


Pattern Language as Support to Software Measurement Planning for Statistical Process Control


Daisy Ferreira Brito

(Ontology and Conceptual Modeling Research Group (NEMO)
Computer Science Department, Federal University of Espírito Santo, Vitória – ES, Brazil
 <https://orcid.org/0000-0002-1089-4883>, dfbrito@inf.ufes.br)

Monalessa P. Barcellos

(Ontology and Conceptual Modeling Research Group (NEMO)
Computer Science Department, Federal University of Espírito Santo, Vitória – ES, Brazil
 <https://orcid.org/0000-0002-6225-9478>, monalessa@inf.ufes.br)

Gleison Santos

(Graduate Program on Information Systems
Federal University of the State of Rio de Janeiro, Rio de Janeiro – RJ, Brazil
 <https://orcid.org/0000-0003-0279-0440>, gleison.santos@uniriotec.br)

Abstract: The growing interest of organizations in improving their software processes has led them to aim at achieving high maturity, where statistical process control (SPC) is required. One of the challenges involved in performing SPC is selecting measures suitable for it. Measures used in SPC can be found in the literature and can be reused by organizations, but the information is dispersed, not favoring reuse. From measures suggested in the literature or used in practical experiences, it is possible to identify patterns that can be used to support organizations in measurement planning. Patterns can be organized as pattern languages, which favor reuse and contribute towards increasing productivity. In this work, from the results of a systematic mapping and a survey, we identified measurement planning patterns in the Goal-Question-Metric format and organized them in a Measurement Planning Pattern Language (MePPLa). MePPLa was created by following a Systematic Approach for creating Measurement Planning Pattern Languages (SAMPPLa), also defined in this work. This paper presents SAMPPLa, MePPLa and the main results of a study carried out to evaluate MePPLa. The results showed that using MePPLa is viable and useful to aid in software measurement planning. Mainly, MePPLa contributes to increasing productivity when creating a measurement plan and the quality of the resulting measurement plan.

Keywords: Software Measurement, Statistical Process Control, Measure, Pattern Language, Measurement Planning

Categories: D.2.8

DOI: 10.3897/jucs.68237

1 Introduction

The need to develop more robust and complex software to meet organizations' and people's needs has contributed to increasing the interest of software organizations in

software process improvement (SPI). There are several standards and maturity models that support SPI implementation. Some of them, such as the Capability Maturity Model Integration for Development (CMMI-DEV) [CMMI Institute 2018] and the Reference Model for Brazilian Software Process Improvement (MR-MPS-SW) [Santos et al. 2015], propose an SPI implementation in levels. At the highest levels (e.g., CMMI-DEV levels 4 and 5 and MR-MPS-SW levels B and A), a more mature way of measurement is required and involves the use of statistical process control (SPC).

SPC employs statistical methods to monitor, control, and increase the knowledge of a process, enabling improving its behavior [López et al. 2018]. One of the difficulties software organizations face in implementing SPC is to define measures suitable for it [Barcellos et al., 2013]. In the literature, several works present measures that can be used in SPC initiatives, and they can be reused by organizations intending to carry out SPC. However, selecting which measures are helpful in a specific context is not trivial because the information is dispersed and non-structured, making access difficult, effort-demanding, and inefficient sometimes. Software measurement programs often fail due to bad decisions on what should be measured [Tahir et al. 2018a].

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 analyze the behavior of specific processes. A pattern encapsulates knowledge and can be understood as a successful solution to a problem [Greenfield et al. 2004]. In this work, we refer to *measurement planning pattern* as the encapsulated knowledge representing a measure related to a process and that can be used to support monitoring a certain goal. Hence, in this work, a measurement planning pattern presents a solution to the problem of selecting suitable measures to be included in a measurement plan according to the goals to be achieved/monitored.

Patterns can be organized into pattern languages (PLs), which represent patterns and their relationships and also define a process that guides pattern selection and use. In the 70s, Christopher Alexander [Alexander et al. 1977] presented a PL for towns, buildings, and construction. Inspired by Alexander's work, software practitioners and researchers have organized solutions related to software as patterns since the 1990s. The PLs structure primarily aims at making knowledge easily applicable [Jörg and Frederik 2007]. Therefore, the use of PLs favors reuse and, consequently, contributes towards improving productivity. In addition, since a PL provides a mechanism for selecting patterns (e.g., a flow that guides the user in pattern selection), even users without much knowledge of the problem domain can be directed towards the solution [Falbo et al. 2013].

It is important to point out that although PLs are often associated with programming and design activities, they have been adopted to support other activities in Software Engineering. Moreover, a more flexible meaning has been assigned to the term pattern, making it broader than the original Christopher Alexander's architectural patterns. Quirino et al. (2018) conducted a study that found 64 PLs supporting different Software Engineering activities and including various patterns, with different representations, abstraction levels, and purposes, such as design patterns, process patterns, architectural patterns, analysis patterns, ontology patterns, instructional patterns, interaction patterns, among others.

Considering the benefits provided by PLs, we argue that they can be useful in the software measurement planning context. For example, they can assist organizations in

elaborating measurement plans through the reuse of measurement planning patterns related to processes and containing goals, information needs, and measures. In other words, organizations planning their measurement programs can reuse measures already used in the past, considering their goals and software processes to be submitted to SPC. For instance, an organization with the goal of improving test coverage and that wants to submit the Testing process to SPC aiming at high maturity could reuse the solution adopted by other organizations that submitted the Testing process to SPC to achieve the same goal (e.g., a measurement planning pattern containing the measure test coverage rate).

Therefore, inspired by the PL notion, we developed a Measurement Planning Pattern Language (MePPLa) that helps organizations elaborate measurement plans suitable for SPC. In MePPLa, each pattern is a solution comprising a measurement goal, a process to be submitted to SPC, and measures to analyze the process behavior and monitor the referred goal. Thus, each pattern is a solution to analyze the behavior of a specific process considering a particular goal. The patterns were identified based on the results of a survey that investigated measures used in SPC in practice and a systematic mapping study that investigated measures used in SPC initiatives reported in the literature [Brito et al. 2018]. These studies provided a set of measures (related to software processes and goals) that have been used by organizations implementing SPC. By analyzing the measures, we identified some that have been used recurrently. Thus, we defined measurement planning patterns based on them and organized the patterns in a PL.

MePPLa is the main contribution of this paper. Currently, it comprises 28 patterns, of which 12 are related to the Project Management process, six are related to the Coding process, and ten are related to the Testing process. It was created by following a Systematic Approach for creating Measurement Planning Pattern Languages (SAMPPLa), also defined in this work. MePPLa can help organizations create measurement plans aiming at SPC, and it can be continuously evolved. SAMPPLa, in turn, can be used to evolve MePPLa and define other measurement planning PLs. This paper presents MePPLa and SAMPPLa. By describing SAMPPLa, we show how we developed MePPLa and how it can be evolved to address other processes and provide new patterns to be more comprehensive and satisfy the needs of different organizations. Considering that each organization has particularities and different realities, MePPLa can constantly be evolving. SAMPPLa will be helpful in this matter.

In a previous work [Brito et al. 2017] we introduced MePPLa by showing its patterns related to the Testing process and presenting the results of MePPLa preliminary evaluation. In this paper, we extend the initial contribution by (i) providing more background on software measurement and SPC; (ii) adding a new section about the studies conducted to investigate measures used in SPC; (iii) introducing SAMPPLa, the approach we defined and followed to create MePPLa; (iv) extending MePPLa presentation by discussing what was done in each activity of SAMPPLa to develop MePPLa and presenting fragments of MePPLa related to the Project Management and Coding processes; (v) presenting results of a new study performed to evaluate MePPLa; (vi) making available a new version of MePPLa specification and MePPLa Tool, the computational tool that supports MePPLa use; and (vii) improving the discussion of related work.

This paper is organized as follows: Section 2 provides the background for the paper

talking about software measurement, SPC, and pattern languages; Section 3 concerns the adopted research method; Section 4 regards the studies carried out to investigate measures used in SPC, whose results were used as a basis to develop MePPLa; Section 5 presents SAMPPLa; Section 6 presents MePPLa and MePPLa Tool, a computational tool developed to support the use of MePPLa; Section 7 addresses the experimental study carried out to evaluate MePPLa; Section 8 discusses related work; and Section 9 presents our final considerations.

2 Background

2.1 Software Measurement and Statistical Process Control

Software measurement is essential to characterize, evaluate, predict and improve software products, processes, and resources [Tahir et al. 2018b]. It is the continuous process of defining, collecting, and analyzing data about software processes and products to understand and control them and supply meaningful information for their improvement [Solingen and Berghout 1999]. Research on software measurement continues to be a hot topic today. Although the benefits of using software measurement are well known, problems in their practice are still ongoing [Tekin et al. 2020].

To perform software measurement, an organization must initially 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 must also 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 for decision-making, supporting the identification of appropriate actions. Finally, the measurement process and its products should be evaluated to identify potential improvements [McGarry et al. 2002 and ISO/IEC/IEEE 2017].

Depending on the organization's maturity level (or on the maturity level of the organization's specific processes), software measurement is performed in different ways. There are some standards and models that help evaluate the organization's maturity level. For example, CMMI [CMMI Institute 2018] is a model that describes guidelines for the definition and implementation of software processes aiming at software process improvement. In CMMI, practices from several areas, such as Project Management, Product Engineering, and Process Management, are present. Organizations that implement processes adherent to CMMI practices can show the market that they can deliver quality products and services within a predictable timeframe and cost.

CMMI establishes six maturity levels. At level 0 (Incomplete), the work is done randomly, i. e., at this level, the work may or may not get completed. At level 1 (Initial), the work done is unpredictable and reactive. At this level, although the work gets completed, it is often delayed and over budget. At level 2 (Managed), works are managed on the project level. Projects are planned, performed, measured, and controlled. Level 3 (Defined) is considered more proactive and less reactive than the previous levels. Organization-wide standard processes are defined, providing guidance

across projects, programs, and portfolios at this level. Finally, maturity levels 4 (Quantitatively Managed) and 5 (Optimizing) are considered high maturity levels. At level 4, the organization is data-driven with quantitative performance improvement objectives that are predictable and align to meet the needs of internal and external stakeholders. At level 5, the organization is focused on continuous improvement and is built to pivot and respond to opportunity and change. At this level, the organization's stability provides a platform for agility and innovation [CMMI Institute 2018].

At initial maturity levels (such as CMMI levels 2 and 3), measurement consists mainly of 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 understand the processes' behavior, determine their performance in previous executions, and predict their performance in current and future projects, verifying if they are capable of achieving the established goals [Barcellos et al. 2013].

SPC uses a set of statistical techniques to determine if a process is under control from a statistical point of view. A process is under control if its behavior is stable, i.e., if its variations are within the expected limits, which are calculated based on historical data. The behavior of a process is described by data collected for measures that characterize the process [Florac and Carleton 1999].

A process under control has repeatable behavior. Consequently, it is possible to predict its performance in future executions and thus prepare achievable plans and continuously improve the process. On the other hand, a process that varies beyond the expected limits is called an unstable process. The causes of these variations (the so-called special causes) must be investigated and addressed by improvement actions aiming at stabilizing the process. Once the processes are stable, their levels of variation can be established and sustained, making it possible to predict their results. Thus, it is also possible to identify the processes capable of achieving the established goals and the processes failing to achieve them. In this case, actions that change the process to make it capable must be carried out. Stabilizing critical processes is a practice of high maturity organizations or organizations aiming to achieve the highest maturity levels [Florac et al. 2000 and Caivano 2005].

In the literature, there are several works addressing software measurement. One of the best known is GQM (Goal-Question-Metric) [Basili et al. 1994]. GQM represents a systematic approach for tailoring and integrating goals to software processes, products, and quality perspectives of interest, based upon the needs of projects and the organization. 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, organizations can derive information needs from their goals and define measures to meet them.

Some ISO standards deal with the software measurement process or software measures. For example, ISO/IEC/IEEE 15939 (2017) is devoted to the measurement process and defines the activities and tasks necessary to implement that process. It also describes a measurement information model that links information needs to measurable entities and attributes required to meet them. Other ISO standards specify quality characteristics that can be used to define software measures. For example, ISO/IEC 9126 (2001) specifies characteristics related to software internal and external quality and quality in use. In turn, ISO/IEC 25020 (2007) provides information about measuring characteristics and sub-characteristics of the quality model defined in

ISO/IEC 25010 (2011).

Another relevant work related to measurement is the Practical Software and Systems Measurement (PSM) [McGarry et al. 2002]. PSM was created to help develop, operate, and continuously improve measurement programs. Its process provides experience-based guidance on defining and implementing a measurement process for a software project. There is also the PSM CID [PSM/NDIA/INCOSE 2021], which provides recommendations focusing on the measurement of continuous iterative development (CID), including common information needs and measures that are effective for evaluating CID approaches. The information needs address team, product, and enterprise perspectives to provide insight and drive decision-making.

Some works in the literature focus on SPC for software processes. For instance, Florac et al. (1999) discuss how characteristics of software products and processes can be measured and analyzed using SPC techniques so that the behavior of the processes that produce the software products can be managed, predicted, controlled, and improved to achieve business and technical goals. Another well-known approach for process improvement based on process performance measurement and SPC is Six Sigma. It focuses on improving customer satisfaction through defect prevention and elimination, consequently improving organizational processes. It is composed of a set of tools involving the measurement of process performance and frameworks for improvement. Its best-known frameworks are DMAIC (Define, Measure, Analyze, Improve, Control) and DFSS (Design for Six Sigma). DMAIC is used to improve existing processes and products, and DFSS is used to design new products and processes [Siviy et al. 2005].

Tarhan and Demirörs (2006) present an approach to assess the suitability of software processes and measures for starting SPC implementation using control charts. The approach includes guidance to identify rational samples of a process as well as to select process measures. Razmochaeva et al. (2019) discuss the main aspects of statistical management of quality control processes. A comparative analysis of the existing software for statistical management systems is carried out. The advantages and disadvantages are noted, and the applicability of each of the systems in the problem of process automation is formed. At last, Maisikeli (2020) utilized a statistical process control approach to monitor, track and evaluate whether a changed process is stable or not and discover situations where the process is in or out of kilter.

2.2 From Patterns to Pattern Languages

In the literature, there are several pattern definitions. In a very generic way, Coplein (1998) defines a pattern as “the thing and the instructions for making the thing”. According to Greenfield et al. (2004), patterns are vehicles for encapsulating knowledge. They make it possible to capture what must be done to solve a given problem. Ferdinandi (2002), in turn, defines a pattern as any reusable template based on experience that can be used to guide the creation of a solution to a problem or need in a specific context. For Kamthan (2007), a pattern is a reusable entity representing knowledge and experience aggregated by an expert in solving a recurring problem in a domain. Tešanovic (2005) refers to patterns as effective means of communication that help bring order into the chaos by representing best practices, proven solutions, and lessons learned that aid in software engineering activities. Although different, in general, the definitions converge to define a pattern as a solution to a recurrent problem.

Hence, before using a pattern, it is necessary to recognize an opportunity to apply it.

In this work, we are aligned to a broader view of what a pattern is. We consider a pattern the way that something is often done or repeated to solve a given problem or need. For example, measures frequently and successfully used to monitor a particular goal can be viewed as a pattern to solve the problem of monitoring that goal. Notice that the main principle behind a pattern remains: reuse a successful solution and not reinvent the wheel.

Many patterns found in the literature are related to others. Still, most fail to explain how patterns can be combined to form solutions to larger problems than those treated by each pattern individually [Buschmann et al. 2007]. Pattern Languages can be used to solve this issue. In Software Engineering, a pattern language (PL) is a network of interrelated patterns that defines a process for systematically solving problems related to software engineering [Greenfield et al. 2004]. A PL must indicate problems that may arise in the domain of interest and inform possible solutions to them by suggesting one or more patterns to solve each problem [Falbo et al. 2013].

Visual notations can be used to represent PLs graphically. The purpose of adopting graphical notations is to overview the patterns and their relationships, contributing towards a holistic understanding of the PL and assisting in pattern selection [Quirino et al. 2017]. Quirino et al. (2017) proposed a cognitively rich visual notation named OPL-ML (Ontology Pattern Language Modeling Language) to represent PLs in the Ontology Engineering domain. Although it was proposed to represent ontology PLs, OPL-ML is based on Software Engineering works. The authors argue that it can also be applied to represent PLs in other domains.

There are several works in the literature proposing PLs in the Software Engineering domain. However, only a few have been proposed to support software measurement, such as [Nikelshpur 2011] and [Braga et al. 2012]. Nikelshpur (2011) presents some best practices of software estimation in the form of nine patterns. Although the author claims that the patterns compose a pattern language, the sense of connection among the patterns is not clear. Braga et al. (2012), in turn, proposes a PL for estimates in agile projects. The PL consists of eight process patterns that can help agile teams to perform estimates for agile software projects.

3 Research Method

We adopted a research method that followed the Design Science Research (DSR) paradigm. DSR concerns extending human and organizational capabilities by creating new artifacts in context [Hevner et al. 2004 and Hevner 2007]. The artifacts are designed to interact and improve something in that context [Wieringa, 2014]. DSR is an interactive process that considers three cycles of related activities: Relevance, Design, and Rigor [Hevner 2007].

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 we addressed is the difficulty software organizations face when planning measurement for SPC, especially when selecting the measures to be used. This problem is reported in the literature (e.g., [Grossi et al. 2014, Kitchenham et al. 2007, Tarhan and Demirors 2006, Tarhan and Demirors 2008, Tarhan and Demirors 2012, Schots et

al. 2014]) and has also been perceived by the authors in practical settings (particularly by one of the authors, who has worked implementing and evaluating maturity models in software organizations for more than 15 years. Aiming to understand the state-of-the-art on measures used in SPC initiatives, we investigated the literature through a systematic mapping study [Brito et al. 2018]. We also surveyed Brazilian practitioners to obtain information from the practice, asking them to inform the measures they have employed in SPC [Brito et al. 2018]. From the studies' results, we perceived: (i) prevalence of defect-related measures; (ii) lack of concern with relations between measures; (iii) lack of approaches to support measure selection; and (iv) lack of operational definitions for the measures.

Considering the problem, the gaps identified from the studies, and the benefits of using PLs reported in the literature, we noticed that using a measurement planning PL could help organizations develop measurement plans for SPC. Thus, we decided to propose a PL with this purpose. We used the results from the studies mentioned above as a source for patterns identification. We established as main requirements that the PL should (i) provide patterns to aid in elaborating measurement plans for SPC, (ii) be able to guide users on selecting patterns to be included in a measurement plan, (iii) represent the relationships between measures, and (iv) be graphically represented. As criteria for results evaluation, the viability of using the PL and its usefulness should be considered.

The Design Cycle refers to the development and evaluation of artifacts or theories to solve the identified problem [Hevner 2007]. To reach our objective, we developed MePPLa, a PL to support measurement planning aiming at SPC. The patterns of MePPLa were identified from the systematic mapping and survey [Brito et al. 2018] results. To support the use of MePPLa, we developed a tool. To evaluate MePPLa, we carried out two experimental studies where participants used MePPLa and provided their perceptions about it. To create MePPLa, we defined SAMPPLa, an approach to guide the creation and evolution of PLs to measurement planning aiming at SPC.

Finally, the Rigor Cycle refers to knowledge use and generation. Rigor is achieved through the adequate application of existing foundations and methodologies. 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]. The main theoretical foundations are secondary studies (systematic mappings in particular), software measurement, SPC, PLs, experimental study, and survey. The main contributions to the knowledge base are: (i) MePPLa, which can support organizations in measurement planning for SPC and can be evolved to incorporate new patterns and processes; (ii) SAMPPLa, which can be used to evolve MePPLa or create new measurement planning PLs; (iii) the systematic mapping [Brito et al. 2018], which consolidates information about measures used in SPC initiatives, providing a panorama of the research topic and indicating possible future research; (iv) the survey conducted with practitioners [Brito et al. 2018], providing information about measures that have been used in SPC initiatives in Brazilian organizations; and (v) MePPLa Tool, the tool developed to support the use of MePPLa.

Figure 1 summarizes the key information related to the DSR cycles in this research.

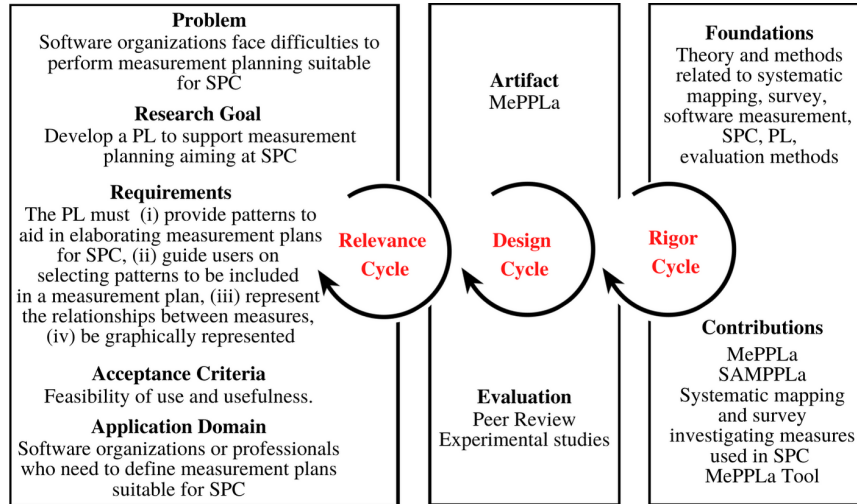


Figure 1: Overview of the DSR cycles in this work (based on [Hevner 2007])

As stated before, in this paper, we focus on presenting MePPLa and SAMPPLa. For that, in the next section, we provide a general view of the studies carried out to identify measures used by software organizations when implementing SPC. As we explained before, we used the studies' results to develop MePPLa.

4 Investigation of Measures applied to SPC

Considering that we were interested in identifying measures used successfully in SPC implementation and represent them as patterns that can be reused by organizations aiming to implement SPC, we investigated measures that have been used in SPC initiatives for software processes. The investigation consisted of a systematic mapping of the literature and a survey with Brazilian practitioners [Brito et al. 2018]. According to Kitchenham and Charters (2007), a systematic mapping study is a type of literature review designed to provide an overview of a research area or research topic and helps identify gaps that can be addressed in future research. It is performed by following a rigorous research protocol. We chose this approach to investigate the literature aiming to reduce bias, increase coverage and ensure the repeatability of the study.

To investigate the literature, we applied the following search string to six digital libraries (IEEE Xplore, ACM Digital Library, Springer Link, Engineering Village, Web of Science, Science Direct, and Scopus): ("statistical process control" OR "SPC" OR "quantitative management") AND ("measurement" OR "measure" OR "metric" OR "indicator") AND ("software").

The selection of the publications was performed in five steps. In the first step, we obtained 558 publications after running the search string in the digital libraries. In the second step, we removed duplications. In the third step, we analyzed the title, abstract, and keywords of the resulting 318 publications considering the following inclusion (IC) and exclusion (EC) criteria: (IC1) the publication addresses SPC in software processes

and measures used in SPC; (EC1) the publication does not have an abstract (EC2) or is just an abstract; and (EC3) the publication is a secondary study, a tertiary study, a summary, or an editorial. Then, we read the full text of resulting 84 publications. We considered the following additional criteria: (IC2) the publication presents measures for SPC in software processes or presents cases involving SPC in which the measures used are cited; (EC4) the publication is a copy or an older version of an already selected publication; (EC5) the publication is not written in English; (EC6) the publication's full text is not available. After this step, we reached 39 studies. In the last step, to increase coverage, we performed snowballing, i.e., we analyzed the references of the selected publications looking for the ones able to provide evidence for the study. In this step, we added 11 publications, reaching a total of 50 publications.

After selecting the publications, we extracted and recorded data about the measures cited in the publications, the processes to which they are related, and the goals monitored by them. As a result, we identified 12 processes, 45 goals, and 82 measures.

After the mapping study, to complement the results, we applied a questionnaire to professionals from Brazilian organizations, aiming to identify processes, goals, and measures they have used in SPC. The participants were professionals with experience in implementing or appraising SPC practices in Brazilian software organizations. Five measures, five goals, and seven related processes were identified based on the participants' answers.

After excluding the duplicated elements between the two studies, we have, as a result, 84 measures, 15 processes, and 47 goals used in SPC. Table 1 presents some of the found goals, while Table 2 presents a fragment of the resulting measures and related processes and goals. The formulas presented in Table 2 were extracted from the publications. Thus, they reflect the way the measure is calculated according to the source publication. Measures preceded by * were used in initiatives involving standards/maturity models. Measures preceded by ° were used in initiatives not involving standards/maturity models. In Table 2, when a measure is related to a process/goal, it means that at least one publication cited the measure related to the process/goal. In Table 1, we kept the goals id established during the study. The full results of the studies can be found in [Brito et al. 2018].

ID	Goal	ID	Sub-goal
G01	Assess and monitor the maintenance process	-	-
G03	Estimate and control defects, effort, and schedule of the testing process	-	-
G04	Evaluate process quality effectiveness	G04.3	Evaluate peer review effectiveness
		G04.4	Manage the effectiveness of defect removal activities
G05	Improve product quality	G05.1	Improve defect detection to reduce the number of delivered defects
		G05.2	Improve software process effectiveness
		G05.4	Increase customer satisfaction (by managing defects)
		G05.8	Reduce requirements volatility

G08	Monitor process efficiency	G08.1	Evaluate coding efficiency
G09	Reduce operational costs	G09.1	Improve productivity
		G09.2	Minimize rework
		G09.3	Monitor project cost and schedule
		G09.5	Improve estimation and planning
G10	Understand software processes performance	G10.3	Understand project management process performance
G15	Win the market competition	G05	Improve product quality

Table 1: Some goals (and sub-goals) found in the studies

Measures	Processes	Goals
*Cost performance index <i>(budget cost for performed work / actual cost for performed work)</i>	Project Management	G09.3
*Duration estimation accuracy <i>(actual duration / estimated duration)</i>	Project Management	G09.1; G09.5; G10.3; G15
*Effort estimation accuracy <i>(actual effort / estimated effort)</i>	Project Management	G09.1; G09.5; G10.3; G15
*Productivity <i>(effort/product size)</i>	Maintenance, Requirements Analysis, Design, Coding, Testing	G01; G03; G09.1; G05.2; G08; G10
*Delivered defect density <i>(defects detected after product release/product size)</i>	Testing	G05; G05.1
*Defect removal effectiveness (by Requirements Development, Design, Coding, and Testing) <i>(number of removed defects in requirements (or design, coding, and testing) / number of detected defects)</i>	Coding, Design, Requirements Development, Testing	G04.4
*Percentage of high severity defects identified in testing <i>(number of high severity defects identified in testing/total number of detected defects *100)</i>	Review, Testing	G09.1; G05.4; G09
*Rework percentage <i>(rework effort/total effort *100)</i>	Requirements Analysis, Design, Coding, and Testing	G09.2; G10
*Requirements change rate <i>(number of changed requirements /total number of requirements)</i>	Requirements Management	G05.8
*Review effectiveness <i>(number of defects detected in reviews/total number of defects)</i>	Review	G04.3

Table 2: Some measures, processes, and goals found in the studies

5 SAMPPLa: Systematic Approach for Creating Measurement Planning Pattern Languages

SAMPPLa consists of a process that guides the creation or evolution of PLs to assist in measurement planning for SPC. It is composed of two activities, *Develop Source for*

Extracting Patterns and Develop Pattern Language. We defined SAMPPLa as a systematic process to be followed so that we could develop MePPLa. It can also be used to evolve MePPLa and to create other pattern languages that aim to aid measurement planning. Although SAMPPLa provides a systematic process to be followed, it is worth pointing out that when performing some of SAMPPLa steps, a lot of tacit knowledge and judgment may be necessary. Next, we provide an overview of SAMPPLa activities. In Section 6, we discuss and exemplify the execution of each of them, showing results produced when developing MePPLa.

5.1 Develop Source for Extracting Patterns

The first step in creating a PL is to obtain a source from which the patterns that will compose the PL can be extracted. In SAMPPLa, a pattern comprises a process to be submitted to SPC, a measurement goal related to the process, and measures to analyze the process behavior and monitor the goal. This activity is thus responsible for obtaining a set of processes, goals, and measures from which measurement planning patterns will be extracted. Figure 2 details this activity, which is broken down into a further six. In Figures 2 and 3, we adopt the UML activity diagram notation. Thus, the black circle represents the start point. Yellow rounded rectangles represent activities. An icon in the bottom right corner means that the activity is complex (i.e., it has sub-activities). White rectangles refer to artifacts. Arrows connecting artifacts to activities indicate which artifacts are used as input or produced as an output of each activity. Arrows connecting activities represent workflows, denoting the order in which the activities must be performed. Finally, a black circle inside a white circle limited by a black line represents the endpoint.

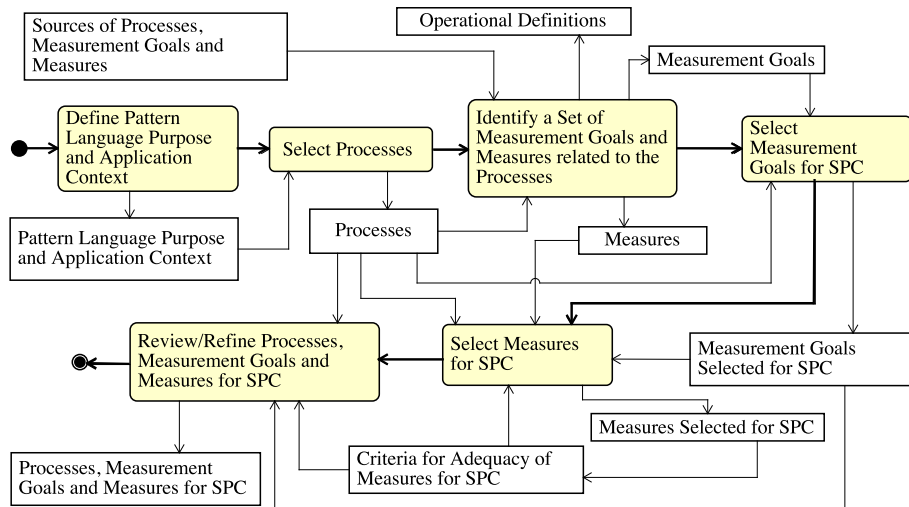


Figure 2: Detailing of Develop Source for Extracting Patterns

The source development starts with the **Define Pattern Language Purpose and Application Context** activity, which defines what the PL will be created for (e.g., to support measurement planning towards SPC to meet CMMI level 4 and 5 requirements)

and where it will be applied (e.g., CMMI level 3 organizations that wish to implement SPC practices). This information delimits the scope of the pattern language, and the next activities will be performed based on it.

Next, one must **Select Processes** that will be addressed in the PL. Thus, in this activity, one can select any software process that can be submitted to SPC, such as Requirements Development, Inspection, Testing, etc. Two basic guidelines are useful. The first is based on the criticality of the process [SOFTEX 2021, CMMI Institute 2018]. That is, selected processes must be relevant to achieve organizations' goals (e.g., Testing is a critical process because it directly affects the product quality). The second concerns the process size and frequency of execution [Barcellos et al. 2013, Florac and Carleton 1999]. Smaller processes, which are performed several times during a project and whose data collection is more frequent, favor reaching a meaningful data set and support decision-making along with their execution in the projects. It is important to note that this decision is directly related to the PL purpose and application context. For example, if the PL aims to aid in the statistical control of processes that deal with defects, then defect-related processes such as Testing and Verification should be selected.

Once the processes are selected, it is necessary to **Identify a Set of Measurement Goals and Measures related to the Processes**, which consists of gathering goals and measures related to the processes addressed in the PL. The goals and measures can be obtained from measurement repositories, literature, practical experiences, etc. containing goals and measures used in SPC. The operational definition of the measures should also be captured if they are available. To establish the set of measurement goals and measures resulting from this activity, one should: (i) extract the elements (measurement goals and measures) as they are recorded in the input sources; (ii) look for equivalences between the elements; (iii) represent equivalent elements in a unique way; and (iv) represent all relationships between the identified elements.

Next, the set of goals and measures must be analyzed to **Select Measurement Goals for SPC** and **Select Measures for SPC**. In these activities, the goals and measures previously gathered are analyzed, and goals and measures suitable for SPC are selected. These activities are necessary to assure the quality of the patterns that will be further extracted because goals and measures obtained in the previous activity may not be appropriate to SPC. Thus, only goals and measures useful for SPC must be selected. Some requirements can be considered to evaluate the suitability of measures for SPC [Barcellos et al. 2013, Tarhan and Demirors 2006, Tarhan and Demirors 2008]. For example, the measure must be aligned to organizational or project goals; the measure must be able to support decision making; the measure must be able to support software process improvement; the measure must be related to a critical process; the measure must be able to characterize process behavior; the measure must have appropriate granularity level (i.e., it should allow a frequent analysis of process behavior, as well as obtaining enough data to SPC); the measure must be correctly normalized (if applicable). Other requirements can be found in [Barcellos et al. 2013].

Finally, it is necessary to **Review/Refine Processes, Measurement Goals and Measures for SPC**, which consists of refining/adjusting the selected elements (processes, goals, and measures) to ensure that they serve as a basis for pattern extraction. The purpose is to produce a set of processes, goals, and measures with good reuse potential. One can, therefore, for example, decompose goals to make their

relationship with measures more direct. Similarly, processes can be broken down into subprocesses to make their relationship with goals and measures more direct. Measures can also be adjusted to fit the considered processes and goals better. The relations between the elements also must be reviewed since some relations may not have been captured until then. Moreover, the refinements made in some elements may imply new relations between them.

5.2 Develop Pattern Language

Once the source from which the patterns will be extracted is defined, the PL can be developed. Figure 3 shows this activity in detail.

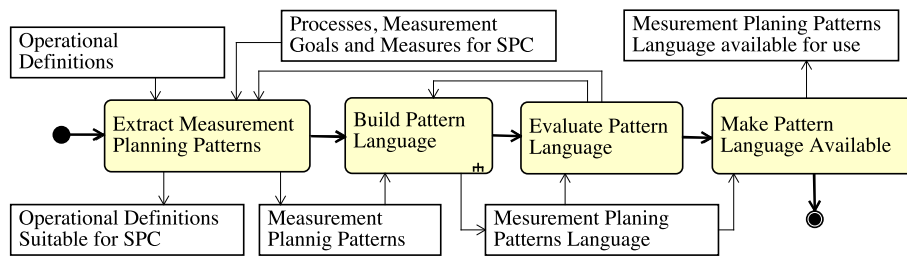


Figure 3: Detailing of Develop Pattern Language

The development of a PL begins with the Extract Measurement Planning Patterns activity, which consists of analyzing the set of processes, measurement goals, and measures resulting from the previous activity, aiming to identify measurement planning patterns. A measurement planning pattern can be related to one or more processes and follows the GQM format [Basili et al. 1994]. This way, it includes a measurement goal, questions that indicate information needs that must be met, and measures that meet the information needs. Each measure must have an operational definition suitable for SPC. If the measure does not have an operational definition, it must be defined. If it already exists (i.e., if it was obtained from the sources of measures), it should be analyzed and adjusted to be suitable for SPC when necessary. Useful information about operational definitions of measures for SPC can be found in [Barcellos et al. 2013]. SAMPPLa provides a template to define a pattern (to be shown later), and it indicates which information should be provided when establishing an operational definition.

Once the patterns are identified, it is possible to Build Pattern Language. In line with Quirino et al.'s (2017) proposal for representing pattern languages, in this activity, the models referring to the structural and behavioral perspectives of the PL must be elaborated, adopting the visual notation defined by Quirino et al. (2017). This activity is broken down into *Develop Structural Model*, which produces a model presenting the patterns that compose the PL and the structural relationships between them (e.g., dependence and composition), and *Develop Behavioral Model*, which produces a model that represents the process to be followed to apply the patterns. Information on how to build these models can be found in [Quirino et al. 2017].

After creating the PL, it is time to Evaluate Pattern Language, verifying if it can achieve its purpose. For that, one can, for instance, verify if the patterns are truly related to identified processes, if they are described satisfactorily (e.g., if the information is clear enough so that other people can use the pattern), if the measures are able to

monitor the goals they are related to, and if other patterns are needed). It is also necessary to verify if the PL representation is correct (e.g., if the notation was correctly used) and if the relationships established in the structural and behavioral models are correct and consistent (e.g., if the flows in the behavioral models are consistent with the dependencies in the structural model). Some ways to evaluate the PL are through peer reviews, where experts evaluate the created PL to identify problems and opportunities for improvement, and experimental studies, where a group of people uses the PL and provides feedback on its usability and usefulness. If needed, adjustments should be made in the PL by returning to previous activities. Finally, it is necessary to **Make Pattern Language Available**. The PL can be made available through a textual specification. However, to enhance the use of the PL, it is suggested to provide a computational tool.

6 MePPLa: Measurement Planning Pattern Language

We followed SAMPPLa to develop MePPLa. Next, we present MePPLa, by describing results produced when we performed each of the SAMPPLa activities. The evaluation of MePPLa is addressed in Section 7.

Define Pattern Language Purpose and Application Context - MePPLa is a PL with the *purpose* of helping measurement planning for SPC. The *application context* of MePPLa encompasses organizations that intend to submit processes to SPC. That includes, for example, CMMI-DEV level 3 or MR-MPS-SW level C organizations wanting to implement SPC practices to achieve the high maturity levels and organizations intending to evolve the processes addressed by MePPLa to the highest levels in the continuous view of CMMI-DEV.

Select Processes - Considering the established purpose and application context, and the results of the systematic mapping study and the survey that investigated processes, goals, and measures used in SPC in software organizations [Brito et al. 2018], we selected the Project Management, Coding and Testing processes as the ones to be first treated in MePPLa. The Coding and Testing processes were both identified in the survey and the literature, with the Testing process being the most cited in both studies. The Project Management process, in turn, was identified only in the literature. Still, 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. Due to the PL purpose and application context, the decision considered that in the first steps to implement SPC practices aiming at high maturity, organizations should include a process covering management aspects, a process related to development, and a process addressing quality aspects [CMMI Institute 2018]. Although there are other suitable processes (e.g., Inspection, Requirements Development), for pragmatic reasons, we needed to select the processes to be addressed first. Others can be added in the future.

Identify a Set of Measurement Goals and Measures related to the Processes - The set of measurement goals and measures related to the processes was also extracted from the systematic mapping and survey [Brito et al. 2018]. As a result of these studies, 84 measures, 15 processes, and 47 goals used in SPC were identified (an extract of the results was shown in Section 4). For the development of MePPLa, we considered the

goals and measures related to the Project Management, Coding, and Test processes.

Select Measurement Goals for SPC & Select Measures for SPC - We analyzed the measurement goals and measures and selected the goals and measures related adequately to Project Management, Coding, and Testing processes. For example, in the set of goals and measures obtained from the studies, the goal “Improve defect detection to reduce the number of delivered defects” was related to the Coding process. However, this goal is not related to this process because Coding does not deal with defect detection. Therefore, we did not select that goal. The measure “defect removal effectiveness” was related to the Testing process. However, they are not truly related because the Testing process is not responsible for removing defects. Therefore, we did not select this measure. When selecting the measures, we also considered some of the requirements defined in [Barcellos et al. 2013] (e.g., the ones cited in Section 5), which can be used to evaluate the suitability of a measure for SPC.

Review/Refine Processes, Measurement Goals and Measures for SPC - The set of processes, goals, and measures obtained in the previous activities was reviewed and adjusted to serve as a basis for pattern extraction. Concerning goals, we broke down general goals into specific goals, which could be individually considered for pattern identification. For example, from the “Monitor project cost and schedule” goal, we defined “Monitor project cost” and “Monitor project schedule” goals. This way, it would be possible to identify patterns related to cost and schedule separately. As for the measures, from specific measures with a lower potential of reuse, we defined general measures. For example, from the “code size estimation accuracy” and “file estimation accuracy” measures, we defined the “size estimation accuracy” measure. Concerning the processes, some of them were broken down into subprocesses. For example, Project Management was broken down into Project Planning and Project Monitoring, and Defect Fixing was identified as a subprocess of Coding. After all the refinements are performed, the resulting set of processes, measurement goals, and measures is the source of patterns that will be used as an input of the following activity. Table 3 presents a fragment of the results of this activity, containing elements associated with the Coding process. The goals associated with the Defect Fixing subprocess are identified by (*).

Process: Coding	
Goals	Measures
Control variation in coding processes; Improve product quality; Evaluate process quality effectiveness; Understand software processes performance; Monitor process quality; Verify quality goals achievement; Reduce defects in the products	Defect density (<i>number of detected defects/product size</i>)
Improve productivity	Rework (<i>effort spent on rework/ product size</i>)
	Productivity (<i>effort / product size</i>)
Reduce cost due to poor quality performance	Cost of poor quality in the code (<i>cost of correcting internal failure + cost of correcting external failure</i>)
	Cost of code quality (<i>cost of appraisal + cost of defect prevention + cost of correcting internal failure + cost of correcting external failure</i>)

Reduce injected defect (*); Manage defect injection distribution in different kinds of activities	Defect injection rate (by phase) (<i>number of injected defects/number of removed detected defects</i>)
Manage the effectiveness of defect removal activities (*)	Defect removal effectiveness (<i>number of removed defects in requirements (or design, coding, and testing) /number of detected defects</i>)
Understand and predict product quality; Win the market competition	Defect injection rate (by phase) (<i>number of injected defects/number of removed detected defects</i>)
	Defect density (<i>number of detected defects/product size</i>)
Monitor process performance	Productivity (<i>effort / product size</i>)

Table 3: Extract of the refined elements associated with the Coding process

Extract Measurement Planning Patterns - Once the source for patterns extraction was defined in the last activity, we analyzed it and identified the patterns of MePPLa. In this activity, we also established operational definitions appropriate to SPC for the measures included in the patterns. As stated earlier, in MePPLa, each pattern is a solution related to a process and comprising a measurement goal, information needs, and measures. The problem that the pattern seeks to solve is monitoring the achievement of that goal and the behavior of the related process. For example, we identified the “Defect Fixing Effectiveness” pattern (related to the Coding process), which is associated with the goal "Improve defect fixing effectiveness." That means that the pattern proposes as a solution containing the measure defect fixing effectiveness, whose data collection and analysis will help characterize the behavior of the Coding process and monitor whether that goal is being achieved.

During patterns identification, we took some actions aiming to define patterns suitable for the MePPLa application context. For instance, we disregarded goals that are too general (e.g., “Win the market competition” and “Understand software processes performance”) since using goals that are too general is not a good practice when implementing SPC [CMMI Institute 2018]. It would be preferable to define performance and quality measurement goals to be monitored by using SPC techniques [CMMI Institute 2018]. As for the measures, we did not consider the ones with less potential to reuse (e.g., the ones related to very specific contexts).

For 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 using MePPLa, the user can choose which granularities he/she wants to consider in the measurement plan. For the Coding process, we identified patterns containing measures related to productivity, defect density and rework. As for the Test process, we extracted patterns related to delivery defects, detected defects, and effort spent on test activities.

To refine the alignment between processes and related patterns, we broke down some processes. The selected processes were decomposed considering the measures related to them and our knowledge of software processes (two of the authors are very experienced in software process reference models and standards). The Project Management process was broken down into Project Planning and Project Monitoring and Control, and the Testing process into Test Preparation and Test Execution. It is

worth mentioning that, although we did not use process models such as CMMI-DEV and standards such as ISO/IEC/IEEE 12207 (2017), they could be used as a basis to decompose processes. Table 4 presents a fragment of the results produced in this activity. It shows goals and measures identified to compose the patterns related to the Coding process. The goals associated with the Defect Fixing subprocess are identified by (*).

Process: Coding	
Measurement Goals	Measures
Improve coding productivity	Productivity (effort / product size)
Improve product quality	Defect density (number of detected defects/product size)
Improve coding quality	Rework (effort spent on rework/ product size)
Improve product reliability	Mean time between failures (Sum of time between failures/(number of failures -1))
Improve defects fixing effectiveness (*)	Defect removal effectiveness (number of removed defects in requirements (or design, coding, and testing) /number of detected defects)
Reduce injected defects (*)	Defect injection rate (by phase) (number of injected defects/number of removed detected defects)

Table 4: Elements selected to compose patterns related to the Coding process

For each pattern identified, a detailed description was established, including the operational definition of each measure. Appendix A shows, as an example, the description of the “Defect Fixing Effectiveness” pattern. For simplification reasons, we omitted pieces of information regarding the operational definition of base measures.

According to Barcellos et al. (2013), to enable consistent measurements, ideally, the operational definition of a measure used in SPC should include name, description, mnemonic, measurable property, measurable entity, scale values, 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 pattern's description. They 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 Appendix A in italics and delimited by << >>) aiming to help organizations complete the operational definition of the selected measures. To exemplify a pattern description completely filled out when applied, Appendix B shows the description of the “Defect Fixing Effectiveness” pattern filled out by an SPC professional to a specific organization.

Build Pattern Language - Once the patterns were defined, we used OPL-ML [Quirino et al. 2017] and developed models for graphically representing MePPLa. As said in Section 2, although OPL-ML was proposed to represent OPLs, it was based on Software Engineering works. Its authors state that it can be used (with adjustments, if

needed) to represent pattern languages in general. To represent MePPLa, we made an adjustment in OPL-ML notation by including a construct to represent the "correlation" relation between patterns, indicating patterns whose measures are correlated, but the patterns are not dependent. Originally, OPL-ML represents only dependency and composition relations.

According to OPL-ML, a PL 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 the patterns.

The *structural model* provides information about structural relations, especially useful during measurement analysis since they reveal related measures and goals that impact others. This model can also be useful in measurement planning by helping to identify which patterns can be selected for a better analysis of goals achievement and identification of causes that may be interfering with it. Figures 4 and 5 present the structural model containing patterns related to the Project Management process and Coding process, respectively.

In MePPLa, two types of structural relations are used. Dependency (referred to in the figures by the *requires* construct) indicates when a pattern requires the application of another pattern. Correlation (referred to in the figures by the *is correlated to* construct) indicates patterns whose measures are correlated, but the patterns are not dependent. In MePPLa, when a pattern is correlated to another, it means that although a pattern does not depend on another, the measure contained in one pattern may influence values of the measure contained in the other. Thus, it is recommended to apply both to reach a deeper data analysis.

As Quirino et al. (2017) suggest, patterns can be organized into groups. In MePPLa, patterns were grouped considering the processes and sub-processes to which they relate. We used the name of the process related to the pattern group to name it. Thus, the pattern groups represent the processes addressed by MePPLa. Inside each pattern group, there are patterns related to the process the pattern group represents. For example, in Figure 4, all patterns are related to the Project Management process. Inside the group, patterns are sub-grouped according to the sub-process they relate to. For example, in Figure 4, *Schedule Performance* and *Cost Performance* are related to Project Monitoring and Control.

As shown in Figure 4, there are dependency relationships between patterns related to cost and duration estimates and between patterns related to duration and effort estimates. For example, *Activity Cost Estimate Accuracy* has a relationship of dependency with *Activity Duration Estimate Accuracy*, since, to estimate the cost of an activity, it is necessary first to estimate its duration. There are also correlation relationships between patterns. For example, in Figure 4, *Activity Effort Estimate Accuracy* is correlated to *Size Estimate Accuracy* because effort and size are related to each other, but it is not necessary to estimate first the product size to estimate the effort of an activity. The patterns from the Coding group are also related by correlation relationships (see Figure 5). For example, the pattern *Product Defect Density* is correlated to the pattern *Coding Quality* since the quality of the coding process can impact the product defect density. Still, the application of the *Product Defect Density* pattern does not depend on the application of the *Coding Quality* pattern (hence there is no dependency relation between them).

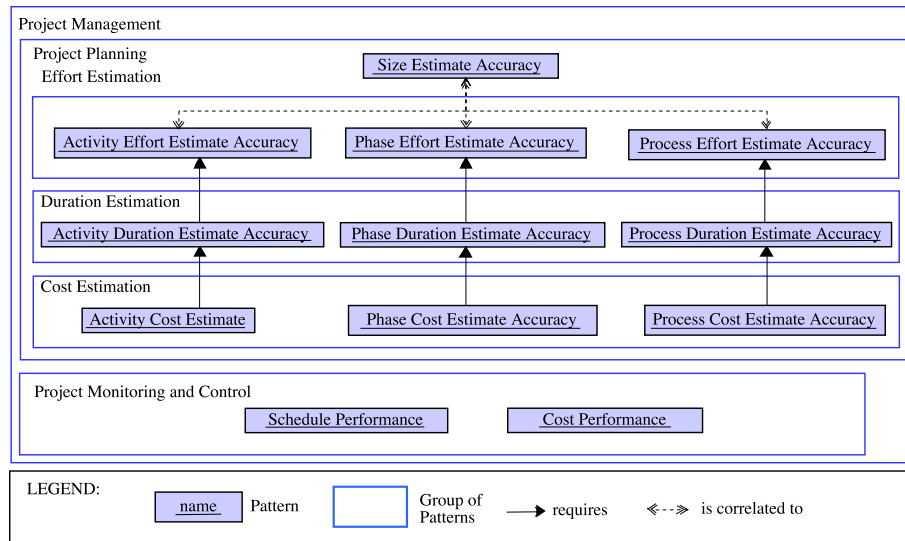


Figure 4: Structural model of the Project Management Pattern Group

During MePPLa development, the structural model was helpful for us to develop the behavioral model since the structural model indicates the dependencies that must be considered in the flows that guide pattern selection. For example, if the structural model shows that pattern B depends on pattern A, pattern A must be applied first in the behavioral model. Moreover, the relations presented in the structural models are useful to analyze data. For example, *Coding Productivity* is correlated to *Coding Quality*. I. e., when analyzing data about the former, one should also analyze data about the latter because productivity may be decreasing because there are too many problems in the code. Although the patterns are not dependent on each other (thus, in the behavioral model, it is not mandatory to apply both patterns), the structural model shows that it is possible to reach a deeper analysis when both are applied.

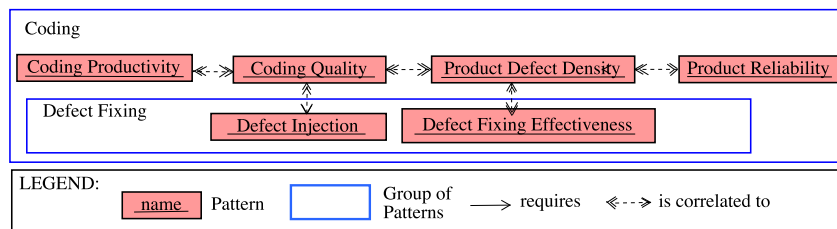


Figure 5: Structural model of the Coding Pattern Group

Once the structural model related to each process was defined, the *behavioral* model was developed. The behavioral model (or process model) has two formats: the black-box format, which provides a general view of the PL from the behavioral perspective, and the detailed format, which provides a detailed view of the PL process, containing the flows that guide pattern applications. Both formats must be understood

as a process to be followed step by step from an entry point to an endpoint. Figure 6 shows the behavioral model of MePPLa in the black-box format.

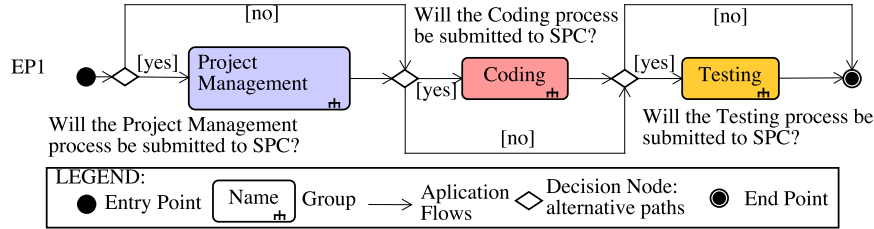


Figure 6: MePPLa behavioral model in the black-box format

In the black-box 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, it is impossible to visualize the patterns and flows within each group in the black-box format. We developed a detailed behavioral model for each pattern group (i.e., for each process), presenting the group's internal content, i.e., its patterns and flows between them. Figures 7 and 8 show the detailed behavioral model related to Coding and Project Management, respectively. The detailed behavioral model of Testing can be found in [Brito et al. 2017].

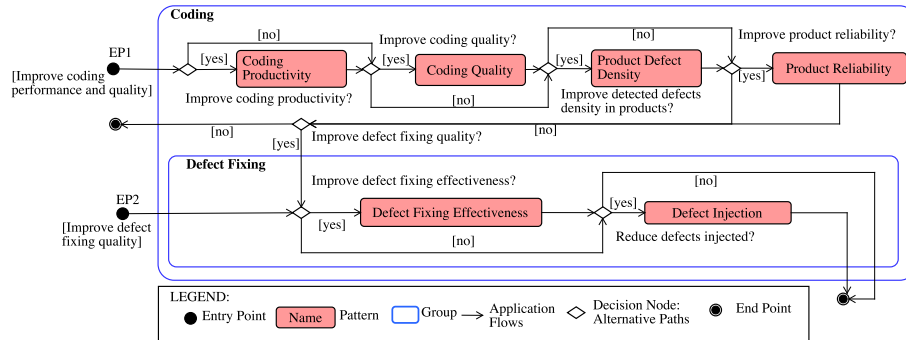


Figure 7: Behavioral model of the Coding Pattern Group

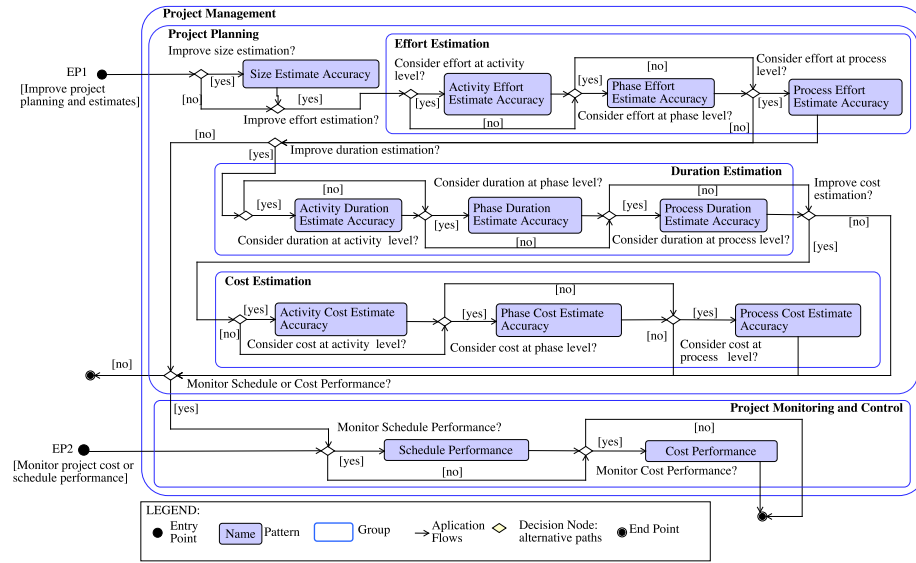


Figure 8: Behavioral model of the Project Management Pattern Group

The detailed behavioral models are consistent with the structural models. 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 pattern selection, respecting the relationships between the patterns defined in the structural model (e.g., 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 8, if the measurement goal “Improve project planning and estimates” is to be included in the measurement plan, the user should start the navigation from EP1. If this measurement goal is not relevant and “Improve project schedule and cost performance” is, the user should start the navigation from EP2. Note that if the user begins the navigation from EP1, he/she will also be guided towards the patterns related to “Improve project schedule and cost performance,” and, by following the flows, he/she 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 he/she wants to treat in the measurement plan. The goals are presented in the entry points and the decision nodes so that the user can decide which relevant goals he/she wants to create. When a pattern is applied, its description is to be included in the measurement plan.

To use MePPLa, the organization must have established its business goals, the processes related to them and selected which of these processes 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. It is worth pointing out that, in the software process improvement context, it is only possible to apply SPC to analyze the behavior of defined processes (i.e., processes established in the organization and executed in the projects). In this sense, organizations can only select to submit to SPC processes that they have already defined.

Make Pattern Language Available - In this activity, a textual specification containing the structural and behavioral models of MePPLa and a detailed description of each pattern was developed. MePPLa specification is available in the form of a document.¹ After creating MePPLa specification, we developed a tool (MePPLa Tool)² to support its use. Figure 9 shows a screenshot of the MePPLa Tool.

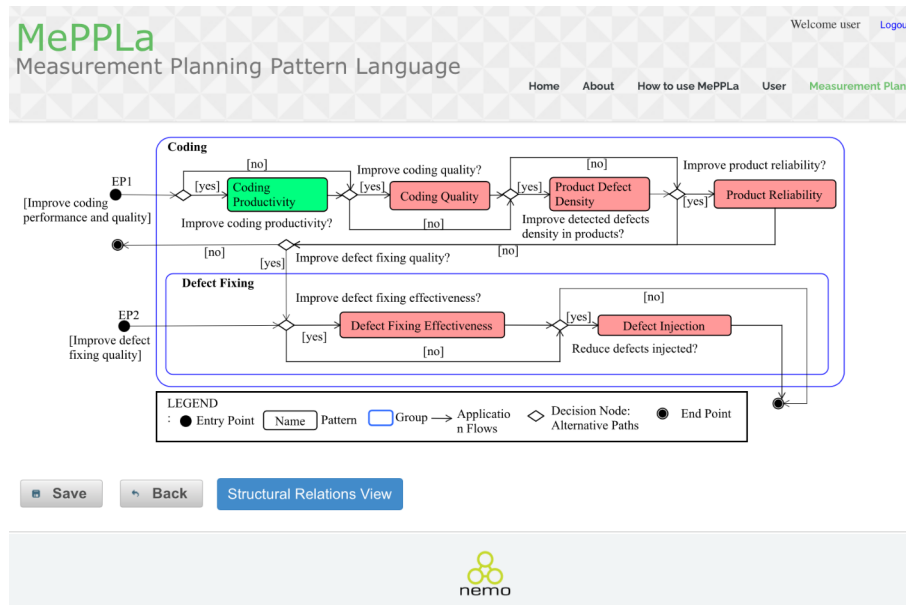


Figure 9: A screenshot of MePPLa Tool

The tool allows the user to navigate the MePPLa behavioral models and select the patterns he/she wants to include in the measurement plan. As a result, a file containing the detailed descriptions of the selected patterns (see example in Appendix B) is produced. This file can be exported in MSWord format. The information recorded in the file can be extracted and included in the organization's measurement plan, or the organization can edit the file (for example, by adding business goals) and turn it into a measurement plan. When the user selects a pattern, its description is included in the measurement plan file, and the pattern is colored green. When applying a pattern, the user has access to the pattern description and can complete information that depends on the organization context (the pieces in italics and delimited by << >> in Appendix A). Moreover, the user can look at the structural models to see the structural relations between the patterns.

¹ MePPLa specification is available at <http://nemo.inf.ufes.br/wp-content/uploads/MeiSE/MePPLaSpecification.pdf>

² MePPLa Tool is available at <http://dev.nemo.inf.ufes.br:8180/MPPL/login.faces>

7 MePPLa Evaluation

MePPLa evaluation was performed through peer reviews and experimental studies. At first, two external researchers with knowledge and practical experience in software measurement and SPC evaluated MePPLa specification through peer review. Based on this first evaluation, we improved MePPLa and developed MePPLa Tool. Then, we conducted two studies to evaluate whether MePPLa is useful to software measurement planning for SPC and whether its use is feasible. The first study [Brito et al. 2017] involved four participants (three professionals and one graduate student). Based on the study results and participants' feedback, we improved MePPLa and MePPLa Tool. After the improvements, we carried out the second study, which involved six professionals and is described as follows.

7.1 Study Planning and Execution

Using the GQM approach [Basili et al. 1994], the study *goal* is formalized as Analyze *MePPLa*, with the purpose of *evaluating the PL*, 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 consisted of three forms, a document and one computational tool: (i) a consent form, in which participants declared to accept to participate 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; (iii) MePPLa specification; (iv) a questionnaire to capture the participants' perceptions about MePPLa; and (v) MePPLa Tool, which was used by the participants to create measurement plans taking real scenarios into account.

The *procedure* adopted in the study consisted of sending to the participants the MePPLa specification, a link to access MePPLa Tool, and a document containing information about how to proceed to use and evaluate MePPLa. The participants were asked to create a measurement plan for at least one organization that aimed to submit some of its processes to SPC as a step to achieve higher maturity levels in models such as CMMI-DEV or MR-MPS-SW. Based on the instructions provided in the document, participants used MePPLa Tool, looked at MePPLa specification if they needed, evaluated MePPLa, and then answered the questionnaire.

The *questionnaire*³ included questions related to MePPLa's usefulness, adequacy of the result obtained from the use of MePPLa, quality of the produced result, and productivity of the measurement planning activity when supported by MePPLa. The questionnaire contained multiple-choice questions and a discursive question. For multiple-choice questions, participants were asked to justify the given answer. The discursive question aimed at the general evaluation and improvement of MePPLa.

³ The questionnaire used in the study is available at <https://forms.gle/bsGrrHBysz5QZ3Yr9>

The study *participants* were six professionals with experience in software measurement and SPC implementation or evaluation in software organizations (CMMI-DEV and MR-MPS-SW appraisers or implementers). One participant declared *medium* practical experience and knowledge of software measurement and SPC, and five declared *high* practical experience and knowledge of software measurement and SPC. As for PLs, two participants declared *medium* experience, one declared *little* experience, and three reported that the use of MePPLa would be their first experience with PLs.

7.2 Results and Discussions

Next, we present the main results and discussions related to each of the three indicators used in the study.

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 (with operational definition) related to processes to be submitted to SPC. As previously said, this set of information can be included in an existing measurement plan or can be used to produce a new plan by adding complementary information (e.g., business goals) not provided by the patterns.

Three participants (50%) reported that the result was adequate, and three (50%) participants considered it partially adequate. The participants who considered the results adequate indicated relevant and adequate information to the measurement plan. Moreover, they highlighted that when using MePPLa Tool, exporting and editing the document containing all the selected patterns is important to elaborate an appropriate measurement plan for a specific organization. On the other hand, the participants who considered the results partially adequate justified that they needed further information to fill out the operational definitions of the measures contained in the patterns. Concerning this issue, since the patterns defined in MePPLa are to be used by several organizations, some information cannot be predefined because they depend on the organization to which the measurement planning is developed. For these pieces of information, the pattern provides guidelines to help organizations to complete the operational definition. Considering the participants' feedback, we have improved some guidelines.

MePPLa usefulness - Five participants (83%) considered MePPLa useful or very useful. However, although considering the patterns useful to define the measurement plan, one participant reported being neutral. He informed us that he had some difficulty understanding the diagrams at the first moment. The other participants state that MePPLa is useful because it incorporates several aspects that need to be considered when using SPC, helps in choosing suitable measures, provides visibility of the processes to which they are related, and provides a basic structure for the measurement plan.

Benefits provided by MePPLa - Three participants (50%) 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 PLs. However, the other three participants (50%) considered that MePPLa contributes only partially. When using MePPLa Tool, they reported that after editing a pattern added to a measurement plan, the changes were not saved to be reused when the pattern was added to another measurement plan (the changes were saved only in the measurement plan

being created). Thus, when they added the pattern to another measurement plan, they needed to edit it again. It is important to notice that this limitation refers to MePPLa Tool and not to MePPLa itself. This limitation of the MePPLa Tool also impacted the participants' perceptions of the use of MePPLa to improve productivity. Most participants (four out six, 66.7%) reported that using MePPLa made the measurement planning activity much more productive or more productive. However, two participants (33.3%) were neutral and justified their answer in the tool limitation to save changes made in applied patterns. The tool limitation can be easily solved by enabling the users to save new versions to the patterns they change. Thus, when adding a pattern to a measurement plan, the user can choose to add its original version defined in MePPLa or a version he/she has created by editing information in the original pattern.

Three participants (50%) considered MePPLa very easy or easy to use, and three (50%) reported the difficulty degree to use MePPLa as neutral. For these participants, the most significant difficulty was understanding the symbols that make up the structural and behavior models of the pattern language.

Four participants (66.7%) highlighted that MePPLa contributed to the quality of the measurement plan since it reuses previous experiences and consolidated GQM-based patterns. However, two participants (33.3%) founded that MePPLa contributed only partially to the quality of the measurement plan because it addresses only three processes, and if the organization needs to add measures related to other processes, quality is not ensured. Concerning that, the quality of the content to be added to the measurement plan after using MePPLa is out of the scope of MePPLa. As for the number of processes treated by MePPLa, new processes can be addressed. Currently, we are adding patterns related to the Review process.

By analyzing the results and comments made by the participants, we noticed that some cited limitations refer to the supporting tool (MePPLa Tool) and not to MePPLa itself. These limitations can be easily addressed by improving the tool. However, we also noticed that half of the participants reported some difficulty understanding the visual notation used, which the first study participants have not reported. In MePPLa, we have adopted OPL-ML [Quirino et al. 2017], a modeling language proposed to represent Ontology Pattern Languages. This kind of PL is used by ontology engineers, who are very familiar with conceptual modeling and Unified Modeling Language (UML). OPL-ML reuses UML constructs; thus, it is easy for ontology engineers to understand OPL-ML constructs and diagrams. The study results suggest that measurement professionals may not be very familiar with UML constructs or conceptual models and, thus, following diagrams inspired by UML activity diagrams may introduce some difficulty. We believe that this difficulty could be minimized by improving MePPLa Tool for the user to be automatically guided flow by flow through the behavioral models.

Although the participants have reported some limitations, considering the results as a whole, we understand that the study provided preliminary evidence that MePPLa is useful, provides benefits in terms of quality and productivity, and produces adequate results. However, some improvements are needed, and some threats must be considered together with the results.

7.3 Threats to Validity

Next, we discuss some threats to the study results by following the classification defined in [Wöhlin et al. 2000].

The main threat to the study results refers to the reduced number of participants and little profile variety. This threat affects *external* and *conclusion validity*. The former refers to the ability to repeat the same behavior with different groups, while the latter is related to the ability of the study to generate conclusions.

There are also important threats affecting the *construct validity*, which regards the relationship between the study instruments and participants and the theory being tested. In this context, participants may have given answers that do not reflect reality due to personal expectations. To minimize this threat, we informed the participants that the study did not represent any type of personal assessment nor aims to evaluate MePPLa. The questionnaire used in the study may also have been a threat. The questions may not have been easily understood by the participants or not enough to capture the participants' perceptions. Moreover, there may have been biases in the drafting of the questions. Another important threat is that the participants provided feedback about MePPLa before using the measurement plans produced during the study. In addition, the use of the supporting tool to evaluate MePPLa is also a threat. By using the tool, participants evaluated not only MePPLa but also aspects related to the tool itself. The two last cited threats affect not only construct validity but also conclusion validity.

Finally, regarding *internal validity*, which is related to the ability of a new study to repeat the behavior of the current study with the same participants and objects, there is the threat of communication and sharing of information among participants. To minimize this threat, we sent the instruments to the participant's email so that he/she could individually do the evaluation. All these threats limit the generalization of results, and therefore they should be considered as preliminary results that cannot be generalized.

8 Related Work

In Section 2.1, we mentioned some works addressing software measurement. When compared to [Florac et al. 1999, ISO/IEC 2001, McGarry et al. 2002, Siviyy et al. 2005, Tarhan and Demirörs 2006, ISO/IEC 2007, ISO/IEC 2011, ISO/IEC/IEEE 2017], our proposal presents similarities and differences. As for similarities, all works concern software measurement, and some of them [Florac and Carleton 1999, ISO/IEC 2007] provide measures that organizations can reuse when creating measurement plans. However, only MePPLa organizes the measures in a visual representation aiming at favoring reuse in a process and goal-oriented approach that helps measurement planning. Furthermore, MePPLa has a supporting tool (MePPLa Tool) that aids in elaborating measurement plans, contributing to improving productivity and quality.

Concerning GQM [Basili et al. 1994], while it guides on how to identify measures by considering information needs (questions) derived from goals, MePPLa provides a “ready” solution based on the GQM format. As a result, from its goals, the organization selects measurement planning patterns containing information needs and measures (with operational definition) to monitor the goals. In practice, when using GQM, the organization has the method to follow but does not have the goals, information needs,

and measures to be added to its measurement plan. By using MePPLa, the organization has a method to follow (given by a behavioral model) and the goals, information needs, and measures that can be added to the organization's measurement plan.

We also have presented in Section 2.2 two works proposing pattern languages related to software measurement [Nikelshpur 2011 and Braga et al. 2012]. Comparing these works with MePPLa, some important differences can be highlighted: (i) the PLs proposed by Nikelshpur (2011) and Braga et al. (2012) do not address SPC; (ii) the patterns of these PLs provide guidelines for project planning and estimation, while MePPLa provides patterns based on the GQM format and their use helps measurement planning aiming at SPC; (iii) the patterns of the PLs do not define measures either provide an operational definition of measures; (iv) the PLs do not use a cognitively rich visual notation (the PL proposed by Nikelshpur (2011) is not even graphically represented), while MePPLa uses OPL-ML [Quirino et al. 2017] and separates the structural and behavioral perspectives of the PL.

9 Conclusion

The importance of SPC for the software industry has increased due to the interest of organizations in achieving high maturity [Fernandez-Corrales et al. 2013]. The use of SPC enables understanding the processes' behavior and predicting their performance [Razmochaeva et al. 2019]. Considering that one of the main difficulties organizations face when implementing SPC is selecting appropriate measures [Tarhan and Demirors 2006, Tarhan and Demirors 2008, Barcellos et al. 2013, Schots et al. 2014], we propose using a PL to support measurement planning for SPC. As we argued before, in this work, we adopt a broader view of what a pattern is.

This paper presented MePPLa (Measurement Planning Pattern Language), developed using a systematic approach named SAMPPLa (Systematic Approach for creating Measurement Planning Pattern Languages), also defined in this work. MePPLa provides measurement planning patterns that help organizations create measurement plans by navigating behavioral models that guide patterns selection and usage. Moreover, the measurement patterns provided by MePPLa were extracted from studies that investigated measures used in practical SPC initiatives. MePPLa has a supporting tool (MePPLa Tool), which was also presented in this paper. Our goal with the work addressed in this paper is to bring an alternative way to represent, access, and reuse knowledge in the form of measurement planning patterns organized in a pattern language.

MePPLa is aimed to support measurement planning. Therefore, we point out that it should be used in the context of a broader measurement process that covers planning, data collection, and analysis (e.g., as suggested in ISO/IEC/IEEE 15539).

Although MePPLa was built to be used aiming at SPC, it can also be used (with some adjustments in operational definitions of measures) in software measurement in general. Moreover, although SAMPPLa was created to develop/evolve pattern languages to support measurement planning aiming at SPC, it can be adapted to disregard SPC particularities and guide the creation of PLs with different purposes, as for software measurement in general.

MePPLa can continuously evolve (e.g., new patterns can be added). In its current version, MePPLa includes only patterns extracted from the systematic mapping of the

literature and the survey [Brito et al. 2018]. As future work, MePPLa's patterns can be refined, and new patterns can be extracted from other studies (e.g., by analyzing measurement repositories of different organizations). Moreover, different patterns can be related to the same process but to different development approaches (e.g., plan-drive, agile). Therefore, the patterns' format can be improved considering the Goal-Question-Indicator-Metric (GQ(I)M) [Park et al. 1996] format.

Research on software measurement continues to be a hot topic today. Although the benefits of using of software measurement are well known, problems in their practice are still ongoing [Tekin et al. 2020]. We consider that by providing the knowledge in a structured way and a support tool to reuse it in the creation of measurement plans, this work brings contributions to the state of the art (knowledge capture and structuring) and to the state of the practice (tool to support the creation of measurement plans). Currently, much value has been given to information obtained from the analysis of data collected throughout the software development process. Based on this information it is possible to make decisions aimed at improving software products and processes. For useful information to be obtained, meaningful data must be available, that is, it is necessary to measure what really matters to the organization. In this sense, MePPLa contributes by assisting in identifying measures to data aligned to the organization's goals aimed at software process improvement. Moreover, one of the difficulties organizations face when moving from initial process maturity levels to high maturity levels (e.g., from CMMI levels 2 and 3 to CMMI levels 4 and 5) is related to software measurement, because if the measures adopted by the organization are not suitable for SPC when it starts the practices to achieve levels 4 or 5, the organization needs to define new measures and collect new data to be used in SPC. Achieving the amount of data needed to properly apply SPC techniques demands time and effort that can be minimized if the organization defines measures suitable for SPC beforehand. MePPLa helps in this matter by providing measures that have already been successfully used in SPC by other organizations. By reusing the measures (i.e., by applying the measurement planning patterns), an organization can define measurement plans suitable for SPC since level 3 and, in this way, when the organization starts the level 4 practices, it has already collected data to perform SPC, which contributes to decrease the time necessary to achieve levels 4 or 5.

Among the limitations of this work, we can highlight the MePPLa evaluation, which consisted of peer review by two researchers and two studies involving ten participants. Although the evaluation performed so far has not included a case study in a software organization, the studies' participants are professionals who have worked mainly as implementers or appraisers of process maturity models (CMMI, MPS BR and ISO) and, as such, have acted in several organizations. Defining measurement plans to implement SPC in software organizations aiming at high maturity is part of their jobs. Thus, they have the capacity (knowledge and experience) needed to evaluate MePPLa. Hence, Even though the studies performed so far are limited and more robust evidence are still needed, the obtained preliminary are an indication of MePPLa usefulness. Another limitation regards SAMPPLa evaluation, as it has only been used by its creators to define MePPLa and by one more researcher to extend MePPLa by adding the Review process (an ongoing work supervised by the first and second authors).

As future work, we intend to improve MePPLa Tool considering the study results, carry out a case study to evaluate MePPLa better, evolve MePPLa to treat other

processes, and include different patterns. As said above, we are currently working on adding patterns related to the Review process to MePPLa. We also plan to evaluate the use of SAMPPLa by third parties in a case study to evolve MePPLa by considering other sources and processes.

Finally, creating a PL involves a lot of judgment and tacit knowledge. For example, it may not be simple to identify the suitable elements to compose the source for pattern extraction or to extract fine patterns. This is a limitation of our approach. To help with this matter, we intend to define guidelines for SAMPPLa activities to better guide the users in creating or evolving PLs.

References

- [Alexander et al. 1977] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S.: “A pattern language”; Oxford University Press, USA (1977).
- [Barcellos et al. 2010] Barcellos, M. P., Falbo, R. A., Rocha, A. R.: “Establishing a well-founded conceptualization about software measurement in high maturity levels”; *Proc. 7th Int. Conf. on the Quality of Information and Communications Technology*, Porto, Portugal (2010), 467–472.
- [Barcellos et al. 2013] Barcellos, M. P., Falbo, R. A., Rocha, A. R.: “A strategy for preparing software organizations for statistical process control”; *J. of the Brazilian Comp. Society*, 19, 4 (2013), 445–473, <https://doi.org/10.1007/s13173-013-0106-x>.
- [Basili et al. 1994] Basili, V. R., Rombach, H. D., Caldiera, G.: “Goal Question Metric Paradigm”; *Encyclopedia of Software Engineering*, USA: John Wiley & Sons, Inc. (1994).
- [Braga et al. 2012] Braga, M. R. R., Bezerra, C. I. M., Monteiro, J. M., Andrade, R.: “A pattern language for agile software estimation”; *Proc. of the 9th Latin-American Conf. on Pattern Languages of Programming*, Natal, RN, Brazil (2012).
- [Brito et al. 2017] Brito, D. F., Barcellos, M. P., Santos, G.: “A Software Measurement Pattern Language for Measurement Planning aiming at SPC”; *16th Brazilian Symposium on Software Quality*, RJ, Brazil (2017).
- [Brito et al. 2018] Brito, D. F., Barcellos, M. P., Santos, G.: “Investigating measures for applying statistical process control in software organizations”; *J. of Software Engineering Research and Development*, Springer Berlin Heidelberg, 6, 10 (2018).
- [Buschmann et al. 2007] Buschmann, F., Henney, K., Schmidt, D.: “Pattern-oriented Software Architecture: On Patterns and Pattern Language”; John Wiley & Sons Ltd (2007).
- [Caivano 2005] Caivano, D.: “Continuous Software Process Improvement through Statistical Process Control”; *Proc. of the Ninth European Conference on Software Maintenance and Reengineering*, (2005), 288–293, doi: <https://doi.org/10.1109/CSMR.2005.20>.
- [CMMI Institute 2018] CMMI Institute: “CMMI for Development”; Version 2.0, Carnegie Mellon University, Pittsburgh (2018).
- [Coplein 1998] Coplien, J. O.: “Software design patterns: common questions and answers”; In: *The patterns handbook: Techniques, strategies, and applications* 13 (1998): 311.
- [Falbo et al. 2013] Falbo, R. A., Barcellos, M. P., Nardi, J. C., Guizzardi, G.: “Organizing ontology design patterns as ontology pattern language”; *Proc. of ESWC (2013), Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 7882 (2013), 61–75.
- [Ferdinandi 2002] Patricia L. F.: “A Requirements Pattern: Succeeding in the Internet Economy”; Addison Wesley (2002).

- [Fernandez-Corrales et al. 2013] Fernandez-Corrales, C., Jenkins, M., Villegas, J.: “Application of Statistical Process Control to Software Defect Metrics: An Industry Experience Report”; 7th ACM/IEEE Int. Symp. on Empirical Soft. Engineering and Measurement, (2013), 323–331.
- [Florac and Carleton 1999] Florac, W. A., Carleton, A. D.: “Measuring the Software Process: Statistical Process Control for Software Process Improvement”; Addison Wesley, Boston (1999).
- [Florac et al. 2000] Florac, W. A., Carleton, A. D., Barnard, J. R.: “Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process”; IEEE Software, 17, 4 (2000), 97–106.
- [Greenfield et al. 2004] Greenfield, J., Short, K., Cook, S., Kent, S. “Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools”, John Wiley & Sons, Canada (2004).
- [Grossi et al. 2014] Grossi, L., Calvo-Manzano, J.A., Feliu, T.S.: “High-maturity levels: achieving CMMI ML-5 in a consultancy company”, J. Softw. Evol. and Proc., 26 (2014), 808–817.
- [Hevner et al. 2004] Hevner, A., March, S., Park, J., Ram, S.: “Design Science in Information Systems Research”; MIS Q. 28 (2003), 75–105. <https://doi.org/10.2307/25148625>.
- [Hevner 2007] Hevner, A. R. A.: “Three Cycle View of Design Science Research”; Scandinavian Journal of Information Systems, 19, 2 (2007), 87–92.
- [IEEE/ISO/IEC 2017] IEEE/ISO/IEC 12207: “ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes”; (2017).
- [ISO/IEC/IEEE 2017] ISO/IEC/IEEE 15939: “Systems and software engineering — Measurement process”; (2017).
- [ISO/IEC 2001] ISO/IEC 9126-1: “Software engineering — Product quality — Quality model”; (2001).
- [ISO/IEC 2007] ISO/IEC 25020: “Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Measurement reference model and guide”; (2007).
- [ISO/IEC 2011] ISO/IEC 25010: “Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models” (2011).
- [Jörg and Frederik 2007] Jörg, F. and Frederik, B.: “Pattern Languages: An approach to manage archetypal engineering knowledge”; Guidelines for a Decision Support Method Adapted to NPD Processes (2007).
- [Kamthan 2007] Kamthan, P.: “A Perspective on Software Engineering Education with Open Source Software”; International Journal of Open Source Software and Processes, 4, 3 (2012), 13–25.
- [Kitchenham and Charters 2007] Kitchenham, B., Charters, S.: “Guidelines for performing systematic literature reviews in software engineering”; EBSE 2007-001. Keele University and Durham University Joint Report, UK (2007).
- [Kitchenham et al. 2007] Kitchenham, B., Jeffery, D. R., Connaughton, C.: “Misleading metrics and unsound analyses”; IEEE Software, 24, 2 (2007), 73–78.
- [López et al. 2018] López, P., Mabe, J., Etxeberria, L., Gorritxategi, E.: “Iterative Prototyping Methodology for the Development of Innovative and Dependable Complex Embedded Systems Through SPC&KPI Techniques”; In: PROFES (2018). Lecture Notes in Computer Science, Springer, Cham, 11271 (2018), 65–80, https://doi.org/10.1007/978-3-030-03673-7_5.
- [Maisikeli 2020] Maisikeli, S. G.: “Tracking Stability of Software Evolution Using Statistical Process Control Limit”; Seventh International Conference on Information Technology Trends (ITT), (2020), 156–160. <https://doi.org/10.1109/ITT51279.2020.9320782>.

- [McGarry et al. 2002] McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J. and Hall F.: “Practical Software Measurement: Objective information for decision makers”; Addison Wesley, Boston, USA (2002).
- [Nikelshpur 2011] Nikelshpur, D.: “The Art of Software Estimation Pattern Language”; 18th Conf. on Pattern Languages of Programs (PLOP), Portland, Oregon, USA (2011).
- [Park et al. 1996] Park, R. E., Goethert, W. B., Florac, W. A.: “Goal-Driven Software Measurement—A Guidebook”; Software Engineering Institute Carnegie Mellon University.
- [PSM/NDIA/INCOSE 2021] PSM/NDIA/INCOSE: “Practical Software and Systems Measurement Continuous Iterative Development Measurement Framework – Part 1”; Version 2.1 (2021), available at <https://www.psmnc.com/CIDMeasurement.asp>, accessed in oct. 2021.
- [Quirino et al. 2017] Quirino, G. K. S., Barcellos, M. P., Falbo, R. A.: “OPL-ML: A Modeling Language for Representing Ontology Pattern Languages”; In: *Advances in Conceptual Modeling. ER 2017. Lecture Notes in Computer Science*, Springer, Cham, 10651 (2017),
- [Quirino et al. 2018] Quirino, G. K. S., Barcellos, M. P., Falbo, R. A.: “Visual notations for software pattern languages: a mapping study”; *Proc. of the 32nd Brazilian Symposium on Software Engineering* (2018), <https://doi.org/10.1145/3266237.3266266>.
- [Razmochaeva et al. 2019] Razmochaeva, N. V., Semenov, V. P., Bezrukov, A. A.: “Investigation of Statistical Process Control in Process Automation Tasks”; *22nd Int. Conference on Soft Computing and Measurements (SCM)*, St. Petersburg, Russia (2019), 248-251.
- [Santos et al. 2015] Santos, G., Prikladnicki, R., Conte, T., Rocha, A. R., Travassos, G. H., Franco, N., Weber, K. C.: “Towards Successful Software Process Improvement Initiatives: Experiences from the Battlefield”; In: *Americas Conf. on Information Systems (AMCIS)*, (2015).
- [Schots et al. 2014] Schots, N., Rocha, A., Santos, G.: “A Body of Knowledge for Executing Performance Analysis of Software Processes”; In *Proc. of the 26th Int. Conf. on Software Engineering & Knowledge Engineering (SEKE 2014)*, Vancouver, Canada (2014), 560-565.
- [SIVIY et al. 2005] Siviyy, J., Penn, M. L., Harper, E., 2005.: “Relationship Between CMMI and Six Sigma”; Technical Note, CMU/SEI-2005-TN-005, (2005).
- [Softex 2021] “MPS.BR-Melhoria de Processo do Software Brasileiro, Guia Geral”. In: <http://www.softex.br/mpsbr> (Portuguese only).
- [Solingen and Berghout 1999] Solingen, R., Berghout, E.: “The Goal/Question/Metric Method: a practical guide for quality improvement of software development”; New York: McGraw-Hill Publishing Company (1999).
- [Tahir et al. 2018a] Tahir, T., Rasool, G., Mehmood, W., Gencel, C.: “An Evaluation of Software Measurement Processes in Pakistani Software Industry”; *IEEE Access*, 6 (2018), 57868-57896.
- [Tahir et al. 2018b] Tahir, T., Rasool, G., Noman, M.: “A Systematic Mapping Study on Software Measurement Programs in SMEs”; *e-Informatica Soft. Engineering J.*, 12, 1 (2018), 133–165.
- [Tarhan and Demirors 2006] Tarhan, A., Demirors, O.: “Investigating suitability of software process and metrics for statistical process control”; *Software Process Improvement*, 4257 (2006), 88–99.
- [Tarhan and Demirors 2008] Tarhan, A., Demirors, O.: “Assessment of Software Process and Metrics to Support Quantitative Understanding”, *Lecture Notes in Computer Science*, 4895 (2008), 102-113.
- [Tarhan and Demirors 2012] Tarhan, A., Demirors, O.: “Apply quantitative management now”, *IEEE Software*, 29 (2012), 77-85.

- [Tekin et al. 2020] Tekin, N., Kosa, M., Yilmaz, M., Clarke, P., Garousi, V.: “Visualization, Monitoring and Control Techniques for Use in Scrum Software Development: An Analytic Hierarchy Process Approach”; In: EuroSPI (2020). Communications in Computer and Information Science, Springer, 1251 (2020), https://doi.org/10.1007/978-3-030-56441-4_4.
- [Tešanovic 2005] Tešanovic, A. "What is a pattern." Dr. ing. course DT8100 (prev. 78901/45942/DIF8901) Object-oriented Systems (2005).
- [Wieringa 2014] Wieringa, R.: “Design Science Methodology for Information Systems and Software Engineering”; Springer Heidelberg, ISBN 978-3-662-43839-8, (2014).
- [Wheeler and Chambers 2010] Wheeler, D. J., Chambers, D. S.: “Understanding Statistical Process Control”; 3rd ed. Knoxville - SPC Press (2010).
- [Wöhlin et al. 2000] Wöhlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: “Experimentation in Software Engineering: An Introduction”; The Kluwer International Series in Software Engineering, Kluwer Academic Publishers (2000)

Appendix A

Defect Fixing Effectiveness	
Name: Defect Fixing Effectiveness Process/Sub-process: Coding / Defect Fixing Goal: Improve defect fixing effectiveness. Information Needs: What is the defect fixing effectiveness? Measures: Defect Fixing Effectiveness, Number of Fixed Defects, Number of Defects. Operational Definition of Measures:	
Derived Measure	Defect Fixing Effectiveness
Mnemonic	DFE
Description	Measure used to quantify the defect fixing effectiveness, which is given by the ratio between the number of fixed defects and the number of detected defects.
Measurable Entity	Defect Fixing Sub-process
Measurable Property	Effectiveness
Scale Value	Positive real numbers accurate to two decimal places.
Measurement Unit	-
Formula	$DFE = (NFD / ND)$
Measurement Procedure	Calculate the defect fixing effectiveness using the formula for calculating the measure, considering the same product (or portion of product) for both measures in the formula.
Measurement Periodicity	<i><<A frequency (e.g., weekly or biweekly) should be established for data collection. The frequency should enable several measurements during the same project, so it is possible to obtain the amount of data suitable for SPC. >></i>
Measurement Responsible	<i><<Indicate the role responsible for collecting data for the measure. It is recommended that the measurement responsible is the data provider >></i>
Measurement Moment	<i><<Indicate the moment at which data collection and recording should be performed. The moment of collection should be an activity of the project process or an organizational process.>></i>
Measurement Analysis Procedure	<p>For process behavior analysis (organizational context):</p> <ul style="list-style-type: none"> - Represent in control chart values collected for the measure in several projects. - Obtain the process control limits and analyze the process behavior: <ul style="list-style-type: none"> (i) If the values pass the stability tests, the process is deemed 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 more than 2σ from the central limit; Test 3: there are at least four out of five successive points at the same side and 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 the stability tests, the process is unstable. It is necessary to investigate the special causes, identify corrective actions and execute them. <p>Note: If there is a baseline already established for the process, it is necessary to verify if a new baseline needs to be established. <i><< Inform when a new baseline should be established (e.g., frequency or criteria to be considered, such as the range difference considering the previous baseline)>></i></p> <p>For quantitative project management (project context):</p> <ul style="list-style-type: none"> - Represent in control chart values collected for the measure in the project. - Analyze the process behavior considering the organizational behavior expected for it (i.e., using the process baseline as reference). <ul style="list-style-type: none"> (i) If the values pass the stability tests considering the process baseline as a reference, then the process behaves according to the behavior expected for it in the organization. (ii) If the values do not pass the stability tests considering the process baseline as a 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	<<Indicate the periodicity based on a time period (e.g., fortnightly) or on an amount of new data collected (e.g., each four new values collected). Different periodicities can be established to analyze data in the project and the organizational contexts.>>
Analysis Responsible	<<Indicate the role responsible for analyzing data collected for the measure>>
Analysis Moment	<<Indicate the moment at which data analysis should be performed. The moment of analysis should be an activity of the project process (to analyze data in the project context) or data of an organizational process (to analyze data in the organizational context).>>
Base Measure 1	Number of Fixed Defects
Mnemonic	NFD
Description	Measure which quantifies the number of defects fixed in a product.
Measurable Entity	Defects Fixing Sub-process
Measurable Property	Performance
Scale Value	Positive real numbers.
Unit of measurement	-
Measurement Procedure	Obtain the number of detected defects that were fixed.
(...)	(...)
Base Measure 2	Number of Defects
Mnemonic	ND
Description	Measure which quantifies the number of defects in a product.
Measurable Entity	Software
Measurable Property	Product quality
Scale Value	Positive real numbers.
Unit of measurement	-
Measurement Procedure	Obtain the number of defects in the product.
(...)	(...)
Related Patterns: Product Defect Density	

Table A1: Description of the Defect Fixing Effectiveness Pattern

Appendix B

Defect Fixing Effectiveness	
Name: Defect Fixing Effectiveness Process/Sub-process: Coding / Defect Fixing Goal: Improve defect fixing effectiveness. Information Needs: What is the defect fixing effectiveness? Measures: Defect Fixing Effectiveness, Number of Fixed Defects, Number of Defects. Operational Definition of Measures:	
Derived Measure	Defect Fixing Effectiveness
Mnemonic	DFE
Description	Measure used to quantify the defect fixing effectiveness, which is given by the ratio between the number of fixed defects and the number of detected defects.
Measurable Entity	Defect Fixing Sub-process
Measurable Property	Effectiveness
Scale Value	Positive real numbers accurate to two decimal places.
Measurement Unit	-
Formula	$DFE = (NFD / ND)$
Measurement Procedure	Calculate the defect fixing effectiveness using the formula for calculating the measure, considering the same product (or portion of product) for both measures in the formula.
Measurement Periodicity	Weekly
Measurement Responsible	Software Architect responsible for prioritizing bug fixing activities.
Measurement Moment	Data collection must occur every Monday, considering the data available the week before. Data collection must comprise all test executions performed in each project the week before. The measurement data of each project must be available before the project's follow-up meeting, usually held every Monday afternoon. The measurement data of all projects must be available before the organizational measurement analysis meeting held on the first Wednesday of the month.
Measurement Analysis Procedure	<p>For process behavior analysis (organizational context):</p> <ul style="list-style-type: none"> - Represent in control chart values collected for the measure in several projects. - Obtain the process control limits and analyze the process behavior: <ul style="list-style-type: none"> (i) If the values pass the stability tests, the process is deemed 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 more than 2σ from the central limit; Test 3: there are at least four out of five successive points at the same side and 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 the stability tests, the process is unstable. It is necessary to investigate the special causes, identify corrective actions and execute them. <p>Note: If there is a baseline already established for the process, it is necessary to verify if a new baseline needs to be established. A new baseline can be established at each three months.</p> <p>For quantitative project management (project context):</p> <ul style="list-style-type: none"> - Represent in control chart values collected for the measure in the project. - Analyze the process behavior considering the organizational behavior expected for it (i.e., using the process baseline as reference). <ul style="list-style-type: none"> (i) If the values pass the stability tests considering the process baseline as a reference, then the process behaves according to the behavior expected for it in

	the organization. (ii) If the values do not pass the stability tests considering the process baseline as a 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	<i>Data analysis of each project must be performed weekly, every Monday.</i> <i>Data analysis of the performance of all organization processes must be performed every two months.</i>
Analysis Responsible	<i>In the scope of each project, the Software Architect is responsible for prioritizing bug fixing activities.</i> <i>In the scope of the organization, the leader of the Software Process Expert Group is responsible for analyzing the organizational performance of each process suitable to SPC.</i>
Analysis Moment	<i>Data analysis of each project must be performed before the project's follow-up meeting.</i> <i>Data analysis of the performance of all organization processes must be performed during the Software Process Expert Group meeting, held every two months.</i>
Base Measure 1	Number of Fixed Defects
Mnemonic	NFD
Description	Measure which quantifies the number of defects fixed in a product.
Measurable Entity	Defects Fixing Sub-process
Measurable Property	Performance
Scale Value	Positive real numbers.
Unit of measurement	-
Measurement Procedure	Obtain the number of detected defects that were fixed.
(...)	(...)
Base Measure 2	Number of Defects
Mnemonic	ND
Description	Measure which quantifies the number of defects in a product.
Measurable Entity	Software
Measurable Property	Product quality
Scale Value	Positive real numbers.
Unit of measurement	-
Measurement Procedure	Obtain the number of defects in the product.
(...)	(...)
Related Patterns: Product Defect Density	

Table B1: Description of the Defect Fixing Effectiveness Pattern filled out by an SPC consultant for a specific organization (information in italics was filled out by the consultant when applying the pattern to create the measurement plan for the organization)