A Levels-based Approach for Defining Software Measurement Architectures

Ciro Xavier Maretto[1], Monalessa Perini Barcellos[1]

[1] Ontology and Conceptual Modeling Research Group (NEMO), Department of Computer Science, Federal University of Espirito Santo, Vitória, Brazil
{ciro.maretto, monalessa}@inf.ufes.br

Abstract. During the execution of software projects, it is necessary to collect, store and analyze data to support project and organizational decisions. Software measurement is a fundamental practice for project management and process improvement. It is present in the main models and standards that address software process improvement, such as ISO/IEC 12207, CMMI and MR MPS.BR. In order to effectively perform software measurement, it is necessary an infrastructure to support data collection, storage and analysis. This infrastructure can be defined by means of an architecture, which describes the components necessary to support software measurement. In this paper we present the main results obtained from a systematic mapping study that investigated software measurement architectures and an approach proposed aiming to help organizations define software measurement architectures.

Keywords: Software Measurement, Measurement Architecture, Measurement Repository, Reference Architecture.

1 Introduction

Software measurement involves defining measures, collecting data for these measures and analyzing data aiming to support decision making [1]. Throughout projects, data are collected for the measures and should be stored in a measurement repository in order to be used in project management and process improvement [2].

There are several standards devoted specifically to software measurement, such as ISO/IEC 15939 [3] and PSM (Practical Software Measurement) [1]. Besides, there are several standards and maturity models, such as ISO/IEC 12207 [4], CMMI (Capability Maturity Model Integration) [2] and MR MPS.BR (Reference Model for Process Improvement of Brazilian Software) [5], that address software process improvement and include measurement as an essential process for organizations to achieve maturity in software development.

In maturity models that address software processes improvement in maturity levels, such as CMMI [2] and MR MPS.BR [5], measurement starts at initial levels (CMMI level 2 and MR MPS.BR level F) and evolves as the maturity level increases. At high maturity levels (CMMI levels 4 and 5 and MR MPS.BR levels A and B) statistical process control (SPC) must be carried out and it requires extra attention to some measurement aspects, such as data collection and storage. In other words, as the maturity levels increases the measurement needs change.

Although standards and maturity models are very important to help organizations by indicating what should be done to implement software measurement, due to the nature of measurement activities, supporting tools are also necessary to successfully implement software measurement.

In an organization, technological solutions (e.g., tools and technologies) are used to support process executions. The selection of these solutions should be aligned to the organizational needs and goals. Technological solutions can support the software measurement process and can be described with a certain level of abstraction by means of architectures.

According to Zachman [6], an architecture can be understood as a logical structure in which the components are organized and integrated. In the software measurement context, an architecture should consider aspects related to the data collection, storage and analysis. In a measurement architecture, one of the main components is the measurement repository. According to Bernstein [7], a repository can be defined as a database sharing information about engineering artifacts. In a measurement architecture, the measurement repository stores measurement data (which are not restricted to data collected for the measures) and acts as a data provider to the analysis. Due to its importance in a software measurement architecture, the measurement repository is sometimes seen as the measurement architecture itself.

It is not easy to define a measurement architecture capable of meeting the needs according to the organization maturity level. Usually, organizations start recording measurement data in spreadsheets or in some systems with little or no integration among them [8]. At the initial maturity levels, spreadsheets seem to be enough, but as the organization's maturity level increases, the problems of using spreadsheets become more expressive. Most times, in order to achieve high maturity, organizations need to discard data stored in spreadsheets, develop a measurement repository by using appropriate technologies (e. g., database management systems), and restart data collection and storage. Thus, a good practice is to define an architecture that supports software measurement and can be used from the beginning of a measurement program until the high maturity levels (or can be extended to that) [9].

Aiming to identify and analyze proposals for software measurement architectures recorded in the literature, we carried out a systematic mapping study. As a result, 8 proposals were identified. Only 2 of them address high maturity measurement needs, which include statistical process control. By analyzing the measurement architectures identified during the study, we noticed that although they support the measurement process, they do not guide organizations on how to define their own measurement architectures. The proposals usually address specific solutions developed to a particular context. Besides, they do not usually address measurement at high maturity levels, which includes SPC implementation. Based on this perception, we decided to develop a measurement architecture that could be used as a basis for organizations to define their own measurement architectures. For this, we proposed a levels-based approach in which the third level is a reference architecture, i.e., an architecture defined aiming at reuse.

Following this introduction, in Section 2, we briefly present software measurement and statistical process control. In Section 3, we describe the systematic mapping and its main results. In Section 4, an overview of the proposed approach is presented. In Section 5 we talk about the use of the approach. Finally, in Section 6, we make some final considerations.

2 Software Measurement and Statistical Process Control

Software measurement is a primary support process for managing projects. It is also a key discipline in evaluating the quality of software products and the performance and capability of organizational software processes. The software measurement process includes the following activities: planning the measurement process, execution of the measurement process, and measurement evaluation [3].

Initially, for performing software measurement, an organization must plan it. Based on its goals, the organization has to define which entities (processes, products and so on) and which of their properties (size, cost, time, etc.) are to be measured. The organization has also to define which measures are to be used to quantify those properties. For each measure, an operational definition should be specified, indicating, among others, how the measure must be collected and analyzed. Once planned, measurement can start. Measurement execution involves collecting data for the defined measures, according to their operational definitions. Once data are collected, they should be analyzed. The data analysis provides information to decision making and supports identifying appropriate actions. Finally, the measurement process and its products should be evaluated aiming to identify potential improvements [10].

Depending on the organization's maturity level, software measurement is performed in different ways. At the initial maturity levels, such as the levels 2 and 3 of CMMI, the focus is on developing and sustaining a measurement capability that is used to support project management information needs. At maturity levels, such as CMMI levels 4 and 5, measurement is performed for the purpose of statistical process control (SPC), in order to understand the process behavior and to support software process improvement efforts [11]. SPC uses a set of statistical techniques to determine if a process is under control, considering the statistical point of view. A process is under control if its behavior is stable, i.e., if their variations are within the expected limits, calculated from historical data. The behavior of a process is described by data collected for performance measures defined to this process [12].

A process under control is a stable process and, as such, has repeatable behavior. So, it is possible to predict its performance in future executions and, thus, to prepare achievable plans and continuously improve the process. On the other hand, a process that varies beyond the expected limits is an unstable process and the causes of these variations (said special causes) must be investigated and addressed by improvement actions in order to stabilize the process. Once the processes are stable, their levels of variation can be established and sustained, being possible to predict their results. Thus, it is also possible to identify the processes that are capable of achieving the established goals and the processes that are failing in meeting the goals. In this case, actions to change the process and make it capable should be carried out [12].

Statistical process control requires some changes in the traditional measurement, specially related to operational definition of measures, data collection frequency, measurement granularity, data homogeneity and data grouping to analysis [13].

3 Software Measurement Architectures: A Systematic Mapping

According to Kitchenham and Charters [14], a systematic mapping (also known as exploratory study) makes a broad study in a topic of a specific theme and aims to identify available evidence about that topic. In this sense, we carried out a systematic mapping aiming to identify evidences regarding measurement architectures proposals recorded in the literature. In order to perform the systematic mapping, we used the process proposed in [15], which was defined based on [14]. It consists of the following three activities:

i) Develop Research Protocol: In this step the researcher prospects the topic of interest, defines the context to be considered in the study, and describes the object of analysis. Next, he/she defines the research protocol that will be used as a guideline to perform the research. The protocol must contain all the necessary information for a researcher to perform the research (research questions, source selection criteria, publication selection criteria, procedures for storing and analyzing the results, and so on). The protocol must be tested in order to verify its feasibility, i.e., if the results obtained are satisfactory and if the protocol execution is viable in terms of time and effort. The test results allow for improving the protocol when necessary. If the protocol is viable, an expert must evaluate it and, once approved, the protocol can be used to guide the research.

ii) Perform Research: In this step the researcher performs the research according to the research protocol. Publications are selected, and data are extracted, stored, and quantitatively and qualitatively analyzed.

iii) Provide Results: In this step the research results produced during the execution of the systematic review process should be packaged and published in a conference, journal, technical report or other publication vehicle.

3.1 Research Protocol

The research protocol used in the study contains the following information: objective, research questions, sources selection criteria, publications selection criteria, data storage and data analysis procedures, and protocol test procedure.

A. Objective

Analyzing the literature in the context of software measurement architectures, with the main purpose of identifying and analyzing:

(i) Proposals for software measurement architectures;
(ii) The proposals characteristics;
(iii) If the proposals are capable of supporting the statistical process control.

B. Research Questions

Q1. Which proposals for software measurement architecture are recorded in the literature?
Q2. What are the proposals characteristics?
Q3. Which proposals include support to statistical process control?

In Q3, support to statistical process control consists in supporting: data collection, storage, representation (by using control charts), and process behavior analysis.

C. Sources

The publications sources must be digital libraries and:

(i) Have a search mechanism that allows using logical expressions and searching in different parts of the publications;
(ii) Be available in the CAPES (Coordination for the Improvement of Higher Education Personnel) Journals Portal[1];
(iii) Include publications in the Physical Science area, in particular Computer Science.

D. Procedure for Publications Selection

The object of analysis are papers published in conferences and journals. Publications selection must be done in three steps:

1st step – Preliminary selection and cataloging: the preliminary selection must be done by applying the following criteria using the digital library search mechanism:

Scope: title, abstract and keywords.
Language: English.
Search String: ("measurement framework" OR "measurement database" OR "measurement repository" OR "measurement architecture" OR "metrics repository" OR "metrics database") AND "software".
Period: from 1990.
Area: Computer Science.

In order to establish the search string, we performed some tests using different terms, logical connectors, and combinations between them, aiming to obtain a search string able to return relevant publications to the study and a viable quantity to be analyzed.

During the informal literature review that preceded the study, we found some relevant publications addressing measurement repositories. In fact, although these publications use the term measurement repository, in the context of the study they address measurement architecture. Thereby, we decided to include in the search string terms related to repositories.

Also during the informal review we identified two relevant publications ([16] and [17]) that we used as control publications to evaluate the search strings (the string must be able to return the control publications). The tests to obtain the search string were carried out using the digital libraries Scopus (*www.scopus.com*) e IEEE (*ieeexplore.ieee.org*). Scopus was selected because during preliminary tests it returned the largest number of publications. IEEE, in turn, was selected because the control publication [17] was only available in IEEE.

Considering the tests results we decided to select a comprehensive string and to restrict the publications selection in the later steps, since more restrictive strings excluded one or both the control publications. The selected string returned many publications that deal with measurement repositories not related to software measurement, but to scientific experiments from other computer areas. However, when we tried to restrict the publications by using the term "software measurement" instead of "software", the search results were very restricted and one of the control publications was not returned. Although the selected string is comprehensive, it was

[1] CAPES Journals Portal (www.periodicos.capes.gov.br/) is sponsored by Brazilian government and offers access to the publications of many international and national sources, covering all knowledge areas.

the one which provided better results in terms of number and relevance of selected publications.

We decided to apply the search string to the title, abstract and keywords, because some tests carried out by applying the string to the full text resulted in a large number of publications, being many of them useless. On the other hand, when restricting the string only to the title, useful publications were eliminated.

2nd Step – Selection of Relevant Publications – 1st filter: selecting publications by applying a search string does not ensure that all selected publications are relevant, because such selection is restricted to syntactic aspects. Thus, the abstract of the publications selected in the 1st step must be analyzed. Publications that do not satisfy one of (or both) the following criteria must be eliminated:

SC1: The publication addresses collection, storage, analysis or recovering of measurement data.

SC2: The publication addresses some kind of software measurement architecture or measurement repository.

We refer explicitly to measurement repositories in SC2 (and in SC3 presented forward), because, as it was said before, we noticed that some publications address measurement repository proposals as an architecture, according to the concept of architecture used in the study (see Introduction).

To avoid premature exclusions of publications, in case of doubt, the publication should not be eliminated. Besides, publications without an abstract should not be eliminated.

3rd Step - Selection of Relevant Publications – 2nd filter: the selection of the publications in the 2nd step considers only the abstract. Consequently, it is possible that some selected publications do not contain relevant information. Therefore, the full text of the publications selected in the 2nd step must be read. Publications that do not satisfy one of (or both) the following criteria must be eliminated:

SC3: The publication describes software measurement architectures or measurement repositories.

SC4: The full text is accessible.

E. Data Storage Procedure

Each publication selected in the 1st step must be catalogued with the following data: title, author(s), year, reference data, source (digital library), and a summary. Each catalogued publication must be examined and submitted to the next two steps. The publications eliminated on the 2nd step must be identified as "E2: SC[number of the criteria not satisfied]". Similarly, publications eliminated on the 3rd step must be identified as "E3: SC[number of the criteria not satisfied]".

F. Data Extraction and Analysis Procedure

For each publication selected in the 3rd step, the following information must be extracted:

- *(i)* Proposal identification. The identification is the proposal name as cited in the publication. If the proposal has no name, it must be identified as "Proposal XYZ", where XYZ are the initial letters of the proposal authors names;
- *(ii)* A brief description of the proposal;
- *(iii)* Proposal characteristics, organized according to the following categories: Technology, Architecture, Collection, Storage, and Analysis;

(iv) Indication if the proposal supports statistical process control.

Regarding (iv), it must be recorded "Yes" to proposals whose publications make explicit the support to SPC. It must be recorded "Probably Applicable" to proposals that do not make explicit the support to SPC, but apparently they are able to support it. It must be recorded "No" to proposals that do not mention support to SPC and it is not possible to conclude that they support it.

After data are extracted from publications, quantitative and qualitative analysis must be done with the main purpose of discussing the findings in relation to the research questions.

G. Test Protocol Procedure

The research protocol must be tested using a reduced number of sources in order to verify its viability. The protocol is viable if the procedures are performed as described, if it is possible to answer the research questions and if the time and effort necessary are viable.

3.2 The Results

The protocol presented in the previous section was evaluated by an expert. Then, it was tested using the digital library IEEE. The protocol was considered viable and it was executed one more time using the digital library Scopus. In this section some results obtained from these two executions are presented. Publications selected in both digital libraries were counted only once. In total, 148 publications were selected in the 1st step, 22 in 2nd the step and 12 in 3rd step.

It is possible to notice a large decrease in the number of publications in the 2nd step. In fact, this result was expected, since we decided to use a comprehensive search string, as argued in the previous section.

It is worth mentioning that the focus of the study is on measurement architectures and, for this reason, publications which described lessons learned and case studies that mention the use of measurement architecture (not describing the architecture) were excluded during the selection criteria application (these publications do not satisfy SC3).

Analyzing the publications by year, from 148 publications selected by the search string (1st step), 25 (17%) are dating from 1990 to 2000, and 123 (83%) are dating from 2001 to 2011. From 12 publications selected in 3rd step, a quarter is dating from 2009 on. Besides, although we have limited the selection to publications from 1990 on, the oldest publications are from 1999 and 2000.

From the publications selected in 3rd step, 8 proposals were identified. Table 1 shows a brief description of the proposals and their respective publications. A summary of the proposals characteristics is presented on Table 2. Publications describe their proposals with different levels of detail and with different foci. Consequently, information regarding the characteristics has also heterogeneous levels of detail. For instance, some proposals describe in details characteristics of the adopted architecture, while others only mention the general model in which the architecture is based on, and others nothing said about their architecture. In Table 2, when information regarding a category is not shown, it means that it was not possible to obtain information about it by reading the publications.

Table 1. Proposals found.

Proposal	Description	Ref
P01 - Generic Measurement Framework Based on MDA	Software measurement framework to support the software measurement entities through metamodels and transformations. For example, given a model of an ER (Entities and Relationships) diagram, measures such as quantity of tables and relationships can be automatically calculated using the framework. For this, the framework uses a domain model and a measurement model, which says which entities will be measured and what methods will be used. These models go through transformation processing QVT (Query View Transformation), which generates the measurements.	[16, 18, 19, 20]
P02 - WebEv (Web for the Evaluation)	System that uses a measurement framework based on GQM (Goal Question Metric) [21] to business process evaluation and gives support to data collection, storage and analysis. It was defined in terms of measures, mechanisms for data collection and guides to use the collected data.	[22, 23]
P03 - NSDIR (National Software Data and Information Repository)	It is an organizational benchmarking repository to software projects from the U.S Air Force. It was operational from 1994 to 1998. Although its use has ended up in 1998, the industry and academy efforts continued through CEBASE (Center for Empirically-Based Software Engineering).	[24]
P04 - MRS (Measurement Repository System)	It is a measurement repository used by a group of telecommunication companies. One of the main purposes was the supply and products evaluation through reporting generation which compiled data from all participating companies. The repository has as the main concern the safety and privacy of the information.	[25]
P05 - MMR Tool	Proposal of a generic and flexible measurement repository for data collection, storage, analysis, and publication. It was projected to give support to all CMMI levels and it was applied in Ericson Research Canada.	[17]
P06 - SPDW+ (Software Development Process Performance Data Warehousing)	It presents the data warehousing architecture SPDW+ as a repository solution centralized in measurements, automatic collection and analysis mechanisms. The SPDW+ is an improvement of the SPDW, which was operational for 3 years in HP Brazil. It was developed aiming at supporting process improvements in mature organizations.	[26]
P07 - A Universal Metrics Repository	It proposes a structure to a flexible measurement repository, able to adapt itself to different lifetime models, methodologies, and software developments process. The proposal uses transformational view concepts of software development, which considers that the software development process is a series of artifacts transformation.	[27]
P08 – Proposal PAU	It presents a generic framework that incorporates database, a formal set of software tests and evaluation measures, as well as an advanced set of analytical techniques for information and knowledge extraction. The approach proposes using this framework and its techniques to extract detailed information and knowledge from the software measurement repositories.	[28]

Table 2. Overview of the general characteristics of the identified proposals.

Proposal	Features					
	Technology	**Architecture**	**Collection**	**Storage**	**Analysis**	**SPC Support**
P01	Use of DSL(Domain-Specific Language) and tools based on Eclipse platform	Based on MDA (Model Driven Architecture)	Automatic (through models transformation)	XML file		No
P02	Use of Java (Java JDBC and Java Servlet API)		Semi-automatic (via web form)	Database	Quantitative analysis resources	Probably Applicable
P03	Use of Sun Solaris Unix, Oracle and client in Visual Basic with ODBC (Open Database Connectivity)	Client-Server (central repository which stores data collected by client software)	Manual and Semi-automatic (through physical or electronics forms)	Database	Analysis tools in a benchmark style	No
P04		Client-Server (central repository which stores data collected by client software)	Semi-automatic (through electronic form)	Database	Generation of quarterly reports	No
P05	Use of Technologies and Microsoft tools (SQL 2000 Server, Analysis Services Enterprise Edition, Internet Information Server, Intranet Share Portal Server, ASP)	Based on data warehouse environment	Semi-automatic. Intend to use ETL (Extraction, Transformation and Loading) to collect voluminous and periodic data.	Data warehouse. The database model is generic for data flexibility	SQL (Structured Query Language) and OLAP (On-line Analytical Processing) cubes. Data is presented via web portal. It is possible to export data to statistical tools.	Yes

Table 2. Overview of the general characteristics of the identified proposals (cont.).

Proposal	Features					
	Technology	**Architecture**	**Collection**	**Storage**	**Analysis**	**SPC Support**
P06	Use of Microsoft technologies and tools. (SQL Server 2005, BI Studio, Visual Studio 2005, SQL Server Integration Services and IIS 6.0)	Oriented to services (SOA – Service Oriented Architecture) and based on data warehouse environment with four components	Semi-automatic and automatic, by using ETL.	Data warehouse	Use of BI (Business Intelligence) tools with web interface, including OLAP and dashboard.	Yes
P07	Use of MySQL (only the repository is implemented)			Database. The database model is generic for data flexibility.		No
P08			Semi-automatic	Database	Use of statistical techniques and others, such as: multiresolution analysis, classification trees, neural networks, and influence diagrams.	No

3.3 Discussions

In general, the proposals identified are very different. Unfortunately, based only on information recorded in the publications, many times it is not possible to compare the proposals in a substantial way. Regarding the proposals characteristics, some considerations are presented below:

Technology

The technologies used in the proposals are diverse, varying from free software to proprietary technologies. This can be a reflex of the variety of technological solutions available in the market.

Architecture

All the proposals, except the Generic Measurement Framework based on MDA [16, 18, 19, 20], include in their architecture a central repository to store and retrieve data, using a client-server architecture. The proposals MRS [25] and NSDIR [24] have specific client programs for communication with the server. WebEv [22, 23], MMR Tool [17], and SPDW+ [26], in turn, use web resources. The proposals SPDW+ [26] e MMR Tool [17] have architectures based on data warehouse environment, including a component for data collection (ETL), a component to storage (data warehouse), and a component for analysis with analytical capabilities (OLAP). The SPDW+ [26] includes a fourth component responsible for the data integration. It acts as a temporary repository for standardization of the collected data.

The Generic Measurement Framework based on MDA [16, 18, 19, 20] is a conceptual architecture and it is an adaptation of MDA. It is divided in levels, ranging from MOF (Meta-Object Facility) to measurement data, also including a measurement meta-model based on a software measurement ontology.

Collection

In Table 2 it is possible to notice three types of collection: manual, semi-automatic and automatic. Manual collection refers to the use of physical forms in order for people to record data collected for the measures. Semi-automatic collection refers to the use of computational supporting (for instance, electronic forms and information systems) to record data collected for the measures. In the semi-automatic collection, although there is computational supporting, data are supplied by people. Automatic collection refers to the use of computational tools and mechanisms that obtain data for the measures without human intervention.

Most of the proposals use semi-automatic collection. The publications which describe the proposals MMR Tool [17] e MRS [25] mention the intention of using automatic collection mechanisms, but these mechanisms are not described. Only two proposals implemented the automatic collection: Generic Measurement Based on MDA, [16, 18, 19, 20], by means of models transformation; and SPDW+ [26], with a ETL component. It is important to emphasize that these proposals deal with very specific types of measures (for instance, quantity of tables and relationships in a certain data model, and number of errors in a portion of source code), which are more favorable for automatic collection. Therefore, proposals that deal with measures whose automatic collection is more difficult or not possible adopt semi-automatic collection. This can be seen as a sign of the difficulty and, in some cases impossibility, of adopting automatic collection. Only one proposal (NSDIR [24]) uses

manual collection and the data collected in physical forms are recorded in electronic forms a posteriori.

Storage

The proposals use three different solutions to data storage: relational database (WebEv [22, 23]), XML (eXtensible Markup Language), files (Generic Measurement Framework Based on MDA [16, 18, 19, 20]), and solutions based on database. Although most of the proposals adopt solutions based on databases, we noticed that each proposal support the storage of different measurement data. We believe that this occurs mainly because the repository structure (the database "model") is defined based on the specification of which entities and elements are to be measured and what information needs are expected to be satisfied by the measurement data.

We also noticed that some proposals provide flexibility regarding which measurement data can be stored. For instance, MMR Tool [17] uses a measurement domain meta-level structure as a data model, with the purpose of allowing adaptation to different measurement contexts. On the other hand, the Universal Metrics Repository [27] is said to be itself a flexible database that aims to store any data from any measures related to different entities.

Finally, we observed that the proposals that include support to statistical process control (SPDW+ [26] e MMR Tool [17]) adopt solutions based on data warehouse.

Analysis

Most of the proposals include mechanisms for data analysis and presentation. Some proposals, such as SPDW+ [26] and PAU [28], have more complex mechanisms and tools. The analysis can be purely qualitative, as in WebEv [22, 23], or have a benchmark type, as in NSDIR [24] and MRS [25], in which general data of products and projects can be analyzed to support identification of best practices. The proposals that support statistical processes control (SPDW+ [26] and MMR Tool [17]) adopt more sophisticated mechanisms to data analysis (both of them use OLAP tools).

Support to SPC

Most proposals do not provide support to statistical process control. For instance, the proposal NSDIR [24] includes a repository which stores general data regarding products and projects with the main purpose of using them as benchmarking. Data concerning the process definition or its executions are not stored, what does not allow for carrying out SPC.

Only two proposals (SPDW+ [26] and MMRTool [17]) include support to SPC. Both of them were developed in the context of large companies aiming at high maturity levels. These two proposals use Microsoft technologies and solutions based on data warehouse.

4 LASMA: A Levels-based Approach for Defining Software Measurement Architectures

As presented in the previous section, there are some proposals for software measurement architectures recorded in the literature. After carrying out the study, we analyzed its results and we concluded that although the proposals found support the

measurement process, they do not guide organizations on how to define their own measurement architectures. In general, the proposals are specific solutions developed to a particular context and most of them do not address measurement at high maturity levels, where SPC practices are needed. Thus, aiming to help organizations define software measurement architectures capable of supporting traditional and high maturity measurement, we proposed a levels-based approach that provides knowledge regarding what a measurement architecture should address in order to properly support software measurement.

LASMA was inspired by the Model Driven Engineering (MDE) [29] and the Model Driven Architecture (MDA) approaches [30]. MDE advocates the use of models with different levels of abstraction and transformations of models from a level to another. MDA, in turn, uses a Platform Independent Model (PIM) as a basis to generate Platform Specific Models (PSM), which are used to implement systems.

Following the MDE principles, LASMA comprises levels at which there are models with different levels of abstraction (from the more to the less abstract). In addition to that, following the MDA principles, LASMA distinguishes platform independent models from platform specific models.

It is important to point out that, although LASMA has been inspired by MDE and MDA, it is not a MDE or MDA approach. Thus, transformations to lead a model into another are not addressed in LASMA.

An overview of LASMA is shown in Figure 1. LASMA has five levels. The higher the level in the figure, the higher is the level of abstraction. The arrows indicate that a model from a level is used as a basis to define a model from the next one. After Figure 1, the LASMA levels are described.

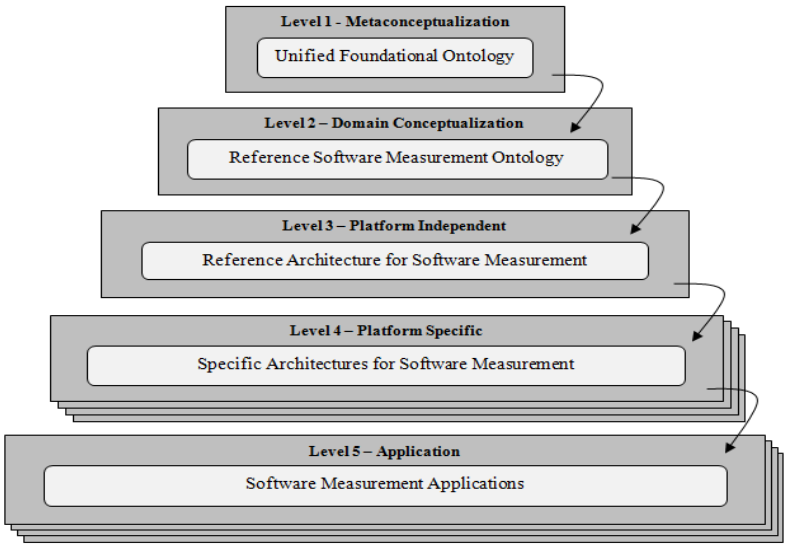


Figure 1. LASMA overview.

4.1 Level 1: Metaconceptualization

The first level concerns conceptual models that describe real-world objects that are domain independent. At this level lies the *Unified Foundational Ontology* (UFO) [31], which is a foundational ontology that has been developed on the basis of a number of theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics and Cognitive Psychology.

UFO is composed by three main parts. UFO-A is an ontology of endurants. A fundamental distinction in UFO-A is between Particulars (Individuals) and Universals (Types). Particulars are entities that exist in reality possessing a unique identity, while Universals are patterns of features, which can be realized in a number of different particulars [31]. UFO-B is an ontology of perdurants (events). UFO-C is an ontology of social entities (both endurants and perdurants) built on the top of UFO-A and UFO-B. One of its main distinctions is between agents and objects. Agents are capable of performing actions with some intention, while objects only participate in events [32].

UFO has been used as a basis to build and reengineer several domain ontologies [33, 34, 35, 36]. Its function in LASMA is to provide the generic conceptualization used as a basis to define the conceptualization of the software measurement domain. A complete description of UFO falls outside the scope of this paper. In the sequel, aiming to present some examples of UFO concepts, we give a brief explanation of some UFO concepts shown in Figure 2. Details regarding UFO concepts can be found in [31, 33].

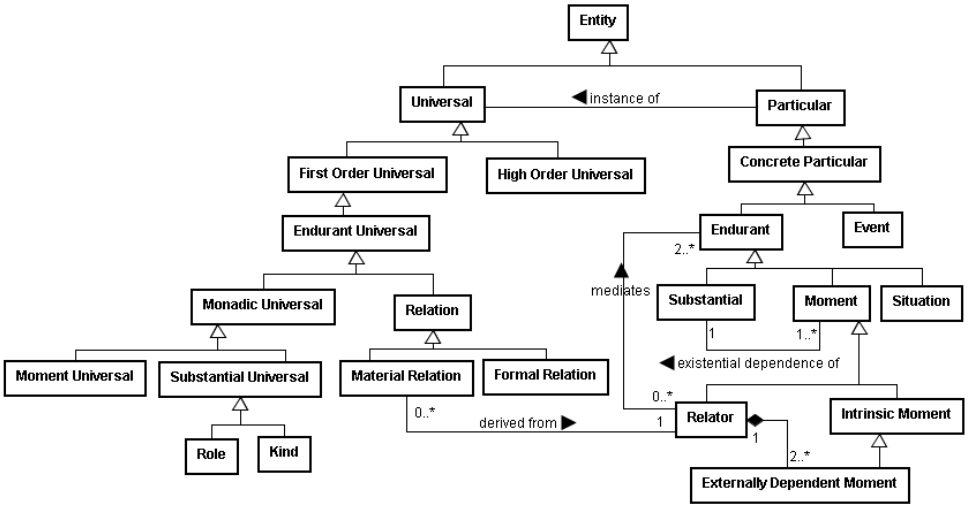


Figure 2. UFO fragment.

An *entity* is something perceivable or conceivable. It is the most general concept in UFO. *Universals* are patterns of features that can be realized in a number of different entities (e.g., Person). *Particulars* are entities that exist in reality, possessing a unique identity (e.g., the person Mary). Universals can be *first order universals*, i.e., universals whose instances are particulars (e.g., Person), or *high order universals*, which are universals whose instances are also universals (e.g., Specie, whose instances could be Mammal, Reptile, and Bird, among others).

Endurant universals are universals that persist in time maintaining their identity. Endurant universals can be monadic universals or relations. *Monadic universals*, in turn, can be further categorized into substantial universals and moment universals (properties). A *moment* is an endurant that is existentially dependent of another endurant, in the way, for example, that the color of an apple depends on the apple in order to exist. Existential dependence can also be used to differentiate intrinsic and relational moments. *Intrinsic moments* are dependent on one single endurant (e.g., color). *Relators* depend on a plurality of endurants (e.g., an employment, a medical treatment, a marriage) and, for this reason, provide the material connection between them. In other words, we can say that they are the foundation for *material relations* such as "working at". Thus, material relations require relators in order to be established. *Formal Relations*, in contrast, hold directly between individuals (e.g., the part-of relation).

Regarding substantial universals, while persisting in time, substantial particulars can instantiate several substantial universals. Some of these types, a substantial instantiates necessarily (i.e., in every possible situation) and define what the substantial is. These are the types named *kind* (e.g., Person). There are, however, types that a substantial instantiates in some circumstances, but not in others, such as is the case of *roles*. A *role* is a type instantiated in a given context, such as the context of a given event participation or a given relation (e.g., Student).

4.2 Level 2: Domain Conceptualization

The second level of LASMA refers to models that represent the domain conceptualization. At this level lies the *Reference Software Measurement Ontology* (RSMO) [10, 11, 34, 36, 37]. RSMO is a domain reference ontology, i.e., a domain ontology that is constructed with the sole objective of making the best possible description of the domain in reality, with regard to a certain level of granularity and viewpoint. A domain reference ontology is a special kind of conceptual model representing a model of consensus within a community. It is a solution-independent specification with the aim of making a clear and precise description of domain entities for the purposes of communication, learning and problem-solving [38].

According to Guarino [39], aiming fidelity to reality and conceptual clarity, ideally domain ontologies should be built based on foundational ontologies. RSMO was developed based on UFO, the foundational ontology present in the first level of LASMA. Discussions regarding the use of UFO as a basis to develop RSMO can be found in [10, 11, 34, 37].

The function of RSMO in LASMA is to provide the domain conceptualization necessary to define models from the level 3 (Platform Independent).

RSMO addresses the software measurement domain considering traditional and high maturity aspects. For this, it is composed of six sub-ontologies: the *Measurable Entities & Measures* sub-ontology, which is the core of the RSMO, treating the entities that can be submitted to measurement, their properties that can be measured, and the measures used to measure them; the *Measurement Goals* sub-ontology that deals with the alignment of measurement to organizational goals; the *Operational Definition of Measures* sub-ontology, which addresses the detailed definition of operational aspects of measures, including data collection and analysis; the *Software*

Measurement sub-ontology that refers to the measurement per se, i.e., collecting and storing data for measures; the *Measurement Analysis* sub-ontology, handling the analysis of the collected data for getting information to support decision making; and finally, the *Software Process Behavior* sub-ontology, which refers to applying measurement results in the analysis of the behavior of software processes.

Since the software measurement domain is strongly related to the domains of software processes and organizations, RSMO reuses some concepts from the software process ontology described in [33] and the software organization ontology presented in [34].

Figure 3 shows the RSMO sub-ontologies and the integrated ontologies as UML packages, and their relationships as dependency relationships. In the figure, the dependency relationships indicate that concepts and relations from a sub-ontology/ontology are used by another.

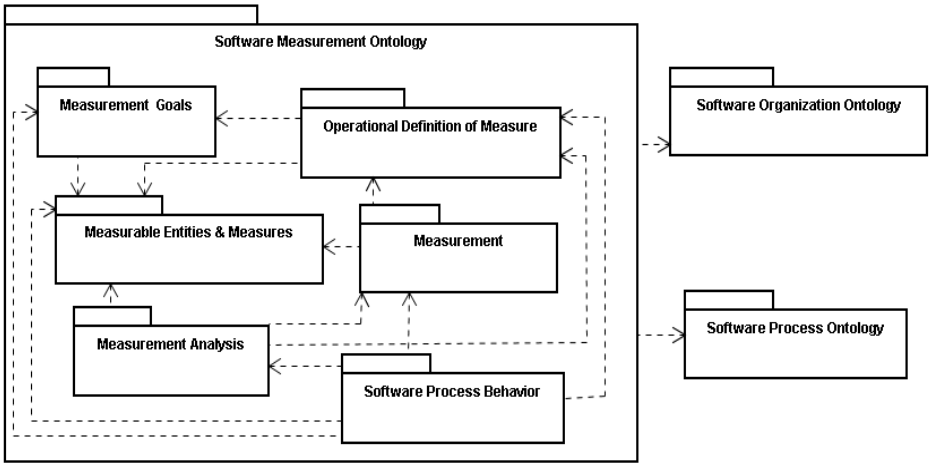


Figure 3. RSMO overview.

RSMO is very extensive and a complete description falls outside the scope of this paper. As an example, we present some of RSMO basic concepts. Figure 4 presents a fragment of the Measurable Entities & Measures sub-ontology. The concepts presented are described below. In the text, the first occurrences of RSMO concepts are shown in **bold** and instances of RSMO concepts are shown underlined.

A **Measurable Entity** is anything that can be measured, such as a process, an artifact, a project and a resource. Measurable entities can be classified according to types (**Measurable Entity Type**). For instance, process is a type of measurable entity.

Measurable Entities are characterized by **Measurable Elements**. A Measurable Element is a property of a Measurable Entity that can be distinguished, and thus measured. Size and productivity are examples of measurable elements. Measurable Elements can be directly (e.g., size) or indirectly (e.g., productivity) measured. **Indirectly Measurable Elements** are measured by means of other measurable elements, said their **sub-elements**. Measurable Entities that are instance of the same Measurable Entity Type are characterized by the same Measurable Elements.

A **Measure** is an instrument that allows associating **Measurable Elements** with **Scale Values** of a **Scale**. For instance, the measure number of requirements can be used to associate a value to the measurable element size that characterizes the measurable entity type project. Thus, a Measure quantifies a Measurable Element and has a Scale composed by Scale Values. Moreover, a Scale is of a **Scale Type** (e.g., interval, ratio).

A Measure can be correlated to other measures, said its **correlated measures**, indicating, for instance, that they have a cause-effect relationship. Finally, Measures can be classified into **Base Measures**, which are functionally independent of other measures (e.g., number of requirements) and used to quantify Directly Measurable Elements, and **Derived Measures** (e.g., requirements changing rate, given by the ratio of the number of changed requirements to the number of requirements), which are defined as a function of other measures and used to quantify Indirectly Measurable Elements.

A **Measure** can be expressed in a **Measure Unit** (e.g., hour). **Derived Measures** are calculated by **Measure Calculation Formulas**, which, in turn, use other measures as **measures for calculation**.

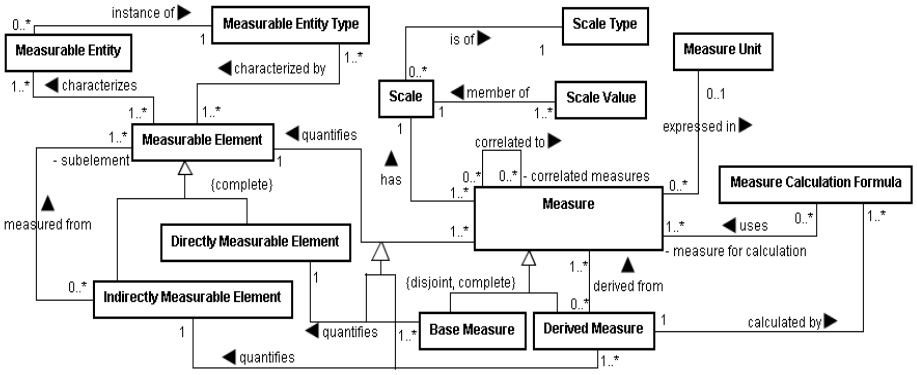


Figure 4. Fragment of the Measurable Entities & Measures sub-ontology.

4.3 Level 3: Platform Independent

The third level of LASMA concerns models that describe a software measurement conceptual architecture, i.e., an architecture that does not take technological aspects into account. In an analogy with the software development process, a model of this level can be compared to models produced during the requirement analysis phase, in which technological aspects are not considered. The purpose of models from this level is to represent the aspects that a technological solution (e.g., tools, integrated systems, etc.) for supporting software measurement should address. The representation is made by means of requirements that should be satisfied by software measurement technological solutions, as well as by conceptual models that describe the structure for data storage that these solutions should be able to provide. At this level lies the *Reference Architecture for Software Measurement* (RASM).

A reference architecture captures the essence of a set of systems. Its purpose is to guide the development of architectures for new systems [40]. Different from the generalist concept of architecture (a logical structure in which components are organized and integrated), reference architectures are conceived mainly aiming at reuse. In this sense, Nakagawa [41] highlights that the use of domain ontologies as a basis to develop reference architectures contributes to understand the domain and identify the requirements to be addressed.

RASM was developed based on the RSMO conceptualization. It is made up of three components, as shown in Figure 5. According to RASM, data related to the organization, projects and their artifacts are captured and stored in a measurement repository. Then, collected data are analyzed and the results are provided to the stakeholders. RASM components are described after Figure 5. Since in this paper the focus is on presenting LASMA as a whole, details regarding the components definitions are not presented.

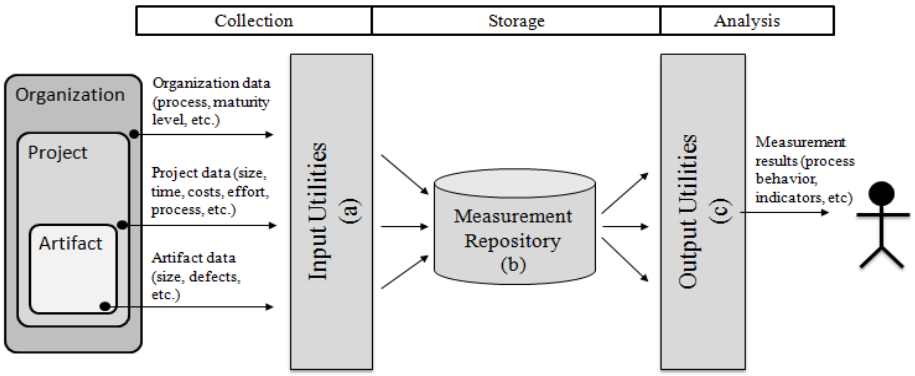


Figure 5. RASM overview.

(a) *Input Utilities*
This component is responsible for capturing measurement data. It is described by functional requirements related to traditional and high maturity software measurement. Each requirement has a detailed description. In a technological solution for supporting software measurement, the input utilities should provide a set of functionalities able to satisfy these requirements. As an example, there is the requirement "It must be possible to record entities to be measured, its types and their elements (properties) that can be measured".

(b) *Measurement Repository*
This component is responsible for storing measurement data. It is described by means of UML (*Unified Modeling Language*) package diagram, class diagrams, detailed descriptions and constraints. Figure 6 shows the package diagram of the measurement repository.

(c) *Output Utilities*
This component is responsible for data analysis and analysis results presentation, aiming to help the stakeholders make decisions. Analogous to the Input Utilities, this component is described by functional requirements.

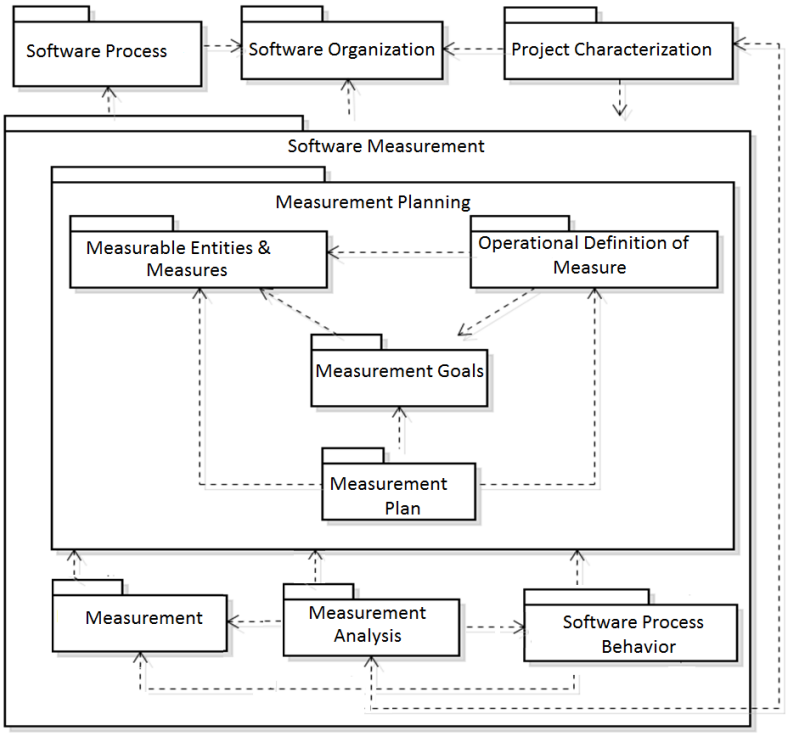


Figure 6. Package diagram of the measurement repository.

The decision on which components would be addressed by RASM took the results from the systematic mapping into account. We noticed that, independently of the used technologies, most of the proposals include functionalities for data capture, storage and analysis. In fact, that was expected, since these are the measurement basic activities. We also noticed that in several proposals data are provided by different sources and captured by different tools. Similarly, in some proposals storage data are used by different tools that support data analysis and provide results. Thus, we decided to define three components and name *utilities* those responsible for data capture and presentation, to make explicit the possibility of implementing them either as one or several tools.

4.4 Level 4: Platform Specific

The fourth level of LASMA refers to models developed on the basis of the platform independent model (RASM) and taking technological issues into account. It is important to point out that in this level it is possible to use only parts of RASM. For instance, if an organization is not interested in high maturity software measurement, the requirements and packages that address high maturity aspects can be disregarded.

In an analogy with the software development process, a model of this level can be compared to models produced during the design phase, in which technological aspects are considered.

In LASMA, the models lied in levels 1, 2 and 3 are predetermined (UFO, RSMO and RASM, respectively). Thus, the users of LASMA use the models from these levels to define platform specific measurement architectures. In fact, users of LASMA use directly RASM to define their own architectures. RASM requirements and measurement repository structure are used as a basis to the definition of platform specific architectures for organizations.

4.5 Level 5: Applications

The last level of LASMA concerns software applications developed based on the specific architectures defined in the previous levels. Although software applications are not models, they were included in a level of LASMA because they can be used as a means to evaluate the architectures defined in levels 2 and 3 [40].

5 Using the Proposed Approach

Aiming to verify if LASMA is practicable, it was used to define a platform specific architecture and a tool. For this, the model defined in the level 3 of LASMA, i.e., the Reference Architecture for Software Measurement (RASM), was used to define a specific architecture.

The input and output utilities defined in RASM were implemented as a tool (M&SPC) and the measurement repository was implemented as a database. Figure 7 presents an overview of the specific architecture defined.

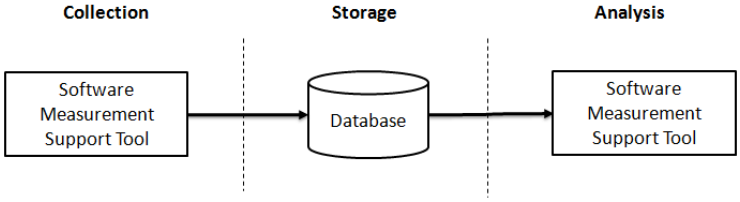


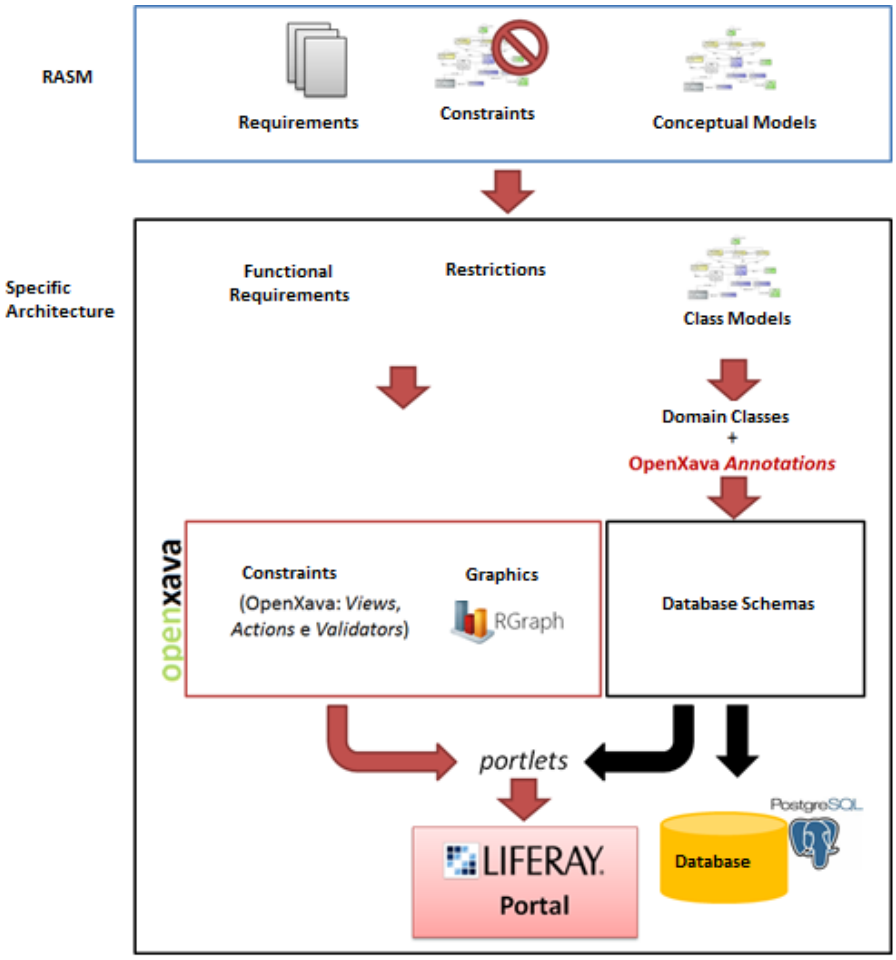
Figure 7. Specific architecture overview.

The specific architecture defined is simple and it is similar to the WebEv proposal [22, 23] found during the systematic mapping, since in that proposal data collection, storage and analysis are supported by a web application.

The technologies used were: OpenXava[2], which is a Java framework with AJAX (Asynchronous Javascript e XML), some Javascript libraries (e.g., RGraph[3]), the LifeRay Portal[4] and Postgres SQL[5]. In order to define the specific architecture, the input and output utilities requirements were used as functional requirements that defined which functionalities should be provided by the tool. The measurement repository conceptual models were used as a basis to define the database schemas. The conceptual models were changed resulting in class models suitable to the technology used. The measurement repository constraints were used to identify

[2] http://openxava.org
[3] http://www.rgraph.net
[4] http://www.liferay.com
[5] http://www.postgresql.org

restrictions that should be addressed by the tool and database. Figure 8 depicts the technologies used in the specific architecture defined.



The specific architecture was used to implement a tool. M&SPC contains a set of functionalities implemented aiming to meet the requirements described in RASM and addressed by the specific architecture defined. Figure 9 shows, as an example, a M&SPC screen used to elaborate the project measurement plan.

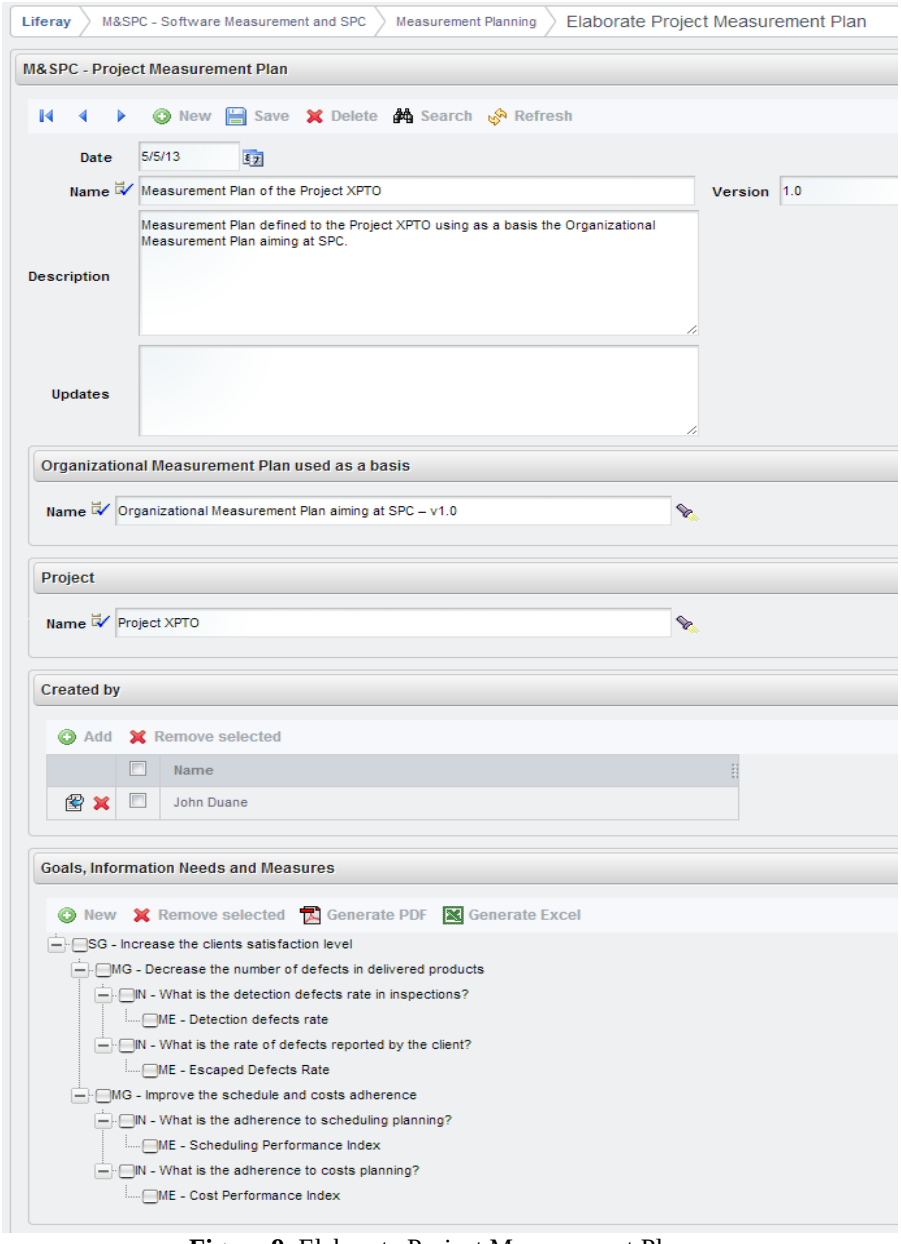


Figure 9. Elaborate Project Measurement Plan screen.

The definition of the specific architecture and the implementation of the M&SPC tool served as a proof of concept of LASMA (particularly of RASM), showing that the proposal is feasible [42]. However, this was only a preliminary result, since it is necessary to evaluate if the proposal works in a real context. After the definition of the specific architecture and tool, an expert in traditional and high maturity software

measurement used the M&SPC tool and made some considerations that helped us to make some adjustments in RASM and, consequently, in the specific architecture and tool. After that, an experimental study was carried out. The participants were computer science MA and PhD students with knowledge and practical experience in software measurement. Details regarding the experimental study fall outside the scope of this paper. The results obtained from the experiment showed that the tool developed using the approach is able to properly support software measurement. According to [40], tools developed in the basis of a reference architecture can be used as a means to evaluate the architecture. However, since the experimental study was carried out in the academic context, the results are not conclusive and new experiments in the industrial context are being planned.

6 Final Considerations

In this paper we presented the main results from a systematic mapping carried out aiming to investigate proposals for software measurement architectures recorded in the literature. Since the results of the study showed us that the proposals recorded in the literature do not guide organizations on the definition of their own measurement architectures, we defined an approach (LASMA) with that purpose. An overview of LASMA was presented in this paper.

Regarding the systematic mapping, altogether, 148 selected publications from the digital libraries IEEE and Scopus were analyzed and 8 software measurement architectures proposals were found. The proposals have some similarities (for instance, the use of solutions based on database for data storage by most of the proposals), but they also present many differences (for example, the technologies adopted).

As limitations of the study, we highlight the use of only two digital libraries as sources of publications and the unavailability of the full text of some publications. Concerning the use of only two sources, although it is a limitation, initial tests showed that the selected publications from some other libraries were similar than the selected publications from the digital libraries used until this moment. Concerning publications whose full text was not available, we contacted the authors and some of them made their publications available. However, four publications were eliminated due to the unavailability of the full text.

The systematic mapping results showed that although there are some proposals for measurement architectures, they do not guide organizations on how to define measurement architectures. Thus, we defined LASMA, an approach made up of five levels. Each level contains models with different levels of abstraction. A model from a level is used as a basis to develop the model from the next one. At the top 3 levels lie UFO (Unified Foundational Ontology), RSMO (Reference Software Measurement Ontology) and RASM (Reference Architecture for Software Measurement). Based on RASM it is possible to define software measurement architectures considering the use of specific technologies.

The proposal made by Mora and colleagues [16, 18, 19, 20] (the P01 proposal in Table 1) also defines a measurement architecture by using a strategy in levels. However, the proposal is limited to some measures that can be automatically collected

and does not support SPC, i.e., high maturity software measurement. Besides, the proposal does not aim to help organizations define new architectures. LASMA, in turn, aims to support the definition of measurement architectures, does not restrict the measures to be used, and, since it is based on the RSMO, it addresses both, traditional and high maturity software measurement.

LASMA was used to define a specific measurement architecture and a tool (M&SPC), which showed us that the proposed approach is viable.

It is worthy pointing out that although LASMA is proposed to the software measurement domain, it can be applied to other domains. For instance, using UFO as a basis, a reference ontology could be defined for the software testing domain. This ontology would be used as a basis to define a reference architecture for the software testing domain that, in turn, would be used for defining specific architectures and tools.

Currently, considering the results of the mapping study, we are working on the use of RSMA for defining a specific measurement architecture using technologies such as data warehouse, ETL (Extract, transform, load) and OLAP (Online Analytic Processing) cubes. By doing this we intend to analyze the difficulties and facilities of using RSMA to define an architecture using more complex technologies. In addition to that, as said before, we are planned new experimental studies aiming to verify if the tool M&SPC is capable of properly supporting the measurement process. By evaluating the tool we are also evaluating the reference architecture defined and the use of LASMA for defining measurement architectures [40].

Acknowledges

This research is funded by the Brazilian Research Funding Agencies FAPES (Process Number 52272362/11) and CNPq (Process Number 485368/2013-7).

References

[1] Mcgarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., Hall, F.: Practical Software Measurement: Objective Information for Decision Makers. Addison Wesley, Boston, USA (2002).

[2] CMMI Product Team: CMMI for Development, Version 1.3., http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm, (2010).

[3] ISO/IEC: ISO/IEC 15939, Systems and Software Engineering – Measurement Process (2007).

[4] ISO/IEC: ISO/IEC 12207, Systems and Software Engineering – Software Life Cycle Processes, Second edition (2008).

[5] SOFTEX: MPS.BR: Melhoria de Processo do Software Brasileiro - Guia Geral, http://www.softex.br/mpsbr, (2012).

[6] Zachman, J.: A framework for information systems architecture. IBM Systems Journal. 276–292 (1987).

[7] Bernstein, P. A.: Repositories and Object-Oriented Databases, Proceedings of the BTW Conference, pp 34–46 (1997).

[8] Dumke, R., Ebert, C.: Software Measurement: Establish – Extract – Evaluate – Execute. Springer-Verlag (2010).

[9] Maretto, C. X., Barcellos, M. P.: Software Measurement Architectures: A Mapping Study, In Proceedings of the 10th ESELAW: Experimental Software Engineering Latin American Workshop, Montevideo, pp 20-33 (2013).
[10] Barcellos, M., Falbo, R. A., Rocha, A. R.: Establishing a Well-Founded Conceptualization about Software Measurement in High Maturity Levels. In Proceedings of the 7th International Conference on the Quality of Information and Communications Technology (QUATIC 2010), Oporto, Portugal, pp 467–472 (2010).
[11] Barcellos, M. P., Falbo, R. A., Rocha, A. R. A.: A Well-Founded Software Process Behavior Ontology to Support Business Goals Monitoring in High Maturity Software Organizations. In Proceedings of the IEEE 5th Joint VORTE-MOST Workshop, Vitória, Brazil, pp 253–262 (2010).
[12] Florac, W. A., Carleton, A. D.: Measuring the Software Process: Statistical Process Control for Software Process Improvement. Addison Wesley, Boston, USA (1999).
[13] Tarhan, A., Demirors, O.: Apply Quantitative Management Now. IEEE Software. 29, 77–85 (2012).
[14] Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01. Keele, Departament of Computer Science Keele University (2007).
[15] Montoni, M.: Investigation Regarding Critical Success Factors in Software Process Improvement Initiatives, Computation and Systems Engineering Program, Doctorate Thesis, Federal University of Rio de Janeiro, COPPE/UFRJ, Rio de Janeiro, Brazil. (2010) (in Portuguese only)
[16] M Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J.Á., Ramos, I.: Software generic measurement framework based on MDA. IEEE Latin America Transactions. 9, 130–137 (2011).
[17] Palza, E., Fuhrman, C., Abran, A., Ouest, N., Québec, H.C.M.: Establishing a Generic and Multidimensional Measurement Repository in CMMI context. In Proceedings of the 28th Annual NASA Goddard Software Engineering Workshop (2003).
[18] Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J.Á., Ramos, I.: Software generic measurement framework based on MDA. IEEE Latin America Transactions. 6, 363–370 (2008).
[19] Mora, B., García, F., Ruiz, F., Piattini, M., Boronat, A., Gómez, A., Carsí, J.Á., Ramos, I.: Software generic measurement framework based on MDA. Latin America Transactions IEEE. 8, 605–613 (2010).
[20] Mora, B., Garcia, F., Ruiz, F., Piattini, M.: Model-Driven Software Measurement Framework: A Case Study. In Proceedings of the Ninth International Conference on Quality Software, pp 239–248 (2009).
[21] Basili, V. R., Caldeira, G., Rombach, H. D.: The goal question metric approach, Encyclopedia of Software Engineering, Wiley. (1994).
[22] Aversano, L., Bodhuin, T., Canfora, G., Tortorella, M.: WebEv - a Collaborative Environment for Supporting Measurement Frameworks. In Proceedings of the 37th Annual Hawaii International Conference, pp 1–10 (2004).

[23] Aversano, L., Bodhuin, T., Canfora, G., Tortorella, M.: A Framework for Measuring Business Processes based on GQM. In Proceedings of the 37th Annual Hawaii International Conference, pp 1–10 (2004).
[24] Goth, G.: focus NSDIR: A Legacy beyond Failure. IEEE Software. 18, 53–56 (2001).
[25] Bastani, F. B., Ntafos, S., Harris, D. E., Morrow, R. R., Paul, R.: A high-assurance measurement repository system. In Proceedings of the Fifth IEEE International Symposium on High Assurance Systems Engineering (HASE 2000), pp 265–272 (2000).
[26] Silveira, P. S., Becker, K., Ruiz, D. D.: SPDW+: a seamless approach for capturing quality metrics in software development environments. Software Quality Journal. 18, 227–268 (2010).
[27] Harrison, W.: A flexible method for maintaining software metrics data: a universal metrics repository. Journal of Systems and Software. 72, 225–234 (2004).
[28] Paul, R.A., Kunii, T.L., Shinagawa, Y., Khan, M.F.: Software Metrics Knowledge and Databases for Project Management. IEEE Transactions on Knowledge and Data Engineering, 11, 255–264 (1999).
[29] Fondement, F., Silaghi, R.: Defining Model Driven Engineering Processes. In: Proceedings of the Third International Workshop in Software Model Engineering (WiSME), Lisbon, Portugal, pp 1–11 (2004).
[30] OMG, 2003, MDA Guide Version 1.0.1. Available: http://www.enterprise-architecture.info. Access: 3 jun. 2012.
[31] Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Universal Press, The Netherlands, ISBN 90-75176-81-3 (2005).
[32] Guizzardi, G., Falbo, R. A., Guizzardi, R. S. S.: Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology. In Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments, pp 244-251 (2008).
[33] Bringuente, A., Falbo, R. A., Guizzardi, G.: Using a Foundational Ontology for Reengineering a Software Process Ontology. In Proceedings of the XXVI Brazilian Symposium on Data Base, Florianópolis, Brazil, pp 1-16 (2011).
[34] Barcellos, M. P., Falbo, R. A.: Using a foundational ontology for reengineering a software enterprise ontology. In Joint International Workshop on Metamodels, 5833, pp 179-188 (2009).
[35] Falbo, R. A., Nardi, J. C.: Evolving a Software Requirements Ontology. In Proceedings of the XXXIV Conferencia Latinoamericana de Informática, Santa Fe, Argentina, pp 300-309 (2008).
[36] Barcellos, M. P., Falbo, R. A., Rocha, A. R.: A strategy for preparing software organizations for statistical process control. Journal of the Brazilian Computer Society, DOI 10.1007/s13173-013-0106-x. (2013).
[37] Barcellos, M. P., Falbo, R. A., Dalmoro, R.: A well-founded software measurement ontology. In Proceedings of the 6th International Conference on Formal Ontology in Information Systems (FOIS 2010), Toronto, Canada, 209, pp 213-226 (2010).
[38] Guizzardi, G.: On Ontology, ontologies, Conceptualizations, Modeling Languages and (Meta)Models, In Proceedings of the conference on Databases

and Information Systems IV: Selected Papers from the Seventh International Baltic Conference, Amsterdam, Netherlands, pp 18-39 (2007).
[39] Guarino, N.: Formal Ontology and Information Systems. In Proceedings of International Conference in Formal Ontology and Information Systems, pp 3-15 (1998).
[40] Muller, G.: A reference architecture primer. Gaudí Project (white paper). Buskerud University College, Kongsberg, Noruega, pp 1-21 (2013).
[41] Nakagawa, E. Y., Barbosa, E. F., Maldonado, J. C.: Exploring ontologies to support the establishment of reference architectures: An example on software testing. In: Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture 2009 & European Conference on Software Architecture 2009, Cambridge, United Kingdom, pp 249-252 (2009).
[42] Oates, B. J.: Researching Information Systems and Computing, SAGE Publications (2006).