii) Benefits provided by MePPLa

All participants considered that MePPLa contributes to the reuse of measures and measurement goals, being aligned with “favoring reuse”, which is one of the expected benefits of using pattern languages. Three participants emphasized that they were properly guided in selecting measures that should be in the measurement plan.

Also, two of the three participants who had previous experiences elaborating measurement plans for SPC reported that MePPLa makes the measurement planning much more productive or more productive. One participant reported that MePPLa made the activity less productive, due to limitations to edit the operational definition of measures. As discussed earlier, this limitation refers to the supporting tool rather than to MePPLa and it was addressed after the study.

All participants considered MePPLa very easy or easy to use. Three of them highlighted that MePPLa contributes to the quality of the measurement plan, since the patterns provide well-defined measures and only particularities have to be addressed by the organizations. One participant believes that MePPLa contributes partially to the quality of measurement plan, because some aspects such as business goals are not included in the patterns. As mentioned before, organizational goals must be established before using MePPLa.

By analyzing the results and comments made by the participants it is possible to notice that most of the limitations pointed out refer to the supporting tool. In fact, the participant who considered MePPLa useless made it clear in a comment that the critics did not refer to MePPLa, but to the supporting tool. Thus, we believe that the tool limitations influenced MePPLa evaluation. Even so, considering the results as a whole, we can understand them as preliminary evidence that MePPLa is easy to use, it is useful, provides benefits in terms of quality and productivity, and is capable of producing adequate results.

5.1. Threats to Validity

Every study presents threats to the validity of its results. Threats should be addressed as much as possible and should be considered together with the results obtained in the study. Following the classification presented by Wöhlin *et al.* (2000), next we discuss the main threats to the study results.

Internal Validity: It is defined as the ability of a new study to repeat the behavior of the current study with the same participants and objects. The main threat to internal validity is communication and sharing of information among participants. To address this threat, we sent the document with instructions for evaluating MePPLa to the participant's personal email, so that s/he could individually use the tool and answer the questionnaire at the time s/he considered most appropriate. This can minimize the threat of communication, since participants are not physically close during the study and do not perform the study at the same time.

External Validity: This threat is related to the ability to repeat the same behavior with different groups from the one that participated in the study. In this context, we identified as a threat the short time participants had to perform the evaluation (less than a week). It is possible that different results are obtained with participants who have more time to perform the evaluation. Another threat refers to the fact that the participants did not use MePPLa in real projects. Evaluating MePPLa with participants using it in real projects may provide different results.

Construction Validity: Refers to the relationship between the study instruments and participants and the theory being tested. We identified as a threat the possibility of the participants give answers that do not reflect the reality due to personal expectations or to imagine that they were being evaluated. To minimize this threat, we informed the participants that the study does not represent any type of personal assessment and aims to evaluate MePPLa. Moreover, we ensured the participants anonymity. Another threat

concerns the document made available to the participants. The document should provide the necessary information for participants to evaluate MePPLa (i.e., information about MePPLa and the tool). We sought to produce a simple and small documentation that did not require much time from the participants. However, in doing so, important information about MePPLa and its use, which could impact on the evaluation results, may not have been included or not being sufficiently clear (e.g., it seems that the need of establishing the business goals before using MePPLa was not clear for one participant). We also considered a threat the use of the supporting tool to evaluate MePPLa. By using the tool, participants evaluated not only MePPLa, but also aspects related to the tool itself. Evaluating MePPLa by using only its specification document may produce different results.

Conclusion Validity: It measures the relationship between the treatments and the results and affects the ability of the study to generate conclusions. In this context the main threats are: (i) small number of participants; (ii) short period for evaluation; (iii) supporting tool limitations and (iii) MePPLa was not used in real projects. These threats limit results generalization, thus they should be considered preliminary results.

6. Final Considerations

The importance of SPC for the software industry has increased in recent years due to the interest of organizations in achieving high maturity [Fernandez-Corrales *et al*. 2013]. The use of SPC allows getting know the processes behavior and predicting their performance.

Process stability, measurement capacity and data reliability are critical issues for a successful implementation of SPC. In other words, if the process is consistently performed, if the correct measures are selected and a reliable data collection mechanism is established, it is possible to obtain the benefits provided by SPC [Abubakar and Jawawi 2013]. Selecting measures suitable for SPC has been pointed as one of the difficulties when implementing SPC in software organizations. The literature cites several measures for it, but information is disperse and non-structured. Moreover, usually there are no guidelines on which measures should be selected in a given context.

Reusing measures applied in SPC initiatives can help select measures appropriate for this purpose. Extracting measurement patterns from previous SPC initiatives can contribute to reuse and assist organizations in measurement planning for SPC. Considering that, we proposed MePPLa (Measurement Planning Pattern Language), a pattern language to help measurement planning aiming at SPC.

In the context of this work, we investigated the state of the art on measures used in SPC initiatives by carrying out a systematic mapping [Brito and Barcellos 2016]. We identified some gaps, such as: (i) lack of approaches to select measures suitable for SPC; (ii) absence of operational definition of measures; and (iii) lack of concern with related measures. These gaps are addressed in MePPLa, since: (i) MePPLa provides a flow that guides the user in selecting patterns containing measures; (ii) MePPLa provides operational definitions for the measures included in the patterns; (iii) MePPLa has a structural model in which information about the structural relationships between patterns reveals correlated measures and goals that impact on others.

Comparing MePPLa with the pattern languages proposed in [Andrade *et al.* 2010; Braga *et al*. 2012] and presented in Section 2, some important differences can be highlighted: *(i)* the pattern languages do not address SPC; *(ii)* the patterns of these pattern languages provide guidelines for project planning, while MePPLa provides

patterns based on the GQM format and using them helps measurement planning aiming at SPC; *(iii)* the patterns of the pattern languages do not provide operational definition of measures; *(iv)* the pattern languages do not use a cognitively rich visual notation; and (*v*) these pattern languages do not separate the structural and behavioral perspectives.

It is worth pointing out that the version of MePPLa addressed in this paper is a first version, which considers three processes and has a limited number of patterns. Pattern languages may be always evolving, for example, having new patterns added and new relationships identified. Thus, as a pattern language, MePPLa can be extended to include new patterns and handle new processes.

Among the limitations of this work, we can highlight MePPLa evaluation, which involved a small number of participants, in a very short period of time and outside an organizational context. In addition, limitations of the supporting tool used in the evaluation influenced the results.

As future work, we plan to conduct new studies for better evaluating MePPLa, to evolve MePPLa to treat other processes and include other patterns, and to improve the supporting tool to favor the use of MePPLa by organizations.

Acknowledgements

This research is funded by the Brazilian Research Funding Agency CNPq (Process 461777/2014-2) and FAPES (Process 69382549/2014), FAPERJ (projects E-26/210.643/2016, E- 211.174/2016) and UNIRIO (grant PQ-UNIRIO 01/2016 e 01/2017).

References

Abubakar, A. M. and Jawawi, D. N. A. (2013). A Study on Code Peer Review Process Monitoring using Statistical Process Control. Software Engineering Postgraduates Workshop (SEPoW), p. 136–141.

Andrade, T. de C., Freitas, F. G. de, Andrade, R. M. de C., Souza, J. T. de. (2010). Software Estimation Patterns applied at a Small Enterprise. Proceedings of the 8th Latin American Conference on Pattern Languages of Programs. Article No. 15. Salvador, Bahia, Brazil (in Portuguese only).

Barcellos, M. P., Falbo, R. A., Rocha, A. R. (2010). Establishing a well-founded conceptualization about software measurement in high maturity levels. In Proc. of the 7th International Conference on the Quality of Information and Communications Technology, p. 467–472.

Barcellos, M. P., Falbo, R. A., Rocha, A. R. (2013). A strategy for preparing software organizations for statistical process control. Journal of the Brazilian Computer Society, v. 19, n. 4, p. 445–473.

Basili, V. R., Rombach, H. D., Caldiera. (1994). G. Goal Question Metric Paradigm, Encyclopedia of Software Engineering, 2 Volume Set, John Wiley & Sons, Inc.

Braga, M.R.R., Bezerra, C.I.M., Monteiro, J.M., Andrade, R. (2012). A pattern language for agile software estimation. Proc. of the 9th Latin-American Conference on Pattern Languages of Programming. Natal, RN, Brazil.

Brito, D. F., Barcellos, M. P. (2016). Measures Suitable for SPC: A Systematic Mapping. Proc. Of the XV Brazilian Symposium on Software Quality, Maceió – AL, Brazil.

Brito, D. F. (2017) Linguagem de Padrões para apoiar o Planejamento de Medição para o Controle Estatístico de Processos de Software. Dissertação de Mestrado, Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, ES, Brasil *(in Portuguese only)*.

Buschmann, F., Henney, K., Schimdt, D. (2007), Pattern-oriented Software Architecture: On Patterns and Pattern Language. John Wiley & Sons Ltd.

Deutsch, P. (2004). Models and Patterns. In: J. Greenfield; K. Short; S. Cook; S. Kent (Orgs.); Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Indianapolis: John Wiley & Sons.

Falbo, R. A., Barcellos, M. P., Nardi, J. C., Guizzardi, G. (2013). Organizing ontology design patterns as ontology pattern language. Proc. of the 10th Extended Semantic Web Conference - ESWC. Montpellier, France.

Fernandez-Corrales, C., Jenkins, M., Villegas, J. (2013). Application of Statistical Process Control to Software Defect Metrics: An Industry Experience Report. ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, p. 323–331.

Florac, W. A. and Carleton, A. D. (1999). Measuring the Software Process: Statistical Process Control for Software Process Improvement. Pearson Education.

Hevner, A. R. A. (2007). Three Cycle View of Design Science Research. Scand. J. Inf. Syst. 19, 87–92.

Hevner, A.R., March, S.T., Park, J., Ram, S. (2004). Design Science in Information Systems Research. 28, 75–105.

ISO/IEC (2007). ISO/IEC15939—Systems and Software Engineering—Measurement Process.

Montoni M, Rocha AR, Weber KC (2009) MPS.BR: A Successful Program for Software Process Improvement in Brazil. Software Process Improvement and Practice 14:289-300.

Quirino, G. K. S. (2016). Uma Notação Visual para Representação de Linguagens de Padrões Ontológicos. Dissertação de Mestrado, Programa de Pós-Graduação em Informática, Universidade Federal do Espírito Santo, ES, Brasil (in Portuguese only).

Rocha, A. R. C. da, Souza, G. dos S. and Barcellos, M. P. (2012). Medição de Software e Controle Estatístico de Processos, Ministério da Ciência, Tecnologia e Inovação - SEPIN - PBQP Software, Brasília – Brasil, ISSN 1679-1878 (in Portuguese only).

Schots, N., Rocha, A. and Santos, G. A (2014) Body of Knowledge for Executing Performance Analysis of Software Processes. SEKE, pp 560-565.

SEI (2010). CMMI for Development, Version 1.3. Carnegie Mellon University.

Solingen, R., Berghout, E. (1999). The Goal/Question/Metric Method: a practical guide for quality improvement of software development. New York: McGraw-Hill Publishing Company.

Wheeler, D. J., Chambers, D. S. Understanding Statistical Process Control. 2nd ed. Knoxville - SPC Press, 1992.

Wöhlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A. (2000). Experimentation in Software Engineering: An Introduction. The Kluwer International Series in Software Engineering, Norwell, USA, Kluwer Academic Publishers.